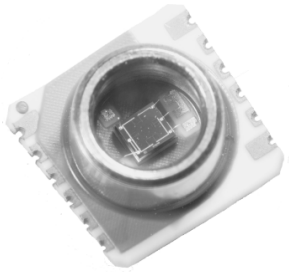
MS5534B BAROMETER MODULE



- **10 - 1100 mbar absolute pressure range**
- **6 coefficients for software compensation stored on-chip**
- **Piezoresistive silicon micromachined sensor**
- **Integrated miniature pressure sensor 9 x 9 mm**
- **15 Bit ADC**
- **3-wire serial interface**
- **1 system clock line (32.768 kHz)**
- **Low voltage and low power consumption**
- **RoHS-compatible & Pb-free[1]**

DESCRIPTION

The MS5534B is a SMD-hybrid device including a piezoresistive pressure sensor and an ADC-Interface IC. It provides a 16 Bit data word from a pressure and temperature dependent voltage. Additionally the module contains 6 readable coefficients for a highly accurate software calibration of the sensor. MS5534B is a low power, low voltage device with automatic power down (ON/OFF) switching. A 3-wire interface is used for all communications with a microcontroller. Sensor packaging options are plastic or metal cap.
Compared to the previous version (MS5534A) the temperature range has been improved (-40 to +125°C) as well as the pressure range (measurement down to 10 mbar). Other improvements concern the ESD sensitivity, current consumption and converter accuracy.

FEATURES

- Supply voltage 2.2 V to 3.6 V
- Low supply current
- -40°C to +125°C operation temperature
- No external components required

APPLICATIONS

- Mobile altimeter / barometer systems
- Weather control systems
- Adventure or multi-mode watches
- GPS receivers

BLOCK DIAGRAM

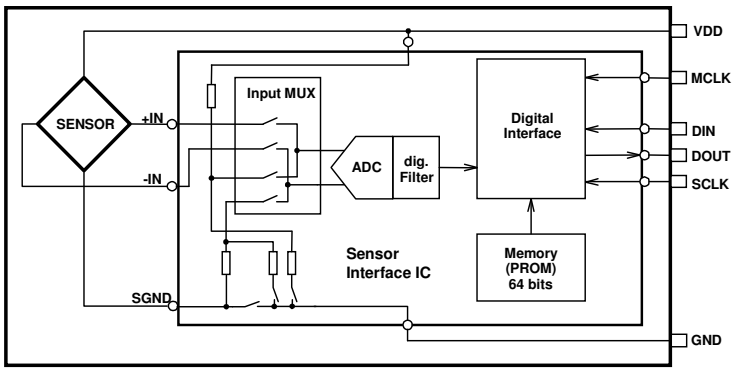


Fig. 1: Block diagram 5534B.

[1] The European RoHS directive 2002/95/EC (Restriction of the use of certain Hazardous Substances in electrical and electronic equipment) bans the use of lead, mercury, cadmium, hexavalent chromium and polybrominated biphenyls (PBB) or polybrominated diphenyl ethers (PBDE).

PIN CONFIGURATION

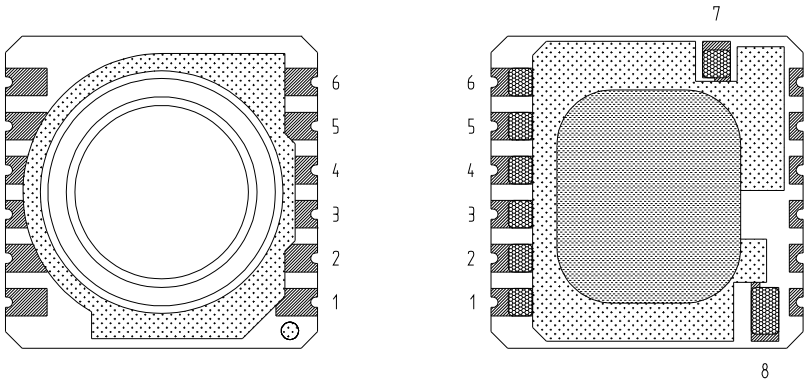


Fig. 2: Pin configuration of MS5534B.

Pin Name	Pin	Type	Function
GND	1	G	Ground
SCLK	2	I	Serial data clock
DOUT	3	O	Data output
DIN	4	I	Data input
MCLK	5	I	Master clock (32.768 kHz)
VDD	6	P	Positive supply voltage
PEN (1)	7	I	Programming enable
PV (1)	8	N	Negative programming voltage

NOTES

1) Pin 7 (PEN) and PIN 8 (PV) are only used by the manufacturer for calibration purposes and should not be connected.

ABSOLUTE MAXIMUM RATINGS

Parameter	Symbol	Conditions	Min	Max	Unit	Notes
Supply voltage	VDD	Ta = 25 °C	-0.3	4	V	
Storage temperature	T_S		-40	+125	°C	1
Overpressure	P	Ta = 25 °C		10	bar	2

NOTES

1) Storage and operation in an environment of dry and non-corrosive gases.
2) The MS5534BM is qualified referring to the ISO Standard 2281 and can withstand an absolute pressure of 11 bar in salt water or 100 m water respectively.

RECOMMENDED OPERATING CONDITIONS

(Ta = 25°C, VDD = 3.0 V unless noted otherwise)

Parameter	Symbol	Conditions	Min.	Typ.	Max	Unit
Operating pressure range	p		10		1100	mbar abs.
Supply voltage	VDD		2.2	3.0	3.6	V
Supply current, average (1) during conversion (2) standby (no conversion)	 I_{avg} I_{sc} I_{ss}	VDD = 3.0 V		 4 1	 0.1	 µA mA µA
Current consumption into MCLK (3)		MCLK = 32.768 kHz			0.5	µA
Operating temperature range	T		-40		+125	°C
Conversion time	t_{conv}	MCLK = 32.768 kHz			35	ms
External clock signal (4)	MCLK		30.000	32.768	35.000	kHz
Duty cycle of MCLK			40/60	50/50	60/40	%
Serial data clock	SCLK				500	kHz

NOTES

1) Under the assumption of one conversion every second. Conversion means either a pressure or a temperature measurement started by a command to the serial interface of MS5534B.
2) During conversion the sensor will be switched on and off in order to reduce power consumption; the total on time within a conversion is about 2 ms.
3) This value can be reduced by switching off MCLK while MS5534B is in standby mode.
4) It is strongly recommended that a crystal oscillator be used because the device is sensitive to clock jitter. A square-wave form of the clock signal is a must.

ELECTRICAL CHARACTERISTICS

DIGITAL INPUTS

(T = -40°C .. 125°C, VDD = 2.2 V .. 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Input High Voltage	V_{IH}		80% VDD		100% VDD	V
Input Low Voltage	V_{IL}		0% VDD		20% VDD	V
Signal Rise Time	t_r			200		ns
Signal Fall Time	t_f			200		ns

DIGITAL OUTPUTS

(T = -40°C .. 125°C, VDD = 2.2 V .. 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Output High Voltage	V_{OH}	I_{source} = 0.6 mA	75% VDD		100% VDD	V
Output Low Voltage	V_{OL}	I_{sink} = 0.6 mA	0% VDD		20% VDD	V
Signal Rise Time	t_r			200		ns
Signal Fall Time	t_f			200		ns

AD-CONVERTER

(T = -40°C .. 125°C, VDD = 2.2 V .. 3.6 V)

Parameter	Symbol	Conditions	Min	Typ	Max	Unit
Resolution				15		Bit
Linear Range			0		40000	LSB
Conversion Time		MCLK = 32.768 kHz			35	ms
INL		Within linear range	-4		+4	LSB

PRESSURE OUTPUT CHARACTERISTICS

With the calibration data stored in the interface IC of the MS5534B the following characteristics can be achieved:

(VDD = 3.0 V unless noted otherwise)

Parameter	Conditions	Min	Typ	Max	Unit	Notes
Resolution			0.1		mbar	1
Absolute Pressure Accuracy	p = 750 .. 1100 mbar T_a = 25°C	-1.5		+1.5	mbar	2
Relative Pressure Accuracy	p = 750 .. 1100 mbar T_a = 25°C	-0.5		+0.5	mbar	3
Error over Temperature	T = 0 .. +70°C p = const.	-2		+2	mbar	4
	T = -40 .. +85°C p = const.		± 2		mbar	4
	T = -40 .. +125°C p = const.		± 3		mbar	4
Long-term Stability	12 months		-1		mbar	5
Maximum Error over Supply Voltage	VDD = 2.2 .. 3.6 V p = const.	-1.6		+1.6	mbar	

NOTES

1) A stable pressure reading of the given resolution requires taking the average of 2 to 4 subsequent pressure values due to noise of the ADC.
2) Maximum error of pressure reading over the pressure range.
3) Maximum error of pressure reading over the pressure range after offset adjustment at one pressure point.
4) With the second-order temperature compensation as described in Section “FUNCTION". See next section for typical operating curves.
5) The long-term stability is measured with non-soldered devices.

TEMPERATURE OUTPUT CHARACTERISTICS

This temperature information is not required for most applications, but it is necessary to allow for temperature compensation of the pressure output.

(VDD = 3.0 V unless noted otherwise)

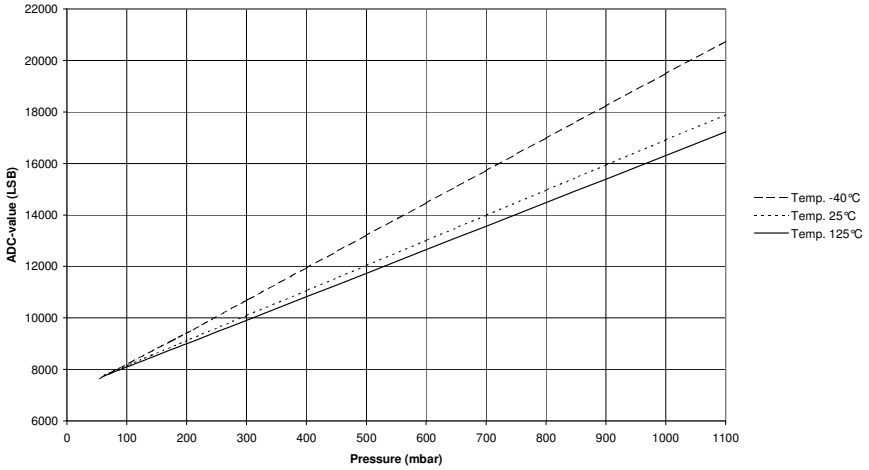
Parameter	Conditions	Min	Typ	Max	Unit	Notes
Resolution		0.005		0.015	°C	
Accuracy	T = 20°C	-0.8		0.8	°C	
	T = -40 .. +125°C		± 2		°C	1
Maximum Error over Supply Voltage	VDD = 2.2 .. 3.6 V	-0.2		+ 0.2	°C	

NOTES

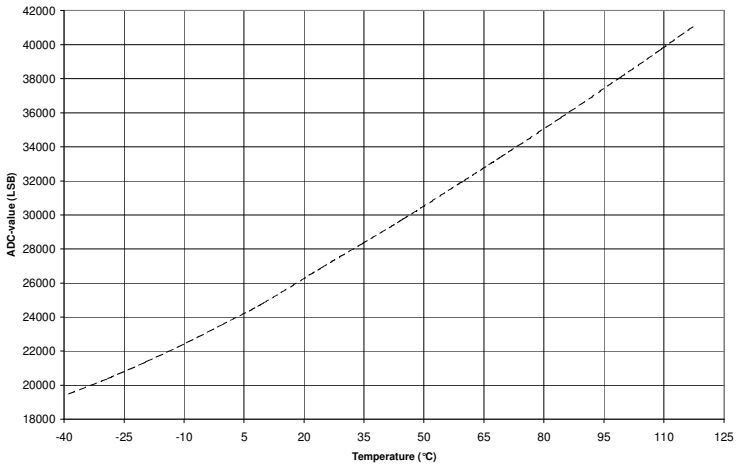
1) With the second-order temperature compensation as described in Section “FUNCTION". See next section for typical operating curves.

TYPICAL PERFORMANCE CURVES

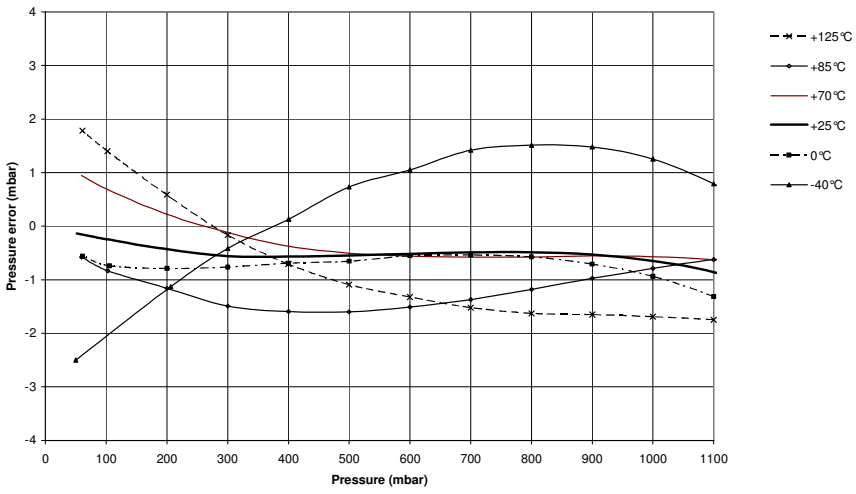
ADC-value D1 vs Pressure (typical)



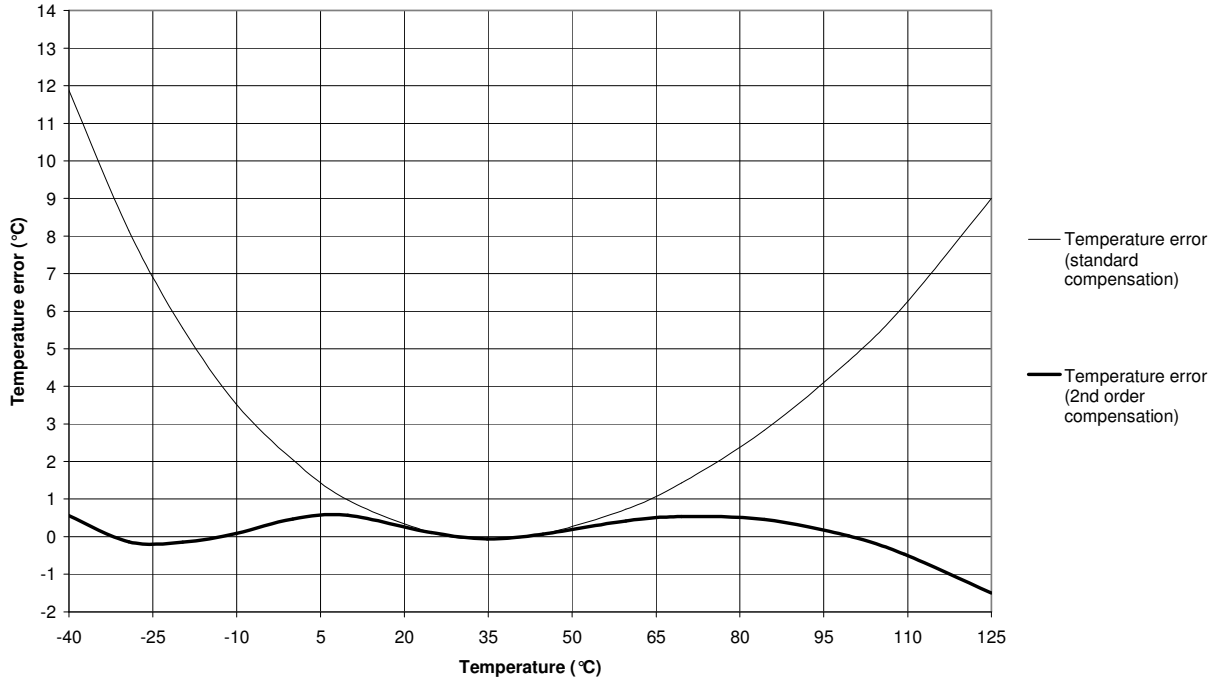
ADC-value D2 vs Temperature (typical)



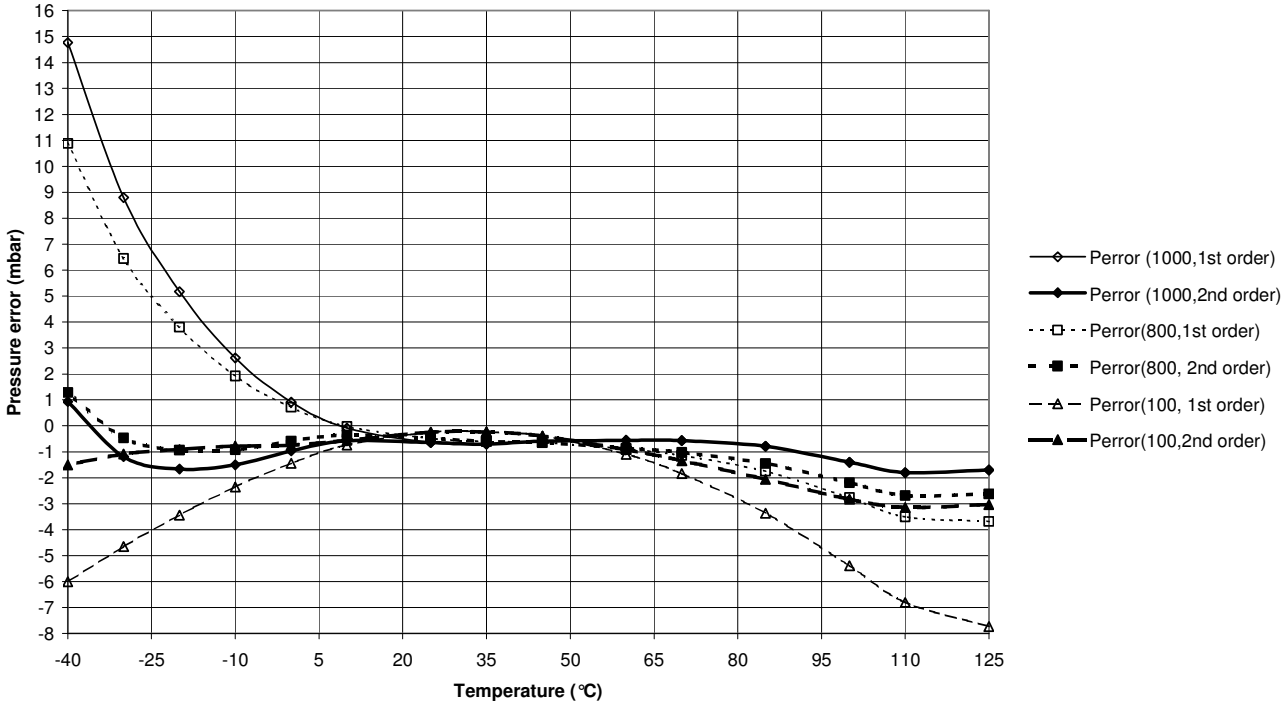
Absolute pressure accuracy after calibration (typical)



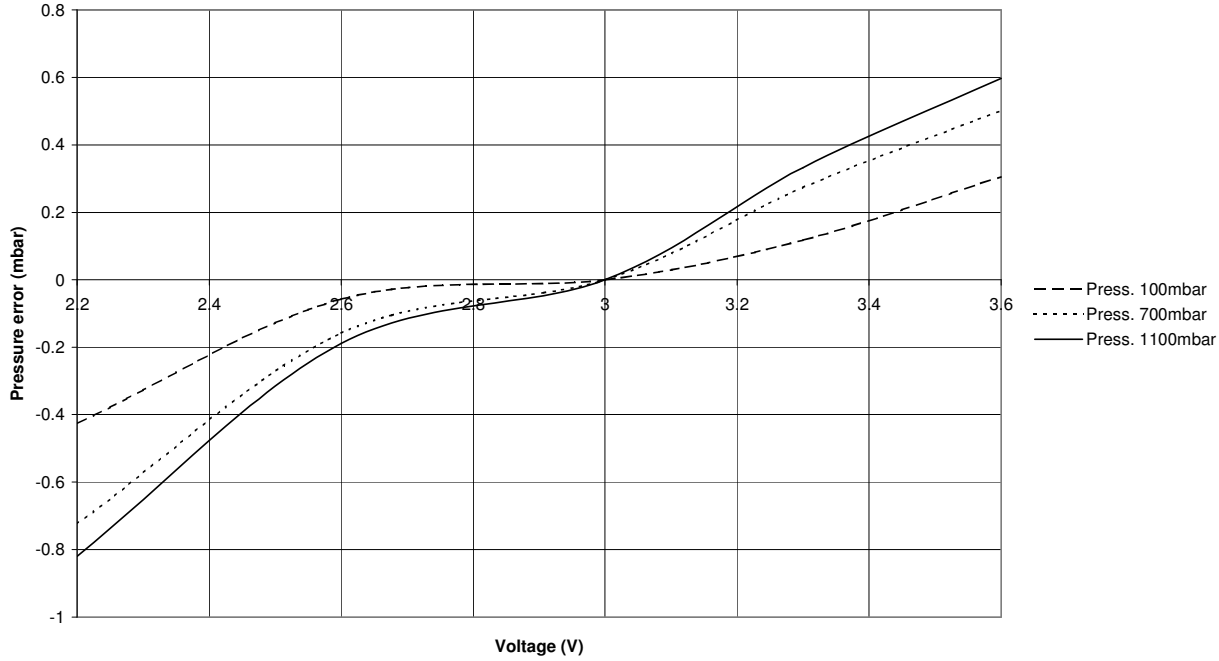
Accuracy vs temperature (typical)



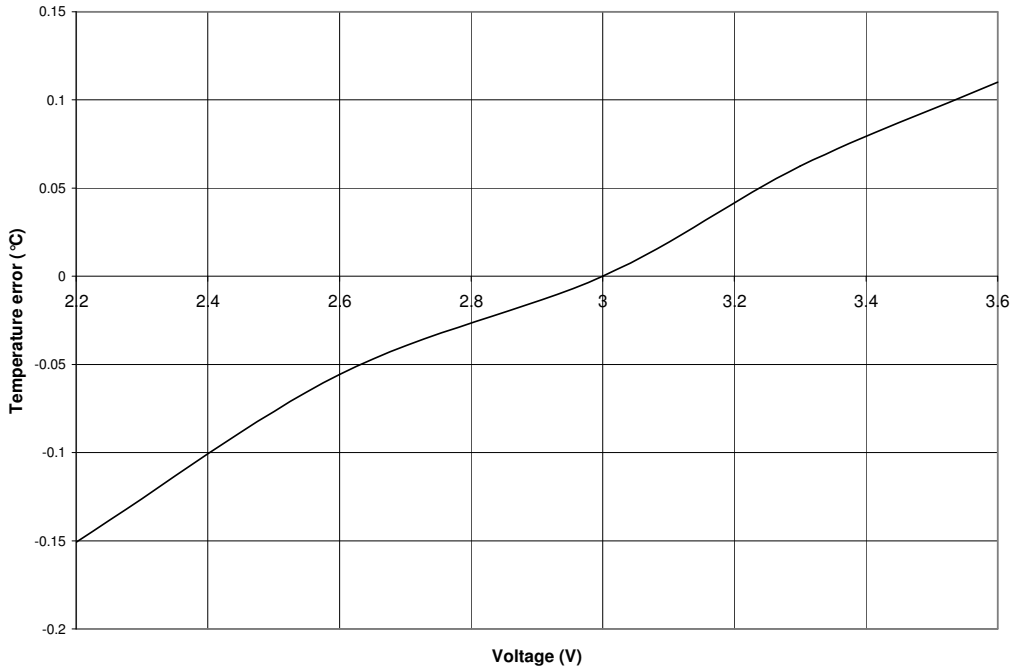
Accuracy vs temperature (typical)



Pressure error vs supply voltage (typical)



Temperature error (25°C) vs supply voltage (typical)



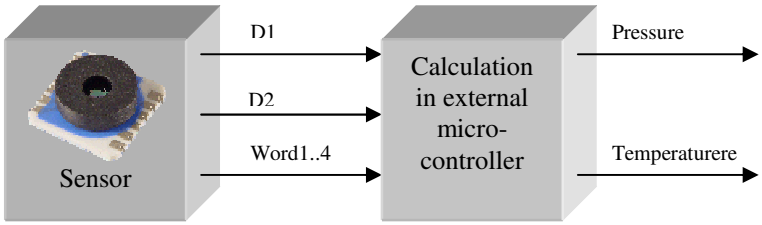
FUNCTION

GENERAL

The MS5534B consists of a piezoresistive sensor and a sensor interface IC. The main function of the MS5534B is to convert the uncompensated analogue output voltage from the piezoresistive pressure sensor to a 16-Bit digital value, as well as providing a 16-Bit digital value for the temperature of the sensor.

- measured pressure (16-Bit) "D1"
- measured temperature (16-Bit) "D2"

As the output voltage of a pressure sensor is strongly dependent on temperature and process tolerances, it is necessary to compensate for these effects. This compensation procedure must be performed by software using an external microcontroller.



For both pressure and temperature measurement the same ADC is used (sigma delta converter):

- for the pressure measurement, the differential output voltage from the pressure sensor is converted
- for the temperature measurement, the sensor bridge resistor is sensed and converted

During both measurements the sensor will only be switched on for a very short time in order to reduce power consumption. As both, the bridge bias and the reference voltage for the ADC are derived from VDD, the digital output data is independent of the supply voltage.

FACTORY CALIBRATION

Every module is individually factory calibrated at two temperatures and two pressures. As a result, 6 coefficients necessary to compensate for process variations and temperature variations are calculated and stored in the 64-Bit PROM of each module. These 64-Bit (partitioned into four words of 16-Bit) must be read by the microcontroller software and used in the program converting D1 and D2 into compensated pressure and temperature values.

PRESSURE AND TEMPERATURE MEASUREMENT

The sequence of reading pressure and temperature as well as of performing the software compensation is depicted in Fig. 3 and Fig. 5.
First the Word1 to Word4 have to be read through the serial interface. This can be done once after reset of the microcontroller that interfaces to the MS5534B. Next the compensation coefficients C1 to C6 are extracted using Bit-wise logical- and shift-operations (refer to Fig. 4 for the Bit-pattern of Word1 to Word4).
For the pressure measurement the microcontroller has to read the 16 Bit values for pressure (D1) and temperature (D2) via the serial interface in a loop (for instance every second). Then, the compensated pressure is calculated out of D1, D2 and C1 to C6 according to the algorithm in Fig. 3 (possibly using quadratic temperature compensation according to Fig. 5). All calculations can be performed with signed 16-Bit variables. Results of multiplications may be up to 32-Bit long (+sign). In the flow according to Fig. 3 each multiplication is followed by a division. This division can be performed by Bit-wise shifting (divisors are to the power of 2). It is ensured that the results of these divisions are less than 65536 (16-Bit).

For the timing of signals to read out Word1 to Word4, D1, and D2 please refer to the paragraph "Serial Interface".

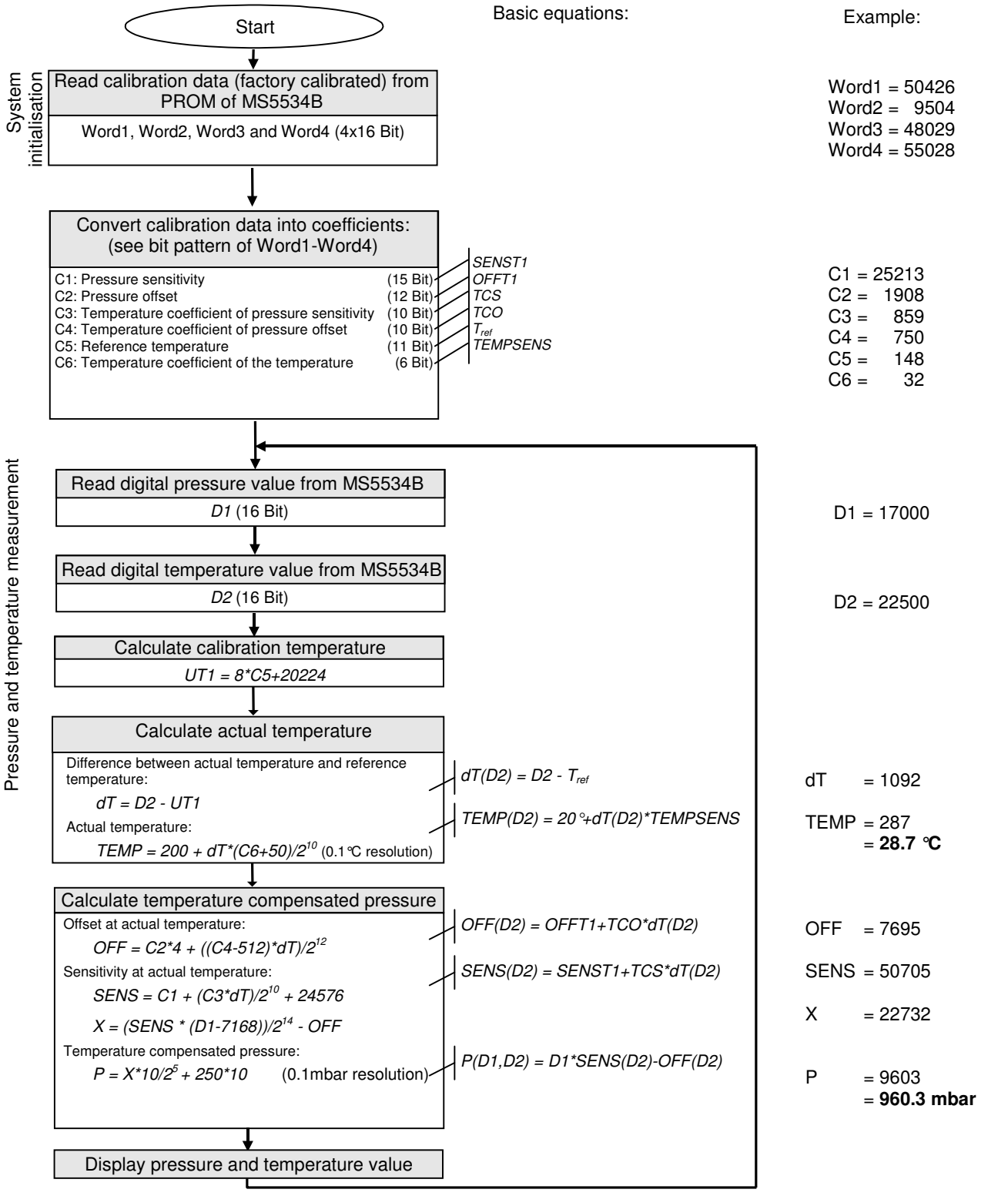


Fig. 3: Flow chart for pressure and temperature reading and software compensation.

NOTES

1) Readings of D2 can be done less frequently, but the display will be less stable in this case.
2) For a stable display of 0.1 mbar resolution, it is recommended to display the average of 8 subsequent pressure values.

	C1 (15 Bit)															C5/I 1 Bit
Word1	DB14	DB13	DB12	DB11	DB10	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DB10
	C5/II (10 Bit)										C6 (6 Bit)					
Word2	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DB5	DB4	DB3	DB2	DB1	DB0
	C4 (10 Bit)										C2/I (6 Bit)					
Word3	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DB11	DB10	DB9	DB8	DB7	DB6
	C3 (10 Bit)										C2/II (6-Bit)					
Word4	DB9	DB8	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0	DB5	DB4	DB3	DB2	DB1	DB0

Fig. 4: Arrangement (Bit-pattern) of calibration data in Word1 to Word4.

SECOND-ORDER TEMPERATURE COMPENSATION

In order to obtain best accuracy over the whole temperature range, it is recommended to compensate for the non-linearity of the output of the temperature sensor. This can be achieved by correcting the calculated temperature and pressure by a second order correction factor. The second-order factors are calculated as follows:

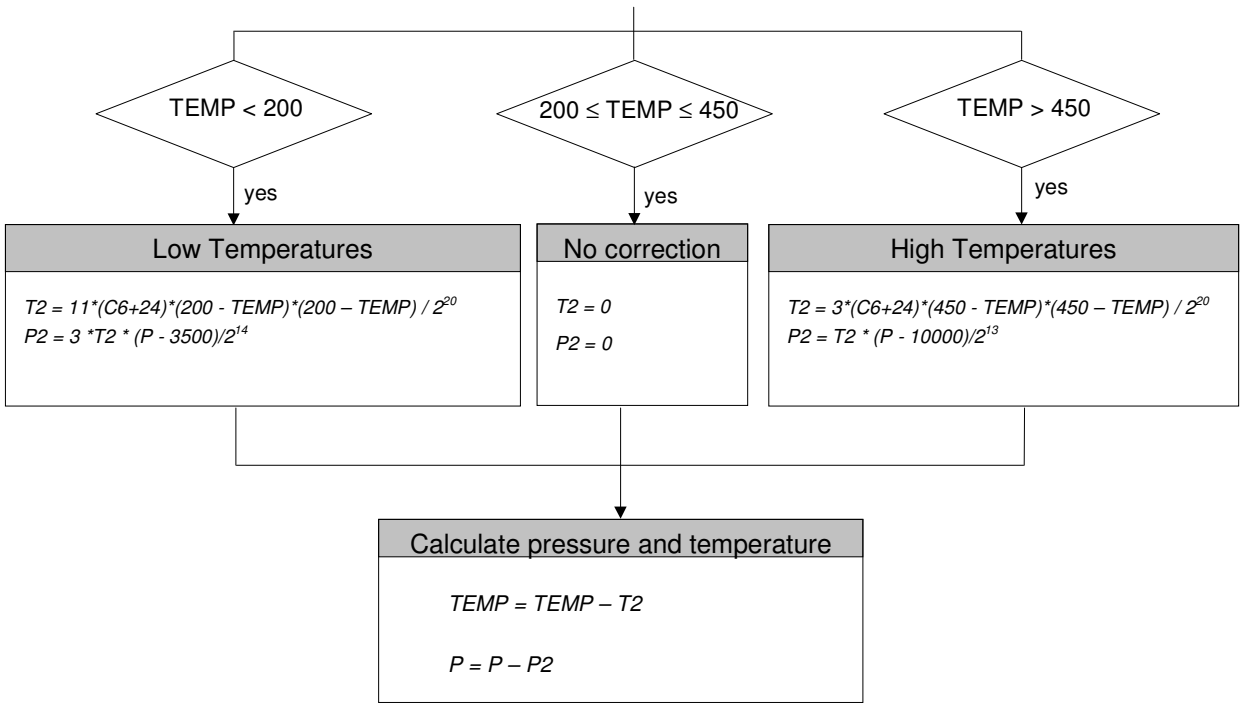


Fig. 5: Flow chart for calculating the temperature and pressure to the optimum accuracy.

SERIAL INTERFACE

The MS5534B communicates with microprocessors and other digital systems via a 3-wire synchronous serial interface as shown in Fig. 1. The SCLK (Serial clock) signal initiates the communication and synchronises the data transfer with each Bit being sampled by the MS5534B on the rising edge of SCLK and each Bit being sent by the MS5534B on the rising edge of SCLK. The data should thus be sampled by the microcontroller on the falling edge of SCLK and sent to the MS5534B with the falling edge of SCLK. The SCLK-signal is generated by the microprocessor's system. The digital data provided by the MS5534B on the DOUT pin is either the conversion result or the software calibration data. In addition the signal DOUT (Data out) is also used to indicate the conversion status (conversion-ready signal, see below). The selection of the output data is done by sending the corresponding instruction on the pin DIN (Data input).

Following is a list of possible output data instructions:

- Conversion start for pressure measurement and ADC-data-out "D1" (Figure 6a)
- Conversion start for temperature measurement and ADC-data-out "D2" (Figure 6b)
- Calibration data read-out sequence for Word1 (Figure 6c)
- Calibration data read-out sequence for Word2 (Figure 6d)
- Calibration data read-out sequence for Word3 (Figure 6c)
- Calibration data read-out sequence for Word4 (Figure 6d)
- RESET sequence (Figure 6e)

Every communication starts with an instruction sequence at pin DIN. Fig. 6 shows the timing diagrams for the MS5534B. The device does not need a 'Chip select' signal. Instead there is a START sequence (3-Bit high) before each SETUP sequence and STOP sequence (3-Bit low) after each SETUP sequence. The SETUP sequence consists in 4-Bit that select a reading of pressure, temperature or calibration data. In case of pressure- (D1) or temperature- (D2) reading the module acknowledges the start of a conversion by a low to high transition at pin DOUT during the last bit of the STOP sequence.
Two additional clocks at SCLK are required after the acknowledge signal. Then SCLK is to be held low by the microcontroller until a high to low transition on DOUT indicates the end of the conversion.
This signal can be used to create an interrupt in the microcontroller. The microcontroller may now read out the 16-Bit word by giving another 17 clocks on the SLCK pin. It is possible to interrupt the data READOUT sequence with a hold of the SCLK signal. **It is important to always read out the last conversion result before starting a new conversion.**

The RESET sequence is special as its unique pattern is recognised by the module in any state. By consequence it can be used to restart if synchronisation between the microcontroller and the MS5534B has been lost. This sequence is 21-Bit long. The DOUT signal might change during that sequence (see Fig. 6e). **It is recommended to send the RESET sequence before each CONVERSION sequence to avoid hanging up the protocol permanently in case of electrical interference.**

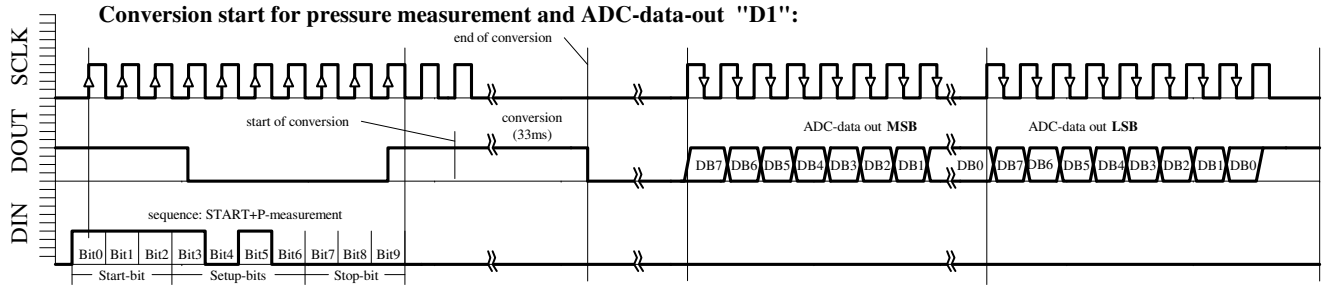


Fig. 6a: D1 ACQUISITION sequence.

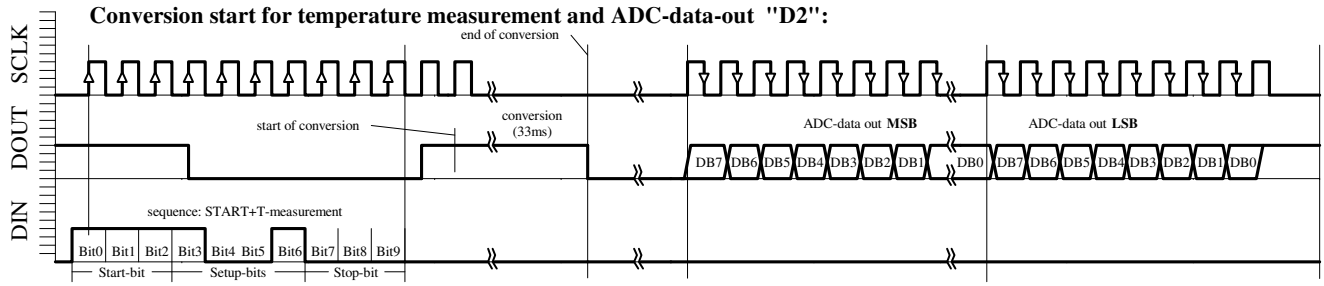


Fig. 6b: D2 ACQUISITION sequence.

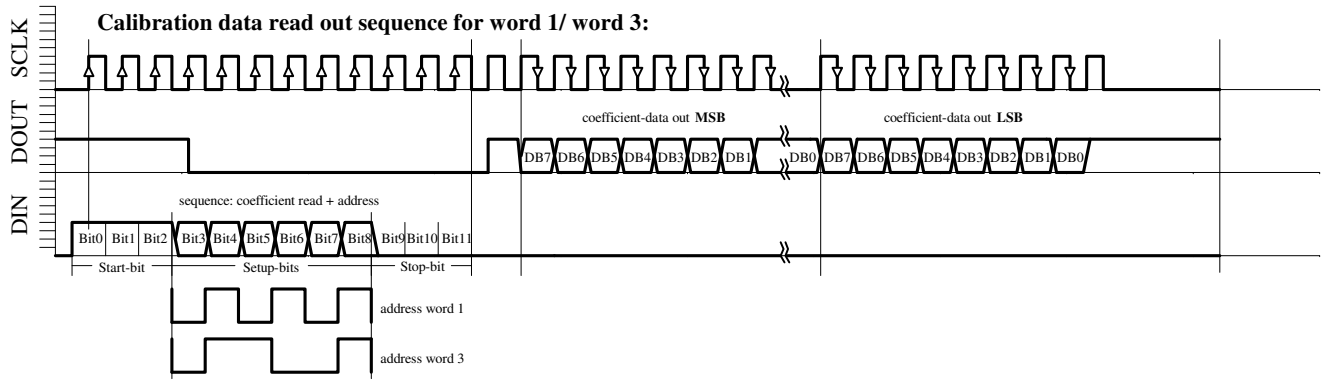


Fig. 6c: Word1, Word3 READING sequence.

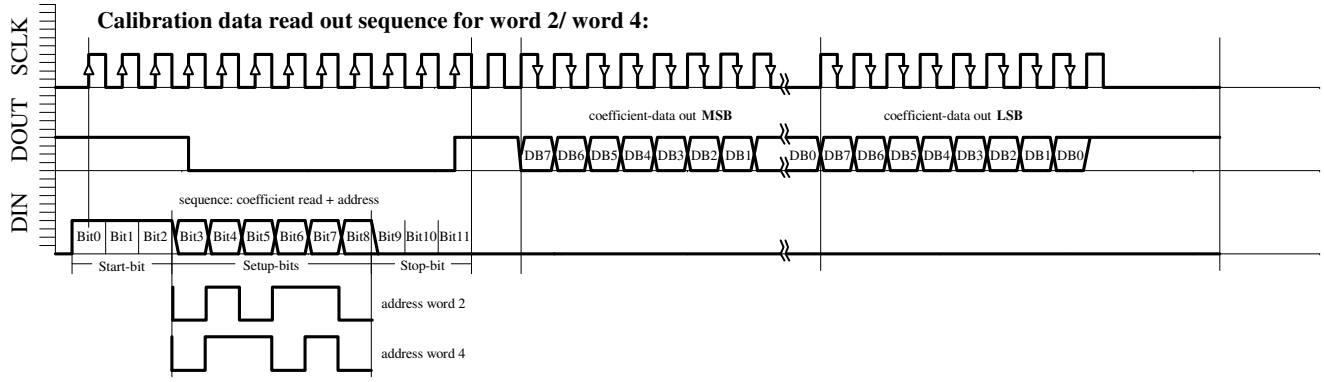


Fig. 6d: W2, W4 READING sequence.

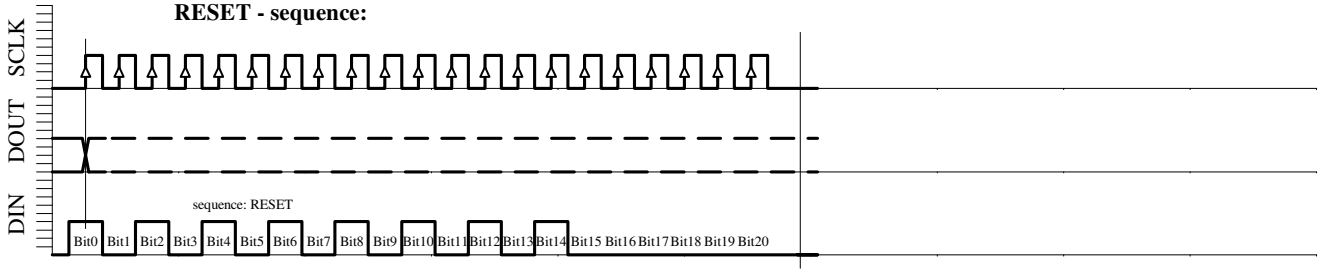


Fig. 6e: RESET sequence (21-Bit).

APPLICATION INFORMATION

GENERAL

The advantage of combining a pressure sensor with a directly adapted integrated circuit is to save other external components and to achieve a very low power consumption. The main application field for this system includes portable devices with battery supply, but its high accuracy and resolution make it also suited for industrial and automotive applications. The possibility to compensate the sensor with a software allows the user to adapt it to his particular application. Communication between the MS5534B and the widely available microcontrollers is realised over an easy-to-use 3-wire serial interface. Customers may select which microcontroller system to be used, and there are no specific standard interface cells required, which may be of interest for specially designed 4 Bit-microcontroller applications.

CALIBRATION

The MS5534B is factory calibrated. The calibration data is stored inside the 64-Bit PROM memory.

SOLDERING

Please refer to the application note AN808 for all soldering issues.

HUMIDITY, WATER PROTECTION

The silicon pressure transducer and the bonding wires are protected by a plastic protection cap on the standard version. MS5534BP is factory protected against humidity by a silicone gel.
The version MS5534BM carries a metal protection cap filled with silicone gel for enhanced protection against humidity. The properties of this gel ensure function of the sensor even when in direct water contact. This feature can be useful for waterproof watches or other applications, where direct water contact cannot be avoided. Nevertheless the user should avoid drying of hard materials like for example salt particles on the silicone gel surface. In this case it is better to rinse with clean water afterwards. Special care has to be taken to not mechanically damage the gel. Damaged gel could lead to air entrapment and consequently to unstable sensor signal, especially if the damage is close to the sensor surface.
The metal protection cap is fabricated of special anticorrosive stainless steel in order to avoid any corrosive battery effects inside the final product. The MS5534BM was qualified referring to the ISO Standard 2281 and can withstand a pressure of 11 bar in salt water. The concentration of the see water used for the qualification is 41 g of see salt for 1 litre of DI water.
For underwater operations as specified in ISO Standard 2281 it is important to seal the sensor with a rubber O-ring around the metal cap. Any salt water coming to the contact side (ceramic and pads) of the sensor could lead to permanent damage. Especially for "water-resistant 100 m" watches it is recommended to provide a stable mechanical pusher from the backside of the sensor. Otherwise the overpressure might push the sensor backwards and even bend the electronic board on which the sensor is mounted.

LIGHT SENSITIVITY

The MS5534B is sensitive to sunlight, especially to infrared light sources. This is due to the strong photo effect of silicon. As the effect is reversible there will be no damage, but the user has to take care that in the final product the sensor cannot be exposed to direct light during operation. This can be achieved for instance by placing mechanical parts with holes in such that light cannot pass.

CONNECTION TO PCB

The package outline of the module allows the use of a flexible PCB to connect it. This can be important for applications in watches and other special devices, and will also reduce mechanical stress on the device.
For applications subjected to mechanical shock, it is recommended to enhance the mechanical reliability of the solder junctions by covering the rim or the corners of MS5534B's ceramic substrate with glue or Globtop-like material.

DECOUPLING CAPACITOR

Particular care must be taken when connecting the device to power supply. A 47 µF tantalum capacitor **must** be placed as close as possible of the MS5534B's VDD pin. This capacitor will stabilise the power supply during data conversion and thus, provide the highest possible accuracy.

APPLICATION EXAMPLE: ALTIMETER SYSTEM USING MS5534B

MS5534B is a circuit that can be used in connection with a microcontroller in mobile altimeter applications. It is designed for low-voltage systems with a supply voltage of 3 V, particularly in battery applications. The MS5534B is optimised for low current consumption as the AD-converter clock (MCLK) can use the 32.768 kHz frequency of a standard watch crystal, which is supplied in most portable watch systems.

For applications in altimeter systems Intersema can deliver a simple formula to calculate the altitude, based on a linear interpolation, where the number of interpolation points influences the accuracy of the formula.

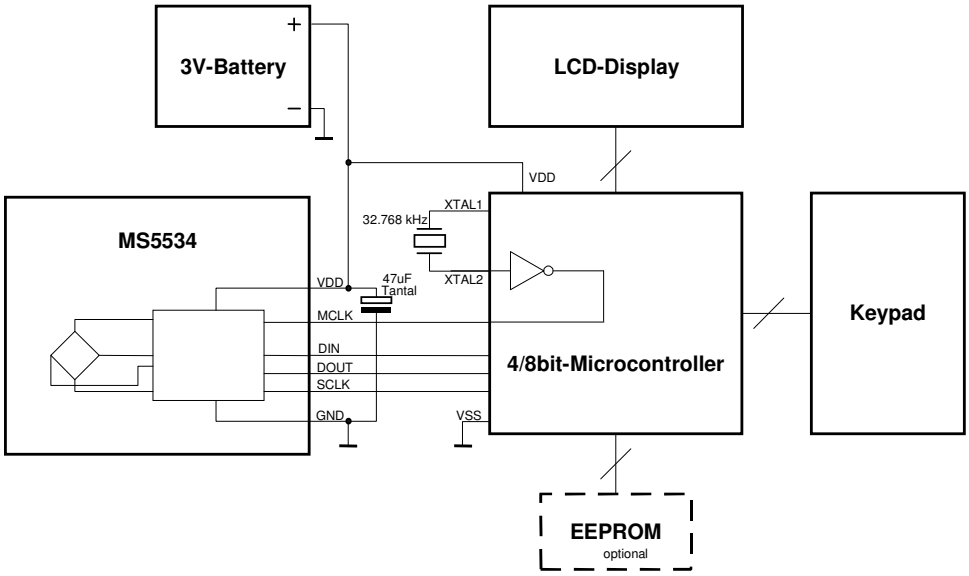


Figure 7: Demonstration of MS5534B in a mobile altimeter.

DEVICE PACKAGE OUTLINES

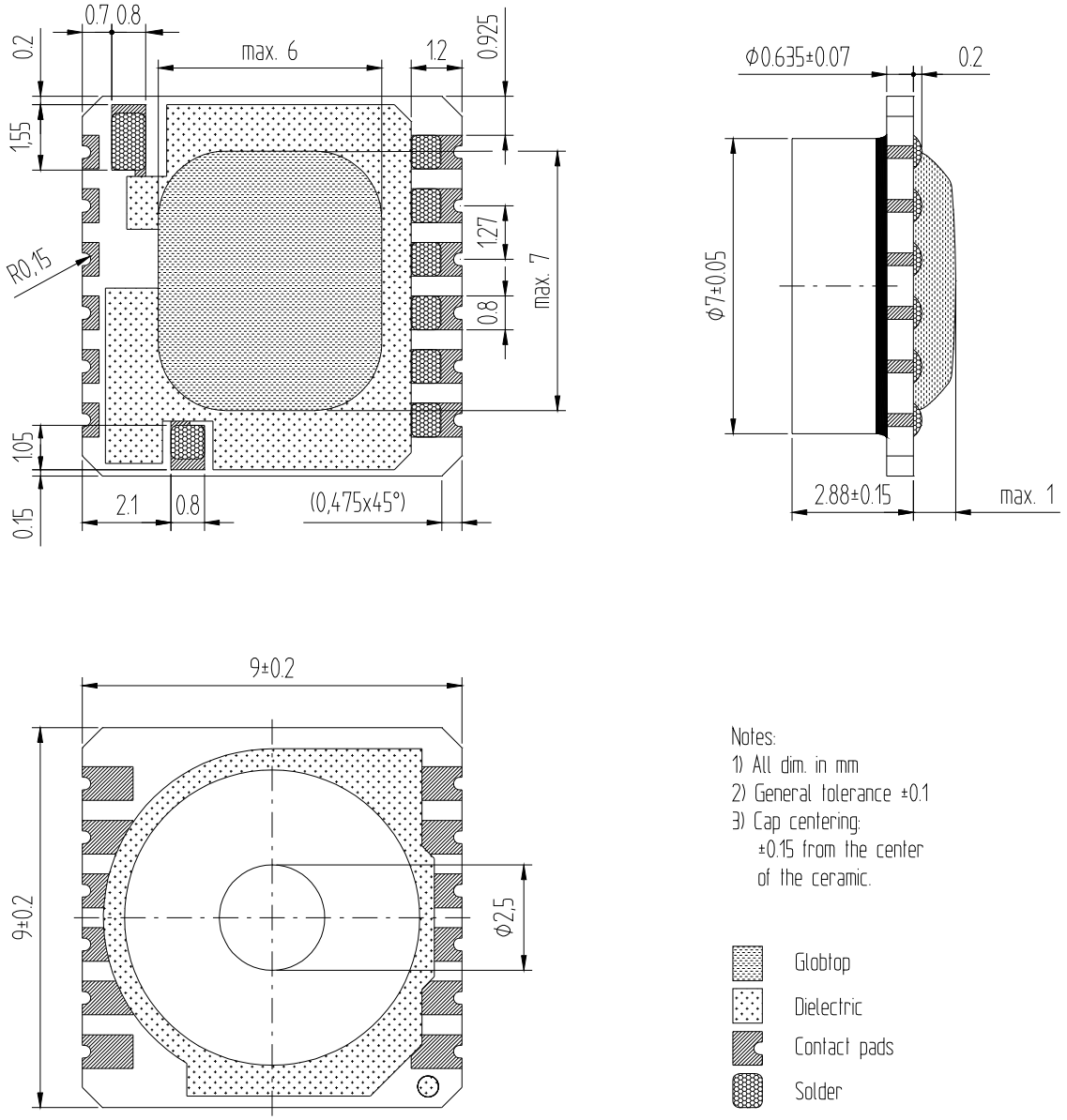


Fig. 8: Device package outlines of **MS5534BP**.

DEVICE PACKAGE OUTLINES (CONT.)

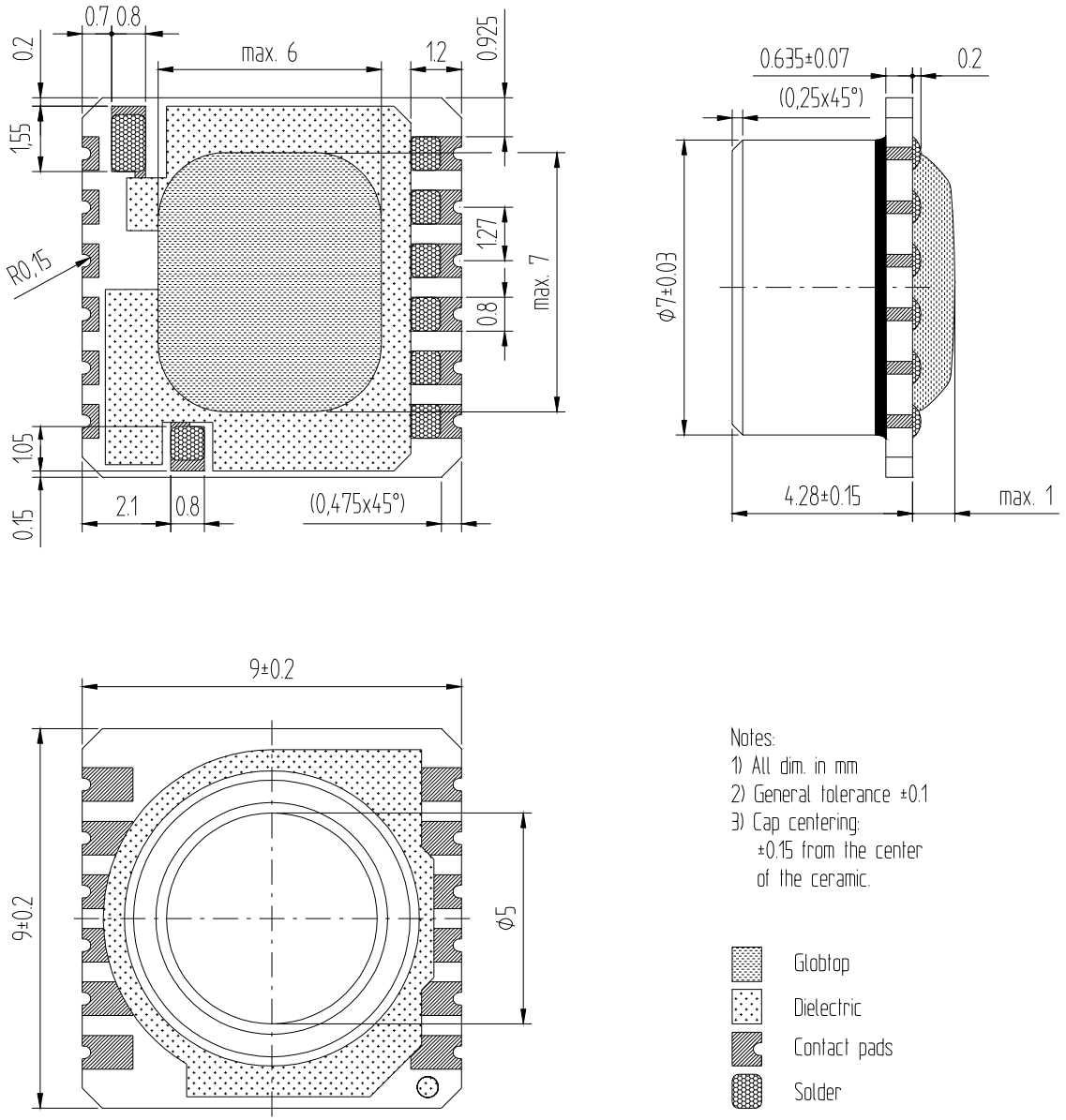
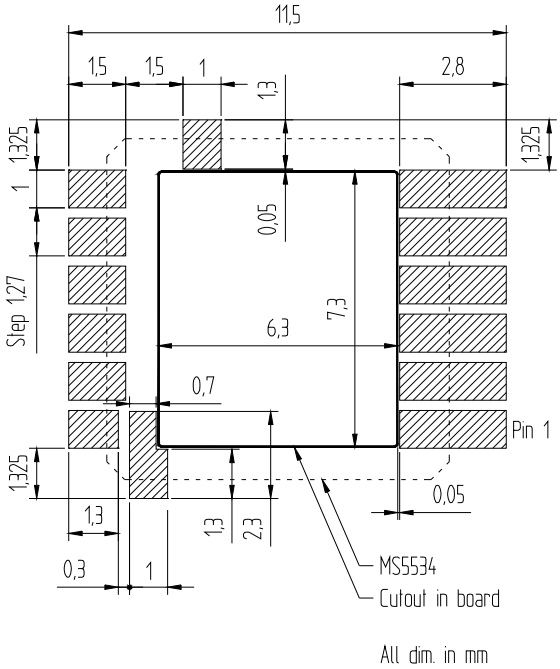


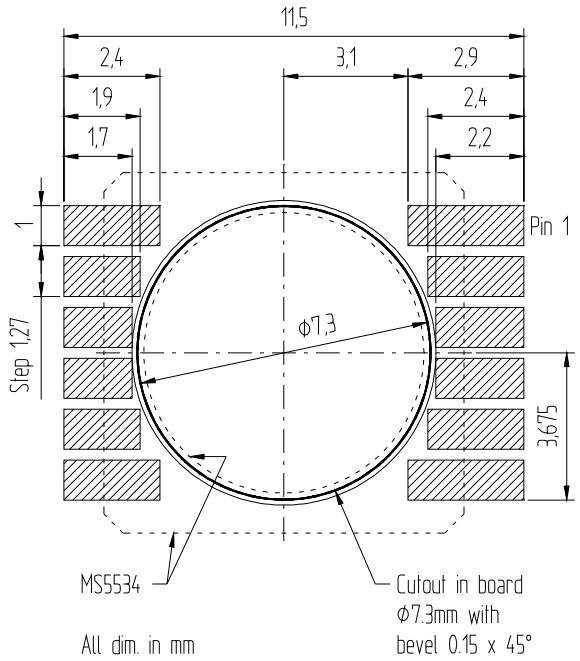
Fig. 9: Device package outlines of **MS5534BM**.

RECOMMENDED PAD LAYOUT

Pad layout for bottom side of MS5534B soldered onto printed circuit board



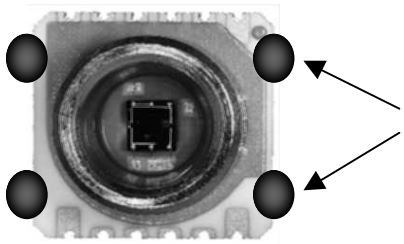
Pad layout for top side of MS5534B soldered onto printed circuit board



ASSEMBLY

MECHANICAL STRESS

It is recommended to avoid mechanical stress on the PCB on which the sensor is mounted. The thickness of the PCB should be not below 1.6 mm. A thicker PCB is more stiff creating less stress on the soldering contacts. For applications where mechanical stress cannot be avoided (for example ultrasound welding of the case or thin PCB's in watches) please fix the sensor with drops of low stress epoxy (for example Hysol FP-4401) at the corners of the sensor as shown below.



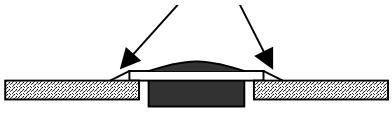
Fixing with Globtop increases mechanical stability

MOUNTING

The MS5534B can be placed with automatic Pick&Place equipment using vacuum nozzles. It will not be damaged by the vacuum. Due to the low stress assembly the sensor does not show pressure hysteresis effects. Special care has to be taken to not touch the protective gel of the sensor during the assembly.

The MS5534B can be mounted with the cap down or the cap looking upwards. In both cases it is important to solder all contact pads. The Pins PEN and PV shall be left open or connected to VDD. **Do not connect the Pins PEN and PV to GND!**

Solder at both sides to increase mechanical stability

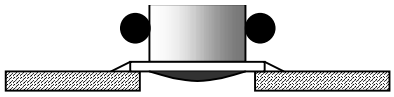


Placement cap down (hole in PCB to fit cap)

Placement cap up (rectangular hole in PCB to fit Globtop)

SEALING WITH O-RING

In products like outdoor watches the electronics must be protected against direct water or humidity. For those products the MS5534BM provides the possibility to seal with an O-ring. The protective cap of the MS5534BM is made of special anticorrosive stainless steel with a polished surface. In addition to this the MS5534BM is filled with silicone gel covering the sensor and the bonding wires. The O-ring (or O-rings) shall be placed at the outer diameter of the metal cap. This method avoids mechanical stress because the sensor can move in vertical direction.



CLEANING

The MS5534B has been manufactured under cleanroom conditions. Each device has been inspected for the homogeneity and the cleanness of the silicone gel. It is therefore recommended to assemble the sensor under class 10'000 or better conditions. Should this not be possible, it is recommended to protect the sensor opening during assembly from entering particles and dust. To avoid cleaning of the PCB, solder paste of type "no-clean" shall be used. **Cleaning might damage the sensor!**

ESD PRECAUTIONS

The electrical contacts except programming pads are protected against ESD according to 2 kV HBM (human body model). The PV programming pad is more sensitive due to the nature of the OTP programming cells that store the calibration coefficients. The breakdown voltage of PV is 1 kV. It is therefore essential to ground machines and personal properly during assembly and handling of the device. The MS5534B is shipped in antistatic transport boxes. Any test adapters or production transport boxes used during the assembly of the sensor shall be of an equivalent antistatic material.

ORDERING INFORMATION

Product Code	Product	Art.-Nr.	Package	Comments
MS5534-BP	Barometer Module with plastic cap	325534000	SMD hybrid with solder paste, plastic protection cap	Standard version
MS5534-BM	Barometer Module with metal cap	325534001	SMD hybrid with solder paste, metal protection cap, silicon gel sensor protection	Recommended for outdoor products

FACTORY CONTACTS

Intersema Sensoric SA
Ch. Chapons-des-Prés 11
CH-2022 BEVAIX

SWITZERLAND

Tel. 032 847 9550
Tel. Int. +41 32 847 9550
Telefax +41 32 847 9569
e-mail: sales@intersema.ch
http://www.intersema.ch

Specification of the Bluetooth System

Wireless connections made easy

Master Table of Contents & Compliance Requirements

Covered Core Package version:
2.0 + EDR
Current Master TOC issued:
4 November 2004

Bluetooth®

Revision History

The Revision History is shown in the Appendix.

Contributors

The persons who contributed to this specification are listed in the Appendix.

Web Site

This specification can also be found on the Bluetooth web site:
http://www.bluetooth.com

Disclaimer and Copyright Notice

RANTY ARISING OUT OF ANY COURSE OF DEALING, USAGE, TRADE PRACTICE, PROPOSAL, SPECIFICATION OR SAMPLE.

Each Member hereby acknowledges that products equipped with the Bluetooth® technology ("Bluetooth® Products") may be subject to various regulatory controls under the laws and regulations of various governments worldwide. Such laws and regulatory controls may govern, among other things, the combination, operation, use, implementation and distribution of Bluetooth® Products. Examples of such laws and regulatory controls include, but are not limited to, airline regulatory controls, telecommunications regulations, technology transfer controls and health and safety regulations. Each Member is solely responsible for the compliance by their Bluetooth® Products with any such laws and regulations and for obtaining any and all required authorizations, permits, or licenses for their Bluetooth® Products related to such regulations within the applicable jurisdictions. Each Member acknowledges that nothing in the Specification provides any information or assistance in connection with securing such compliance, authorizations or licenses. NOTHING IN THE SPECIFICATION CREATES ANY WARRANTIES, EITHER EXPRESS OR IMPLIED, REGARDING SUCH LAWS OR REGULATIONS.

ALL LIABILITY, INCLUDING LIABILITY FOR INFRINGEMENT OF ANY INTELLECTUAL PROPERTY RIGHTS OR FOR NONCOMPLIANCE WITH LAWS, RELATING TO USE OF THE SPECIFICATION IS EXPRESSLY DISCLAIMED. BY USE OF THE SPECIFICATION, EACH MEMBER EXPRESSLY WAIVES ANY CLAIM AGAINST BLUETOOTH SIG AND ITS PROMOTER MEMBERS RELATED TO USE OF THE SPECIFICATION.

Bluetooth SIG reserves the right to adopt any changes or alterations to the Specification as it deems necessary or appropriate.

MASTER TABLE OF CONTENTS

This table of contents (TOC) covers the entire Bluetooth Specification.
In addition each volume has a TOC and each part of a volume is preceded by a detailed TOC.

MASTER TOC FOR THE BLUETOOTH SPECIFICATION

In the following table:

- The TOC for each Volume starts at the top of a page.
- The Volume No. is followed by the name of the Volume written in red.

Note: Each Volume is a self contained book which is published and updated separately and is equipped with a TOC of its own. However, this Master TOC is also revised as soon as any of the other Volumes are updated.

- A Volume cover one or more Parts (A, B, etc.), each Part can be viewed independently and has its own TOC.

Red or blue text on the following pages indicates hypertext links that will take you directly to the indicated section, on condition that you have access to a complete specification.

Specification Volume 0
Master Table of Contents & Compliance Requirements

Part A
MASTER TABLE OF CONTENTS

Part B
BLUETOOTH COMPLIANCE REQUIREMENTS

Part C
APPENDIX

Specification Volume 1
Architecture & Terminology Overview

Specification Volume 2
Core System Package
[Controller volume]

Part B
BASEBAND SPECIFICATION

Part C
LINK MANAGER PROTOCOL

Contents

Part E
HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

Part F
MESSAGE SEQUENCE CHARTS

Part G
SAMPLE DATA

Part H
SECURITY SPECIFICATION

Specification Volume 3
Core System Package
[Host volume]

Part B
SERVICE DISCOVERY PROTOCOL (SDP)

Part D
TEST SUPPORT

Master Table of Contents & Compliance Requirements

Part B

BLUETOOTH COMPLIANCE REQUIREMENTS

This document specifies the requirements for Bluetooth® compliance.

CONTENTS

1 INTRODUCTION

The Bluetooth Qualification Program Reference Document (PRD) is the primary reference document for the Bluetooth Qualification Program and defines its requirements, functions, and policies. The PRD is available on the Bluetooth Web site.

Passing the Bluetooth Qualification Process demonstrates a certain measure of compliance and interoperability, but because products are not tested for every aspect of this Bluetooth Specification, qualification does not guarantee compliance. Passing the Bluetooth Qualification Process only satisfies one condition of the license grant. The Member has the ultimate responsibility to ensure that the qualified product complies with this Bluetooth Specification and interoperates with other products.

2 SCOPE

This part of the specification defines some fundamental concepts used in the Bluetooth Qualification Program.

3 DEFINITIONS

Bluetooth Qualification Process – The process defined in the Bluetooth Qualification Program Reference Document (PRD) to qualify a design used in implementations of Bluetooth wireless technology.

Bluetooth Qualification Program – The Bluetooth Qualification Process together with other related requirements and processes.

3.1 TYPES OF BLUETOOTH PRODUCTS

Bluetooth Product – Any product containing an implementation of Bluetooth wireless technology. All Bluetooth Products shall be one of the following:

- Bluetooth End Product
- Bluetooth Host Subsystem Product
- Bluetooth Controller Subsystem Product
- Bluetooth Profile Subsystem Product
- Bluetooth Component Product
- Bluetooth Development Tool
- Bluetooth Test Equipment

Bluetooth End Product - An implementation of Bluetooth wireless technology, which implements, at a minimum, all mandatory requirements in Radio, Baseband, Link Manager, Logical Link Control and Adaptation Protocol, Service Discovery Protocol and Generic Access Profile parts of the Specification.

Bluetooth Subsystem Product - An implementation of Bluetooth wireless technology, which implements only a portion of the Specification in compliance with such portion of the Specification and in accordance with the mandatory requirements as defined herein. Bluetooth Subsystem Products can be qualified solely for distribution and the use of Bluetooth wireless technology in Bluetooth Subsystem Products require such Bluetooth Subsystem Products to be combined with a complementary Bluetooth End Product or one or more complementary Bluetooth Subsystem Products such that the resulting combination satisfies the requirements of a Bluetooth End Product. There are three types of Bluetooth Subsystem Products as defined below:

- *Bluetooth Host Subsystem Product* – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in the Host Controller Interface, Logical Link Control and Adaptation Protocol, Service Discovery Protocol and Generic Access Profile parts of this Specification, but none of the protocols below Host Controller Interface (HCI). In addition, a Bluetooth Host Subsystem Product may contain, at a minimum, all the

mandatory requirements defined in one or more of the protocols and profiles above HCI.

- *Bluetooth Controller Subsystem Product* – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in the Bluetooth Radio, Baseband, Link Manager and HCI parts of this Specification, but none of the Protocols and Profiles above HCI.
- *Bluetooth Profile Subsystem Product* – A Bluetooth Subsystem Product containing, at a minimum, all the mandatory requirements defined in one or more of the profile specifications.

Bluetooth Component Product - An implementation of Bluetooth wireless technology, which does not meet the requirements of a Bluetooth End Product, but implements, at a minimum, all the mandatory requirements of either one or more of any of the protocol and profile parts of the Specification in compliance with such portion of the Specification. Bluetooth Component Products can be qualified solely for distribution and the use of the Bluetooth wireless technology in Bluetooth Component Products require such Bluetooth Component Products to be incorporated in Bluetooth End Products or Bluetooth Subsystem Products.

Bluetooth Development Tool - An implementation of Bluetooth wireless technology, intended to facilitate the development of new Bluetooth designs. Bluetooth Development Tools can be qualified solely for distribution and the use of the Bluetooth wireless technology in development of new Bluetooth Products.

Bluetooth Test Equipment - An implementation of Bluetooth wireless technology, intended to facilitate the testing of new Bluetooth Products. Bluetooth Test Equipment can be qualified solely for distribution and the use of the Bluetooth wireless technology in testing of new Bluetooth Products. Where necessary, Bluetooth Test Equipment may deviate from the Specification in order to fulfill the test purposes in the Bluetooth Test Specifications.

4 CORE CONFIGURATIONS

This section defines the set of features that are required for a product to be qualified to a specification name. Each core configuration is defined by a set of LMP feature bits or L2CAP feature bits that shall be supported to allow the configuration name to be used.

The configuration requirements imposed on a device depends on the profiles that it supports.

4.1 SPECIFICATION NAMING CONVENTIONS

Each specification is named by its core specification version number, followed by a list of the core configuration names that are implemented and qualified.

A complete specification name shall be stated as the core specification version number followed by “+”, and then either a single core configuration name or a sequence of core configuration names separated by “+”.

Examples of complete specification names including the core configuration names:

- Bluetooth v2.0
- Bluetooth v2.0 + EDR

In this example, a product claiming “Bluetooth v2.0” may implement some of the EDR features, following the requirements in other parts of the specifications, and be qualified for those features. If the full set required in Section 4.2 are not supported the “+EDR” configuration name shall not be used in product literature.

4.2 EDR CONFIGURATIONS

This section specifies additional compliance requirements that shall be followed if the configuration name “EDR” is used within the complete specification name. The configu-ration name “EDR” may only be used with the current core specification version number 2.0 or later versions of the specification.

Table 4.1 defines three categories of Transport Requirements that shall be satisfied subject to the following rules:

- A Bluetooth product shall support category 1 whenever it supports asynchronous transports for the profiles it incorporates.
- A Bluetooth product shall support category 2 whenever it supports asynchronous transports with multislot ACL packets for the profiles it incorporates.
- A Bluetooth product shall support category 3 whenever it supports eSCO syn-chronous transports for the profiles it incorporates.

.

No.	Transport Requirements	LMP Feature Bits Required	L2CAP Feature Bits Required
1	EDR for asynchronous transports (single slot)	Enhanced Data Rate ACL 2 Mbps mode (25) Enhanced Data Rate ACL 3 Mbps mode (26)	None
2	EDR for asynchronous transports (multi-slot)	3-slot Enhanced Data Rate ACL packets (39) 5-slot Enhanced Data Rate ACL packets (40)	None
3	EDR for synchronous transports	Enhanced Data Rate eSCO 2 Mbps mode (45) Enhanced Data Rate eSCO 3 Mbps mode (46)	None

Table 4.1: EDR configuration requirements

Note: No additional requirements are stated on the support of 2-EV5 and 3-EV5 packets.

APPENDIX

CONTENTS

1 REVISION HISTORY

Public versions are marked with bold in the tables below.

Beginning with the v1.2 of the Core System Package the core Bluetooth specification documents, protocols and profiles, are transferred to a new partitioning comprising 3 volumes and individual profile specifications are each contained in an individual document instead of the two volumes (Core and Profiles) previously used.

For more detailed information about changes between version published before v1.2, please see the Appendix I “Revision History” in v1.1.

1.1 [VOL 0] MASTER TOC & COMPLIANCE REQUIREMENTS

1.1.1 Bluetooth Compliance Requirements

Rev	Date	Comments
v2.0 + EDR	Oc t 15 2004	This version of the specification is intended to be a separate Bluetooth Specification that has all the functional characteristics of the v1.2 Bluetooth Specification that adds the Enhanced Data Rate (EDR) feature which required changes to Volume 0, Part A, Master Table of Contents.
v1.2	Nov 05 2003	This Part was moved from the Core volume. No content changes been made to this document since v1.1

1.2 [VOL 1] ARCHITECTURE & TERMINOLOGY OVERVIEW

Rev	Date	Comments
v2.0 + EDR	Oc t 15 2004	This version of the specification is intended to be a separate Bluetooth Specification that has all the functional characteristics of the v1.2 Bluetooth Specification that adds the Enhanced Data Rate (EDR) feature which incorporates changes to Volume 1, Part B, Acronyms and Abbreviations.
v1.2	Nov 05 2003	New volume with informational content. This volume will always be updated in parallel with the Core System volumes

1.3 [VOL 2 & 3] CORE SYSTEM PACKAGE

Rev	Date	Comments
v2.0 + EDR	Aug 01 2004	This version of the specification is intended to be a separate Bluetooth Specification. This specification was created by adding EDR and the errata for v1.2 ESR 1.
v1.2	Nov 05 2003	New features added in v1.2: - Architectural overview - Faster connection - Adaptive frequency hopping - Extended SCO links - Enhanced error detection and flow control - Enhanced synchronization capability - Enhanced flow specification The Core System Package now comprises two volumes and the text has gone through a radical change both in terms of structure and nomenclature. The language is also more precise and is adapted to meet the IEEE standard. The following parts are moved from the Core System Package to other volumes or has been deprecated: RFCOMM [vol 7], Object Exchange (IrDA Interoperability) [vol 8], TCS [vol 9], Interoperability Requirements for Bluetooth as a WAP Bearer [vol 6], HCI USB Transport Layer [vol4], HCI RS232 Transport Layer [vol 4], HCI UART Transport Layer [vol 4], Bluetooth Compliance Requirements [vol 0], Optional Paging Schemes [deprecated]
1.1	Feb 22nd 2001	The specification was updated with Errata items previously published on the web site. The Bluetooth Assigned Numbers appendix was lifted out from the specification to allow continuos maintenance on the web site.
1.0B	Dec. 1st 1999	The specification was updated with Errata items previously published on the web site and was revised from a linguistic point of view. The following parts was added: Interoperability Requirements for Bluetooth as a WAP Bearer, Test Control Interface, Sample Data (appendix), Bluetooth Audio (appendix), Baseband Timers (appendix) and Optional Paging Scheme (appendix)
1.0a	July 26th 1999	The first version of the Bluetooth Specification published on the public web site. Added part: Bluetooth Compliance Requirements.
1.0 draft	July 5th 1999	The following parts was added: Service Discovery Protocol (SDP), Telephony Control Specification (TCS), Bluetooth Assigned Numbers (appendix) and Message Sequence Charts (appendix)
0.9	April 30th 1999	The following parts was added: IrDA Interoperability, HCI RS232 Transport Layer, HCI UART Transport Layer and Test Mode
0.8	Jan 21st 1999	The following parts was added: Radio Specification, L2CAP, RFCOMM, HCI & HCI USB Transport Layer

Rev	Date	Comments
0.7	Oct 19th 1998	This first version only included Baseband and Link Manager Protocol

2 CONTRIBUTORS

2.1 [VOL 0] MASTER TOC & COMPLIANCE REQUIREMENTS

2.1.1 Part B: Bluetooth Compliance Requirements

BQRB (Editor)	
Wayne Park	3Com Corporation
Lawrence Jones	ComBit, Inc.
Gary Robinson	IBM Corporation
Georges Seuron	IBM Corporation
Rick Jessop	Intel Corporation
John Webb	Intel Corporation
Bruce Littlefield	Lucent Technologies, Inc.
Brian A. Redding	Motorola, Inc.
Waldemar Hontscha	Nokia Corporation
Petri Morko	Nokia Corporation
Magnus Hansson	Telefonaktiebolaget LM Ericsson
Magnus Sommansson	Telefonaktiebolaget LM Ericsson
Göran Svennarp	Telefonaktiebolaget LM Ericsson
Warren Allen	Toshiba Corporation
John Shi	Toshiba Corporation

2.2 [VOL 1] ARCHITECTURE &TERMINOLOGY OVERVIEW

2.2.1 Part A: Architectural Overview

Jennifer Bray	CSR
Robin Heydon	CSR
Henrik Hedlund	Ericsson
Martin van der Zee	Ericsson
Andy Glass	Microsoft
Brian Redding	Motorola
Jürgen Schnitzler	Nokia
Thomas Müller	Nokia
Joel Linsky	Silicon Wave
Terry Bourk	Silicon Wave
Michael Hasling	Tality
Yoshimitsu Shimojo	Toshiba
Toshiki Kizu	Toshiba
John Mersh	TTPCom

2.2.2 Part B: Acronyms & Abbreviations

Dan Sönnerstam	Pyramid Communication AB
Steve Koester	Bluetooth SIG, Inc.

2.2.3 Part C: Changes from Bluetooth Specification v1.1

Tom Siep	Bluetooth SIG Inc.
Robin Heydon	CSR
Henrik Hedlund	Ericsson
Rob Davies	Philips
Joel Linsky	Silicon Wave
John Mersh	TTPCom

2.3 [VOL 2] CORE SYSTEM PACKAGE, CONTROLLER

2.3.1 Part A: Radio Specification

Version 2.0 + EDR

Steven Hall	RF Micro Devices
Robert Young	CSR
Robert Kokke	Ericsson
Harald Kafemann	Nokia
Jukka Reunamäki	Nokia
Morton Gade	Digianswer
Mike Fitton	Toshiba
Oren Eliezer	Texas Instruments
Stephane Laurent-Michel	Tality

Version 1.2

Tom Siep	Bluetooth SIG Inc.
Jennifer Bray	CSR
Robin Heydon	CSR
Morten Gade	Digianswer/Motorola
Henrik Hedlund	Ericsson
Stefan Agnani	Ericsson
Robert Kokke	Ericsson
Roland Hellfajer	Infineon
Thomas Müller	Nokia
Antonio Salloum	Philips
Joel Linsky	Silicon Wave
Steven Hall	Silicon Wave
Oren Eliezer	Texas Instruments
Mike Fitton	Toshiba

Previous versions

Steve Williams	3Com Corporation
Todor V. Cooklov	Aware
Poul Hove Kristensen	Digianswer A/S
Kurt B. Fischer	Hyper Corporation
Kevin D. Marquess	Hyper Corporation
Troy Beukema	IBM Corporation
Brian Gaucher	IBM Corporation
Jeff Schiffer	Intel Corporation
James P. Gilb	Mobilian
Rich L. Ditch	Motorola, Inc.
Paul Burgess	Nokia Corporation
Olaf Joeressen	Nokia Corporation
Thomas Müller	Nokia Corporation

Arto T. Palin	Nokia Corporation
Steven J. Shellhammer	Symbol
Sven Mattisson	Telefonaktiebolaget LM Ericsson
Lars Nord (Section Owner)	Telefonaktiebolaget LM Ericsson
Anders Svensson	Telefonaktiebolaget LM Ericsson
Mary A. DuVal	Texas Instruments
Allen Hotari	Toshiba Corporation

2.3.2 Part B: Baseband Specification

Version 2.0 + EDR

Steven Hall	RF Micro Devices
Robert Young	CSR
Robert Kokke	Ericsson
Harald Kafemann	Nokia
Joel Linsky	RF Micro Devices
Terry Bourk	RF Micro Devices
Arto Palin	Nokia

Version 1.2

P G Madhavan	Agere
Hongbing Gan	Bandspeed, Inc.
Tod Sizer	Bell Labs
Alexander Thoukydides	CSR
Jennifer Bray	CSR
Robin Heydon	CSR
Kim Schneider	Digianswer/Motorola
Knud Dyring-Olsen	Digianswer/Motorola
Niels Nielsen	Digianswer/Motorola
Henrik Andersen	Digianswer/Motorola
Christian Gehrmann	Ericsson
Henrik Hedlund	Ericsson
Jan Åberg	Ericsson
Martin van der Zee	Ericsson
Rakesh Taori	Ericsson
Jaap Haartsen	Ericsson
Stefan Zürbes	Ericsson
Roland Hellfajer	Infineon
YC Maa	Integrated Programmable Communications, Inc.
HungKun Chen	Integrated Programmable Communications, Inc.
Steve McGowan	Intel
Adrian Stephens	Mobilian Corporation
Jim Lansford	Mobilian Corporation
Eric Meihofer	Motorola
Arto Palin	Nokia

Carmen Kühl	Nokia
Hannu Laine	Nokia
Jürgen Schnitzler	Nokia
Päivi Ruuska	Nokia
Thomas Müller	Nokia
Antonio Salloum	Philips
Harmke de Groot	Philips
Marianne van de Casteelee	Philips
Rob Davies	Philips
Roland Matthijssen	Philips
Joel Linsky (section owner)	Silicon Wave
Terry Bourk	Silicon Wave
Gary Schneider	Symbol Technologies, Inc.
Stephen J. Shellhammer	Symbol Technologies, Inc.
Michael Hasling	Tality
Amihai Kidron	Texas Instruments
Dan Michael	Texas Instruments
Eli Dekel	Texas Instruments
Jie Liang	Texas Instruments
Oren Eliezer	Texas Instruments
Tally Shina	Texas Instruments
Yariv Raveh	Texas Instruments
Anuj Batra	Texas Instruments
Katsuhiro Kinoshita	Toshiba
Toshiki Kizu	Toshiba
Yoshimitsu Shimojo	Toshiba
Charles Sturman	TTPCom
John Mersh	TTPCom
Sam Turner	TTPCom
Christoph Scholtz	University of Bonn
Simon Baatz	University of Bonn

Previous versions

Kevin D. Marquess	Hyper Corporation
Chatschik Bisdikian	IBM Corporation
Kris Fleming	Intel Corporation
James P. Gilb	Mobilian
David E. Cypher	NIST
Nada Golmie	NIST
Olaf Joeressen	Nokia Corporation
Thomas Müller	Nokia Corporation
Charlie Mellone	Motorola, Inc.
Harmke de Groot	Philips
Terry Bourk	Silicon Wave
Steven J. Shellhammer	Symbol

Jaap Haartsen	Telefonaktiebolaget LM Ericsson
Henrik Hedlund (Section Owner)	Telefonaktiebolaget LM Ericsson
Tobias Melin	Telefonaktiebolaget LM Ericsson
Joakim Persson	Telefonaktiebolaget LM Ericsson
Mary A. DuVal	Texas Instruments
Onn Haran	Texas Instruments
Thomas M. Siep	Texas Instruments
Ayse Findikli	Zeevo, Inc.

Previous versions [Encryption Sample Data, appendix]

Thomas Müller	Nokia Corporation
Thomas Sander	Nokia Corporation
Joakim Persson (Section Owner)	Telefonaktiebolaget LM Ericsson

Previous versions [Bluetooth Audio, appendix]

Magnus Hansson	Telefonaktiebolaget LM Ericsson
Fisseha Mekuria	Telefonaktiebolaget LM Ericsson
Mats Omrin	Telefonaktiebolaget LM Ericsson
Joakim Persson (Section Owner)	Telefonaktiebolaget LM Ericsson

Previous versions [Baseband Timers, appendix]

David E. Cyper	NIST
Jaap Haartsen (Section Owner)	Telefonaktiebolaget LM Ericsson
Joakim Persson	Telefonaktiebolaget LM Ericsson
Ayse Findikli	Zeevo, Inc.

2.3.3 Part C: Link Manager Protocol

Version 2.0 + EDR

John Mersh	TTPCom Ltd.
Joel Linsky	RF Micro Devices
Harald Kafemann	Nokia
Simon Morris	CSR

Version 1.2

Jennifer Bray	CSR
Robin Heydon	CSR
Simon Morris	CSR
Alexander Thoukydides	CSR
Kim Schneider	Digianswer/Motorola
Knud Dyring-Olsen	Digianswer/Motorola
Henrik Andersen	Digianswer/Motorola
Jan Åberg	Ericsson
Martin van der Zee	Ericsson

Roland Hellfajer	Infineon
YC Maa	Integrated Programmable Communications, Inc.
Steve McGowan	Intel
Tod Sizer	Lucent Technologies
Adrian Stephens	Mobilian
Jürgen Schnitzler	Nokia
Thomas Müller	Nokia
Carmen Kuhl	Nokia
Arto Palin	Nokia
Thomas Müller	Nokia
Roland Matthijssen	Philips
Rob Davies	Philips
Harmke de Groot	Philips
Antonio Salloum	Philips
Joel Linsky	Silicon Wave
Terry Bourk	Silicon Wave
Yariv Raveh	Texas Instruments
Tally Shina	Texas Instruments
Amihai Kidron	Texas Instruments
Yoshimitsu Shimojo	Toshiba
Toshiki Kizu	Toshiba
John Mersh (section owner)	TTPCom
Sam Turner	TTPCom

Previous versions

Kim Schneider	Digianswer A/S
Toru Aihara	IBM Corporation
Chatschik Bisdikian	IBM Corporation
Kris Fleming	Intel Corporation
David E. Cypher	NIST
Thomas Busse	Nokia Corporation
Julien Corthial	Nokia Corporation
Olaf Joeressen	Nokia Corporation
Thomas Müller	Nokia Corporation
Dong Nguyen	Nokia Corporation
Harmke de Groot	Philips
Terry Bourk	Silicon Wave
Johannes Elg	Telefonaktiebolaget LM Ericsson
Jaap Haartsen	Telefonaktiebolaget LM Ericsson
Tobias Melin (Section Owner)	Telefonaktiebolaget LM Ericsson
Mary A. DuVal	Texas Instruments
Onn Haran	Texas Instruments
John Mersh	TTPCom

2.3.4 Part D: Error Codes

Version 1.2

Robin Heydon (section owner)	CSR
Roland Hellfajer	Infineon
Joel Linsky	Silicon Wave
John Mersh	TTPCom

This part was earlier included in the LMP and HCI functional Specifications.

2.3.5 Part E: Bluetooth Host Controller Interface Functional Specification

Version 2.0 + EDR

Neil Stewart	Tality UK, Ltd.
Joel Linsky	RF Micro Devices
Robin Heydon	CSR

Version 1.2

Robin Heydon (section owner	CSR
Jennifer Bray	CSR
Alexander Thoukydides	CSR
Knud Dyring-Olsen	Digianswer/Motorola
Henrik Andersen	Digianswer/Motorola
Jan Åberg	Ericsson
Martin van der Zee	Ericsson
Don Liechty	Extended Systems
Kevin Marquess	Hyper Corp
Roland Hellfajer	Infineon
YC Maa	Integrated Programmable Communications, Inc.
Steve McGowan	Intel
Tod Sizer	Lucent Technologies
Tsuyoshi Okada	Matsushita Electric Industrial Co. Ltd
Andy Glass	Microsoft
Adrian Stephens	Mobilian
Jürgen Schnitzler	Nokia
Thomas Müller	Nokia
Rene Tischer	Nokia
Rob Davies	Philips
Antonio Salloum	Philips
Joel Linsky	Silicon Wave
Terry Bourk	Silicon Wave
Len Ott	Socket Communications
Randy Erman	Taiyo Yuden
Yoshimitsu Shimojo	Toshiba
Toshiki Kizu	Toshiba

Katsuhiro Kinoshita	Toshiba
Sam Turner	TTPCom
John Mersh	TTPCom

Previous versions

Todor Cooklev	3Com Corporation
Toru Aihara	IBM Corporation
Chatschik Bisdikian	IBM Corporation
Nathan Lee	IBM Corporation
Akihiko Mizutani	IBM Corporation
Les Cline	Intel Corporation
Bailey Cross	Intel Corporation
Kris Fleming	Intel Corporation
Robert Hunter	Intel Corporation
Jon Inouye	Intel Corporation
Srikanth Kambhatla	Intel Corporation
Steve Lo	Intel Corporation
Vijay Suthar	Intel Corporation
Bruce P. Kraemer	Intersil
Greg Muchnik	Motorola, Inc.
David E. Cypher	NIST
Thomas Busse	Nokia Corporation
Julien Courthial	Nokia Corporation
Thomas Müller	Nokia Corporation
Dong Nguyen	Nokia Corporation
Jürgen Schnitzler	Nokia Corporation
Fujio Watanabe	Nokia Corporation
Christian Zechlin	Nokia Corporation
Johannes Elg	Telefonaktiebolaget LM Ericsson
Christian Johansson (Section Owner)	Telefonaktiebolaget LM Ericsson
Patrik Lundin	Telefonaktiebolaget LM Ericsson
Tobias Melin	Telefonaktiebolaget LM Ericsson
Mary A. DuVal	Texas Instruments
Thomas M. Siep	Texas Instruments
Masahiro Tada	Toshiba Corporation
John Mersh	TTPCom

2.3.6 Part F: Message Sequence Charts

Version 1.2

Tom Siep	Bluetooth SIG Inc.
Robin Heydon (section owner)	CSR
Simon Morris	CSR
Jan Åberg	Ericsson
Christian Gehrmann	Ericsson
Joel Linsky	Silicon Wave
John Mersh	TTPCom

Previous versions

Todor Cooklev	3Com Corporation
Toru Aihara	IBM Corporation
Chatschik Bisdikian	IBM Corporation
Nathan Lee	IBM Corporation
Kris Fleming	Intel Corporation
Greg Muchnik	Motorola, Inc.
David E. Cypher	NIST
Thomas Busse	Nokia Corporation
Dong Nguyen (Section Owner)	Nokia Corporation
Fujio Watanabe	Nokia Corporation
Christian Johansson	Telefonaktiebolaget LM Ericsson
Tobias Melin	Telefonaktiebolaget LM Ericsson
Mary A. DuVal	Texas Instruments

2.3.7 Part G: Sample Data

Version 1.2

Joel Linsky	Silicon Wave

Previous versions

Thomas Müller	Nokia Corporation
Thomas Sander	Nokia Corporation
Joakim Persson (Section Owner)	Telefonaktiebolaget LM Ericsson

2.3.8 Part H: Security Specification

Please see Part B (Section 2.3.2 on page 62). The Security specification was until recently a part of the Baseband specification.

2.4 [VOL 3] CORE SYSTEM PACKAGE, HOST

2.4.1 Part A: Logical Link Control and Adaptation Protocol Specification

Version 1.2

Tom Siep	Bluetooth SIG Inc.
Carsten Andersen (section owner	CSR
Jennifer Bray	CSR
Jan Åberg	Ericsson
Martin van der Zee	Ericsson
Sam Ravnborg	Ericsson
Stefan Agnani	Ericsson
Steve McGowan	Intel Corporation
Joby Lafky	Microsoft
Doron Holan	Microsoft
Andy Glass	Microsoft
Brian Redding	Motorola
Jürgen Schnitzler	Nokia
Thomas Müller	Nokia
Rob Davies	Philips
Terry Bourk	Silicon Wave
Michael Hasling	Tality

Previous versions

Jon Burgess	3Com Corporation
Paul Moran	3Com Corporation
Doug Kogan	Extended Systems
Kevin D. Marquess	Hyper Corporation
Toru Aihara	IBM Corporation
Chatschik Bisdikian	IBM Corporation
Kris Fleming	Intel Corporation
Uma Gadamsetty	Intel Corporation
Robert Hunter	Intel Corporation
Jon Inouye	Intel Corporation
Steve C. Lo	Intel Corporation
Chunrong Zhu	Intel Corporation
Sergey Solyanik	Microsoft Corporation
David E. Cypher	NIST
Nada Golmie	NIST
Thomas Busse	Nokia Corporation
Rauno Makinen	Nokia Corporation
Thomas Müller	Nokia Corporation
Petri Nykänen	Nokia Corporation
Peter Ollikainen	Nokia Corporation
Petri O. Nurminen	Nokia Corporation

Johannes Elg	Telefonaktiebolaget LM Ericsson
Jaap Haartsen	Telefonaktiebolaget LM Ericsson
Elco Nijboer	Telefonaktiebolaget LM Ericsson
Ingemar Nilsson	Telefonaktiebolaget LM Ericsson
Stefan Runesson	Telefonaktiebolaget LM Ericsson
Gerrit Slot	Telefonaktiebolaget LM Ericsson
Johan Sörensen	Telefonaktiebolaget LM Ericsson
Goran Svennarp	Telefonaktiebolaget LM Ericsson
Mary A. DuVal	Texas Instruments
Thomas M. Siep	Texas Instruments
Kinoshita Katsuhiro	Toshiba Corporation

2.4.2 Part B: Service Discovery Protocol (SDP)

Version 1.2

Previous versions

Ned Plasson	3Com Corporation
John Avery	Convergence
Jason Kronz	Convergence
Chatschik Bisdikian	IBM Corporation
Parviz Kermani	IBM Corporation
Brent Miller	IBM Corporation
Dick Osterman	IBM Corporation
Bob Pascoe	IBM Corporation
Jon Inouye	Intel Corporation
Srikanth Kambhatla	Intel Corporation
Jay Eaglstun	Motorola, Inc.
Dale Farnsworth (Section Owner)	Motorola, Inc.
Jean-Michel Rosso	Motorola, Inc.
Jan Grönholm	Nokia Corporation
Kati Rantala	Nokia Corporation
Thomas Müller	Nokia Corporation
Johannes Elg	Telefonaktiebolaget LM Ericsson
Kazuaki Iwamura	Toshiba Corporation

2.4.3 Part C Generic Access Profile

Version 1.2

Jennifer Bray	CSR
Alexander Thoukydides	CSR
Christian Gehrmann	Ericsson
Henrik Hedlund	Ericsson
Jan Åberg	Ericsson
Stefan Agnani	Ericsson
Thomas Müller	Nokia
Joel Linsky	Silicon Wave
Terry Bourk	Silicon Wave
Katsuhiro Kinoshita	Toshiba

Previous versions

Ken Morley	3Com Corporation
Chatschik Bisdikian	IBM Corporation
Jon Inouye	Intel Corporation
Brian Redding	Motorola, Inc.
David E. Cypher	NIST
Stephane Bouet	Nokia Corporation
Thomas Müller	Nokia Corporation
Martin Roter	Nokia Corporation
Johannes Elg	Telefonaktiebolaget LM Ericsson
Patric Lind (Section Owner)	Telefonaktiebolaget LM Ericsson
Erik Slotboom	Telefonaktiebolaget LM Ericsson
Johan Sörensen	Telefonaktiebolaget LM Ericsson

2.4.4 Part D: Test Support

Version 1.2

Emilio Mira Escartis	Cetecom
Robin Heydon	CSR
Jennifer Bray	CSR
Stefan Agnani (section owner)	Ericsson
Terry Bourk	Silicon Wave
Joel Linsky	Silicon Wave
Michael Hasling	Tality

Previous versions [Test Mode]

Jeffrey Schiffer	Intel Corporation
David E. Cypher	NIST
Daniel Bencak	Nokia Corporation
Arno Kefenbaum	Nokia Corporation
Thomas Müller (Section Owner)	Nokia Corporation

Roland Schmale	Nokia Corporation
Fujio Watanabe	Nokia Corporation
Stefan Agnani	Telefonaktiebolaget LM Ericsson
Mårten Mattsson	Telefonaktiebolaget LM Ericsson
Tobias Melin	Telefonaktiebolaget LM Ericsson
Lars Nord	Telefonaktiebolaget LM Ericsson
Fredrik Töörn	Telefonaktiebolaget LM Ericsson
John Mersh	TTPCom
Ayse Findikli	Zeevo, Inc.

Previous versions [Test Control Interface]

Mike Feldman	Motorola, Inc.
Thomas Müller	Nokia Corporation
Stefan Agnani (Section Owner)	Telefonaktiebolaget LM Ericsson
Mårten Mattsson	Telefonaktiebolaget LM Ericsson
Dan Sönnerstam	Telefonaktiebolaget LM Ericsson

Specification of the Bluetooth System

Wireless connections made easy

Architecture & Terminology Overview

Covered Core Package version:
2.0 + EDR
Current Master TOC issued:
4 November 2004

Bluetooth®

Revision History

The Revision History is shown in the “Appendix” on page 51[vol. 0].

Contributors

The persons who contributed to this specification are listed in the Appendix.

Web Site

This specification can also be found on the official Bluetooth web site:
http://www.bluetooth.com

Disclaimer and Copyright Notice

Part A
ARCHITECTURE

Contents

Architecture

CONTENTS

1 GENERAL DESCRIPTION

Bluetooth wireless technology is a short-range communications system intended to replace the cable(s) connecting portable and/or fixed electronic devices. The key features of Bluetooth wireless technology are robustness, low power, and low cost. Many features of the core specification are optional, allowing product differentiation.

The Bluetooth core system consists of an RF transceiver, baseband, and protocol stack. The system offers services that enable the connection of devices and the exchange of a variety of classes of data between these devices.

This chapter of the specification provides an overview of the Bluetooth system architecture, communication topologies and data transport features. The text in this chapter of the specification should be treated as informational and used as a background and for context-setting.

1.1 OVERVIEW OF OPERATION

The Bluetooth RF (physical layer) operates in the unlicensed ISM band at 2.4 GHz. The system employs a frequency hop transceiver to combat interference and fading and provides many FHSS carriers. RF operation uses a shaped, binary frequency modulation to minimize transceiver complexity. The symbol rate is 1 Megasymbol per second (Ms/s) supporting the bit rate of 1 Megabit per second (Mb/s) or, with Enhanced Data Rate, a gross air bit rate of 2 or 3Mb/s. These modes are known as Basic Rate and Enhanced Data Rate respectively.

During typical operation a physical radio channel is shared by a group of devices that are synchronized to a common clock and frequency hopping pattern. One device provides the synchronization reference and is known as the master. All other devices are known as slaves. A group of devices synchronized in this fashion form a piconet. This is the fundamental form of communication in the Bluetooth wireless technology.

Devices in a piconet use a specific frequency hopping pattern, which is algorithmically determined by certain fields in the Bluetooth address and clock of the master. The basic hopping pattern is a pseudo-random ordering of the 79 frequencies in the ISM band. The hopping pattern may be adapted to exclude a portion of the frequencies that are used by interfering devices. The adaptive hopping technique improves Bluetooth co-existence with static (non-hopping) ISM systems when these are co-located.

The physical channel is sub-divided into time units known as slots. Data is transmitted between Bluetooth devices in packets, that are positioned in these slots. When circumstances permit, a number of consecutive slots may be allocated to a single packet. Frequency hopping takes place between the transmis-

sion or reception of packets. Bluetooth technology provides the effect of full duplex transmission through the use of a Time-Division Duplex (TDD) scheme.

Above the physical channel there is a layering of links and channels and associated control protocols. The hierarchy of channels and links from the physical channel upwards is physical channel, physical link, logical transport, logical link and L2CAP channel. These are discussed in more detail in Section 3.3 on page 34 - Section 3.6 on page 50 but are introduced here to aid the understanding of the remainder of this section.

Within a physical channel, a physical link is formed between any two devices that transmit packets in either direction between them. In a piconet physical channel there are restrictions on which devices may form a physical link. There is a physical link between each slave and the master. Physical links are not formed directly between the slaves in a piconet.

The physical link is used as a transport for one or more logical links that support unicast synchronous, asynchronous and isochronous traffic, and broadcast traffic. Traffic on logical links is multiplexed onto the physical link by occupying slots assigned by a scheduling function in the resource manager.

A control protocol for the baseband and physical layers is carried over logical links in addition to user data. This is the link manager protocol (LMP). Devices that are active in a piconet have a default asynchronous connection-oriented logical transport that is used to transport the LMP protocol signalling. For historical reasons this is known as the ACL logical transport. The default ACL logical transport is the one that is created whenever a device joins a piconet. Additional logical transports may be created to transport synchronous data streams when this is required.

The Link Manager function uses LMP to control the operation of devices in the piconet and provide services to manage the lower architectural layers (radio layer and baseband layer). The LMP protocol is only carried on the default ACL logical transport and the default broadcast logical transport.

Above the baseband layer the L2CAP layer provides a channel-based abstraction to applications and services. It carries out segmentation and reassembly of application data and multiplexing and de-multiplexing of multiple channels over a shared logical link. L2CAP has a protocol control channel that is carried over the default ACL logical transport. Application data submitted to the L2CAP protocol may be carried on any logical link that supports the L2CAP protocol.

1.2 NOMENCLATURE

Where the following terms appear in the specification they have the meaning given in Table 1.1 on page 15.

Ad Hoc Network	A network typically created in a spontaneous manner. An ad hoc network requires no formal infrastructure and is limited in temporal and spatial extent.
Active Slave Broadcast (ASB)	The Active Slave Broadcast logical transport that is used to transport L2CAP user traffic to all active devices in the piconet. See Section 3.5.7 on page 46
Beacon Train	A pattern of reserved slots within a basic or adapted piconet physical channel. Transmissions starting in these slots are used to resynchronize parked devices.
Bluetooth	Bluetooth is a wireless communication link, operating in the unlicensed ISM band at 2.4 GHz using a frequency hopping transceiver. It allows real-time AV and data communications between Bluetooth Hosts. The link protocol is based on time slots.
Bluetooth Baseband	The part of the Bluetooth system that specifies or implements the medium access and physical layer procedures to support the exchange of real-time voice, data information streams, and ad hoc networking between Bluetooth Devices.
Bluetooth Clock	A 28 bit clock internal to a Bluetooth controller sub-system that ticks every 312.5µs. The value of this clock defines the slot numbering and timing in the various physical channels.
Bluetooth Controller	A sub-system containing the Bluetooth RF, baseband, resource controller, link manager, device manager and a Bluetooth HCI.
Bluetooth Device	A Bluetooth Device is a device that is capable of short-range wireless communications using the Bluetooth system.
Bluetooth Device Address	A 48 bit address used to identify each Bluetooth device.
BD_ADDR	The Bluetooth Device Address, BD_ADDR, is used to identify a Bluetooth device.
Bluetooth HCI	The Bluetooth Host Controller Interface (HCI) provides a command interface to the baseband controller and link manager and access to hardware status and control registers. This interface provides a uniform method of accessing the Bluetooth baseband capabilities.

Table 1.1: Nomenclature.

Bluetooth Host	A Bluetooth Host is a computing device, peripheral, cellular telephone, access point to PSTN network or LAN, etc. A Bluetooth Host attached to a Bluetooth Controller may communicate with other Bluetooth Hosts attached to their Bluetooth Controllers as well.
Channel	Either a physical channel or an L2CAP channel, depending on the context.
Connect (to service)	The establishment of a connection to a service. If not already done, this also includes establishment of a physical link, logical transport, logical link and L2CAP channel.
Connectable device	A Bluetooth device in range that periodically listens on its page scan physical channel and will respond to a page on that channel.
Connected devices	Two Bluetooth devices in the same piconet and with a physical link between them.
Connecting	A phase in the communication between devices when a connection between them is being established. (Connecting phase follows after the link establishment phase is completed.)
Connection	A connection between two peer applications or higher layer protocols mapped onto an L2CAP channel.
Connection establishment	A procedure for creating a connection mapped onto a channel.
coverage area	The area where two Bluetooth devices can exchange messages with acceptable quality and performance.
Creation of a secure connection	A procedure of establishing a connection, including authentication and encryption.
Creation of a trusted relationship	A procedure where the remote device is marked as a trusted device. This includes storing a common link key for future authentication and pairing (if the link key is not available).
Device discovery	A procedure for retrieving the Bluetooth device address, clock, class-of-device field and used page scan mode from discoverable devices.
Discoverable device	A Bluetooth device in range that periodically listens on an inquiry scan physical channel and will respond to an inquiry on that channel. Discoverable device are normally also connectable.
Inquiring device	A Bluetooth device that is carrying out the inquiry procedure.
Inquiry	A procedure where a Bluetooth device transmits inquiry messages and listens for responses in order to discover the other Bluetooth devices that are within the coverage area.

Table 1.1: Nomenclature.

Inquiry scan	A procedure where a Bluetooth device listens for inquiry messages received on its inquiry scan physical channel.
Interoperability	The ability of two or more systems or components to exchange information and to use the information that has been exchanged.
Isochronous data	Information in a stream where each information entity in the stream is bound by a time relationship to previous and successive entities.
Known device	A Bluetooth device for which at least the BD_ADDR is stored.
L2CAP Channel	A logical connection on L2CAP level between two devices serving a single application or higher layer protocol.
L2CAP Channel establishment	A procedure for establishing a logical connection on L2CAP level.
Link establishment	A procedure for establishing the default ACL link and hierarchy of links and channels between devices.
Link	Shorthand for a logical link.
Link key	A secret key that is known by two devices and is used in order to authenticate each device to the other
LMP authentication	An LMP level procedure for verifying the identity of a remote device.
LMP pairing	A procedure that authenticates two devices and creates a common link key that can be used as a basis for a trusted relationship or a (single) secure connection.
Logical Channel	Identical to an L2CAP channel, but deprecated due to an alternative meaning in Bluetooth 1.1
Logical link	The lowest architectural level used to offer independent data transport services to clients of the Bluetooth system.
Logical transport	Used in Bluetooth to represent commonality between different logical links due to shared acknowledgement protocol and link identifiers.
Name discovery	A procedure for retrieving the user-friendly name (the Bluetooth device name) of a connectable device.
Packet	Format of aggregated bits that are transmitted on a physical channel.
Page	The initial phase of the connection procedure where a device transmits a train of page messages until a response is received from the target device or a time-out occurs.
Page scan	A procedure where a device listens for page messages received on its page scan physical channel.

Table 1.1: Nomenclature.

Paging device	A Bluetooth device that is carrying out the page procedure.
Paired device	A Bluetooth device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase).
Parked device	A device operating in a basic mode piconet that is synchronized to the master but has given up its default ACL logical transport.
Physical Channel	Characterized by synchronized occupancy of a sequence of RF carriers by one or more devices. A number of physical channel types exist with characteristics defined for their different purposes.
Physical Link	A Baseband-level connection between two devices established using paging.
Piconet	A collection of devices occupying a shared physical channel where one of the devices is the Piconet Master and the remaining devices are connected to it.
Piconet Physical Channel	A Channel that is divided into time slots in which each slot is related to an RF hop frequency. Consecutive hops normally correspond to different RF hop frequencies and occur at a standard hop rate of 1600 hops/s. These consecutive hops follow a pseudo-random hopping sequence, hopping through a 79 RF channel set.
Piconet Master	The device in a piconet whose Bluetooth Clock and Bluetooth Device Address are used to define the piconet physical channel characteristics.
Piconet Slave	Any device in a piconet that is not the Piconet Master, but is connected to the Piconet Master.
PIN	A user-friendly number that can be used to authenticate connections to a device before paring has taken place.
PMP	A Participant in Multiple Piconets. A device that is concurrently a member of more than one piconet, which it achieves using time division multiplexing (TDM) to interleave its activity on each piconet physical channel.
The Parked Slave Broadcast (PSB)	The Parked Slave Broadcast logical transport that is used for communications between the master and parked devices. Section 3.5.8 on page 47.
Scatternet	Two or more piconets that include one or more devices acting as PMPs.
Service Layer Protocol	A protocol that uses an L2CAP channel for transporting PDUs.
Service discovery	Procedures for querying and browsing for services offered by or through another Bluetooth device.

Table 1.1: Nomenclature.

Silent device	A Bluetooth device appears as silent to a remote device if it does not respond to inquiries made by the remote device.
Unknown device	A Bluetooth device for which no information (Bluetooth Device Address, link key or other) is stored.

Table 1.1: Nomenclature.

2 CORE SYSTEM ARCHITECTURE

The Bluetooth core system covers the four lowest layers and associated protocols defined by the Bluetooth specification as well as one common service layer protocol, the Service Discovery Protocol (SDP) and the overall profile requirements are specified in the Generic Access Profile (GAP). A complete Bluetooth application requires a number of additional service and higher layer protocols that are defined in the Bluetooth specification, but are not described here. The core system architecture is shown in Figure 2.1 on page 21 except for SDP that is not shown for clarity.

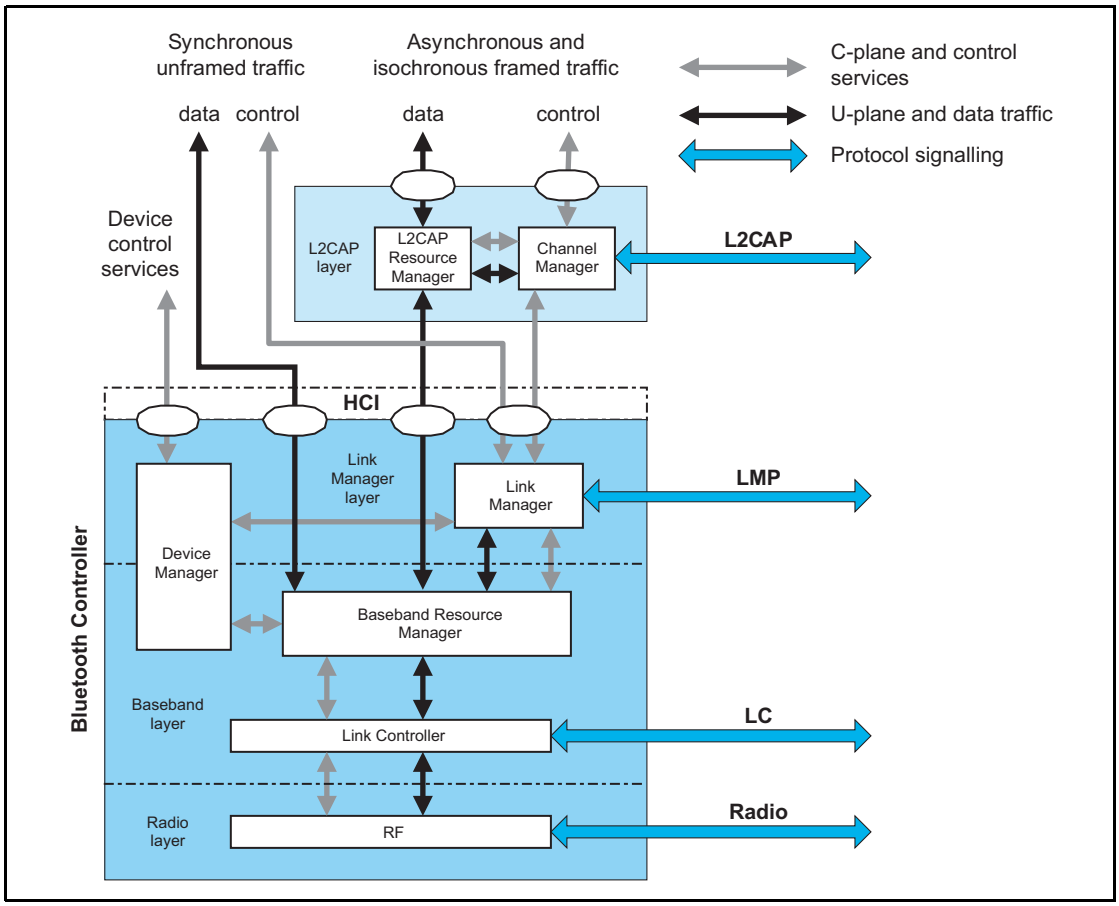


Figure 2.1: Bluetooth core system architecture

Figure 2.1 on page 21 shows the four lowest layers, each with its associated communication protocol. The lowest three layers are sometimes grouped into a subsystem known as the Bluetooth controller. This is a common implementation involving a standard physical communications interface between the Bluetooth controller and remainder of the Bluetooth system including the L2CAP, service and higher layers (known as the Bluetooth host.) Although this interface is optional the architecture is designed to allow for its existence and characteristics. The Bluetooth specification enables inter-operability between independent Bluetooth systems by defining the protocol messages exchanged between equivalent layers, and also inter-operability between independent

Bluetooth sub-systems by defining a common interface between Bluetooth controllers and Bluetooth hosts.

A number of functional blocks are shown and the path of services and data between these. The functional blocks shown in the diagram are informative; in general the Bluetooth specification does not define the details of implementations except where this is required for inter-operability. Thus the functional blocks in Figure 2.1 on page 21 are shown in order to aid description of the system behavior. An implementation may be different from the system shown in Figure 2.1 on page 21.

Standard interactions are defined for all inter-device operation, where Bluetooth devices exchange protocol signalling according to the Bluetooth specification. The Bluetooth core system protocols are the Radio (RF) protocol, Link Control (LC) protocol, Link Manager (LM) protocol and Logical Link Control and Adaptation protocol (L2CAP), all of that are fully defined in subsequent parts of the Bluetooth specification. In addition the Service Discovery Protocol (SDP) is a service layer protocol required by all Bluetooth applications.

The Bluetooth core system offers services through a number of service access points that are shown in the diagram as ellipses. These services consist of the basic primitives that control the Bluetooth core system. The services can be split into three types. There are device control services that modify the behavior and modes of a Bluetooth device, transport control services that create, modify and release traffic bearers (channels and links), and data services that are used to submit data for transmission over traffic bearers. It is common to consider the first two as belonging to the C-plane and the last as belonging to the U-plane.

A service interface to the Bluetooth controller sub-system is defined such that the Bluetooth controller may be considered a standard part. In this configuration the Bluetooth controller operates the lowest three layers and the L2CAP layer is contained with the rest of the Bluetooth application in a host system. The standard interface is called the Host to Controller Interface (HCI) and its service access points are represented by the ellipses on the upper edge of the Bluetooth controller sub-system in Figure 2.1 on page 21. Implementation of this standard service interface is optional.

As the Bluetooth architecture is defined with the possibility of separate host and controller communicating through an HCI, a number of general assumptions are made. The Bluetooth controller is assumed to have limited data buffering capabilities in comparison with the host. Therefore the L2CAP layer is expected to carry out some simple resource management when submitting L2CAP PDUs to the controller for transport to a peer device. This includes segmentation of L2CAP SDUs into more manageable PDUs and then the fragmentation of PDUs into start and continuation packets of a size suitable for the controller buffers, and management of the use of controller buffers to ensure availability for channels with Quality of Service (QoS) commitments.

The Baseband layer provides the basic ARQ protocol in Bluetooth. The L2CAP layer can optionally provide a further error detection and retransmission to the L2CAP PDUs. This feature is recommended for applications with requirements for a low probability of undetected errors in the user data. A further optional feature of L2CAP is a window-based flow control that can be used to manage buffer allocation in the receiving device. Both of these optional features augment the QoS performance in certain scenarios.

Although these assumptions may not be required for embedded Bluetooth implementations that combine all layers in a single system, the general architectural and QoS models are defined with these assumptions in mind, in effect a lowest common denominator.

Automated conformance testing of implementations of the Bluetooth core system is required. This is achieved by allowing the tester to control the implementation through the RF interface, which is common to all Bluetooth systems, and through the Test Control Interface (TCI), which is only required for conformance testing.

The tester uses exchanges with the Implementation Under Test (IUT) through the RF interface to ensure the correct responses to requests from remote devices. The tester controls the IUT through the TCI to cause the IUT to originate exchanges through the RF interface so that these can also be verified as conformant.

The TCI uses a different command-set (service interface) for the testing of each architectural layer and protocol. A subset of the HCI command-set is used as the TCI service interface for each of the layers and protocols within the Bluetooth Controller subsystem. A separate service interface is used for testing the L2CAP layer and protocol. As an L2CAP service interface is not defined in the Bluetooth core specification it is defined separately in the Test Control Interface specification. Implementation of the L2CAP service interface is only required for conformance testing.

2.1 CORE ARCHITECTURAL BLOCKS

This section describes the function and responsibility of each of the blocks shown in Figure 2.1 on page 21. An implementation is not required to follow the architecture described above, though every implementation shall conform to the protocol specifications described in subsequent parts of the Bluetooth specification, and shall implement the behavioral aspects of the system outlined below and specified in subsequent parts of the Bluetooth specification.

2.1.1 Channel manager

The channel manager is responsible for creating, managing and destroying L2CAP channels for the transport of service protocols and application data streams. The channel manager uses the L2CAP protocol to interact with a channel manager on a remote (peer) device to create these L2CAP channels and connect their endpoints to the appropriate entities. The channel manager interacts with its local link manager to create new logical links (if necessary) and to configure these links to provide the required quality of service for the type of data being transported.

2.1.2 L2CAP resource manager

The L2CAP resource manager block is responsible for managing the ordering of submission of PDU fragments to the baseband and some relative scheduling between channels to ensure that L2CAP channels with QoS commitments are not denied access to the physical channel due to Bluetooth controller resource exhaustion. This is required because the architectural model does not assume that the Bluetooth controller has limitless buffering, or that the HCI is a pipe of infinite bandwidth.

L2CAP Resource Managers may also carry out traffic conformance policing to ensure that applications are submitting L2CAP SDUs within the bounds of their negotiated QoS settings. The general Bluetooth data transport model assumes well-behaved applications, and does not define how an implementation is expected to deal with this problem.

2.1.3 Device manager

The device manager is the functional block in the baseband that controls the general behavior of the Bluetooth device. It is responsible for all operation of the Bluetooth system that is not directly related to data transport, such as inquiring for the presence of other nearby Bluetooth devices, connecting to other Bluetooth devices, or making the local Bluetooth device discoverable or connectable by other devices.

The device manager requests access to the transport medium from the baseband resource controller in order to carry out its functions.

The device manager also controls local device behavior implied by a number of the HCI commands, such as managing the device local name, any stored link keys, and other functionality.

2.1.4 Link manager

The link manager is responsible for the creation, modification and release of logical links (and, if required, their associated logical transports), as well as the update of parameters related to physical links between devices. The link manager achieves this by communicating with the link manager in remote Bluetooth devices using the Link Management Protocol (LMP.)

The LM protocol allows the creation of new logical links and logical transports between devices when required, as well as the general control of link and transport attributes such as the enabling of encryption on the logical transport, the adapting of transmit power on the physical link, or the adjustment of QoS settings for a logical link.

2.1.5 Baseband resource manager

The baseband resource manager is responsible for all access to the radio medium. It has two main functions. At its heart is a scheduler that grants time on the physical channels to all of the entities that have negotiated an access contract. The other main function is to negotiate access contracts with these entities. An access contract is effectively a commitment to deliver a certain QoS that is required in order to provide a user application with an expected performance.

The access contract and scheduling function must take account of any behavior that requires use of the Bluetooth radio. This includes (for example) the normal exchange of data between connected devices over logical links, and logical transports, as well as the use of the radio medium to carry out inquiries, make connections, be discoverable or connectable, or to take readings from unused carriers during the use of adaptive frequency hopping mode.

In some cases the scheduling of a logical link results in changing to a different physical channel from the one that was previously used. This may be (for example) due to involvement in scatternet, a periodic inquiry function, or page

scanning. When the physical channels are not time slot aligned, then the resource manager also accounts for the realignment time between slots on the original physical channel and slots on the new physical channel. In some cases the slots will be naturally aligned due to the same device clock being used as a reference for both physical channels.

2.1.6 Link controller

The link controller is responsible for the encoding and decoding of Bluetooth packets from the data payload and parameters related to the physical channel, logical transport and logical link.

The link controller carries out the link control protocol signalling (in close conjunction with the scheduling function of the resource manager), which is used to communicate flow control and acknowledgement and retransmission request signals. The interpretation of these signals is a characteristic of the logical transport associated with the baseband packet. Interpretation and control of the link control signalling is normally associated with the resource manager's scheduler.

2.1.7 RF

The RF block is responsible for transmitting and receiving packets of information on the physical channel. A control path between the baseband and the RF block allows the baseband block to control the timing and frequency carrier of the RF block. The RF block transforms a stream of data to and from the physical channel and the baseband into required formats.

3 DATA TRANSPORT ARCHITECTURE

The Bluetooth data transport system follows a layered architecture. This description of the Bluetooth system describes the Bluetooth core transport layers up to and including L2CAP channels. All Bluetooth operational modes follow the same generic transport architecture, which is shown in Figure 3.1 on page 27.

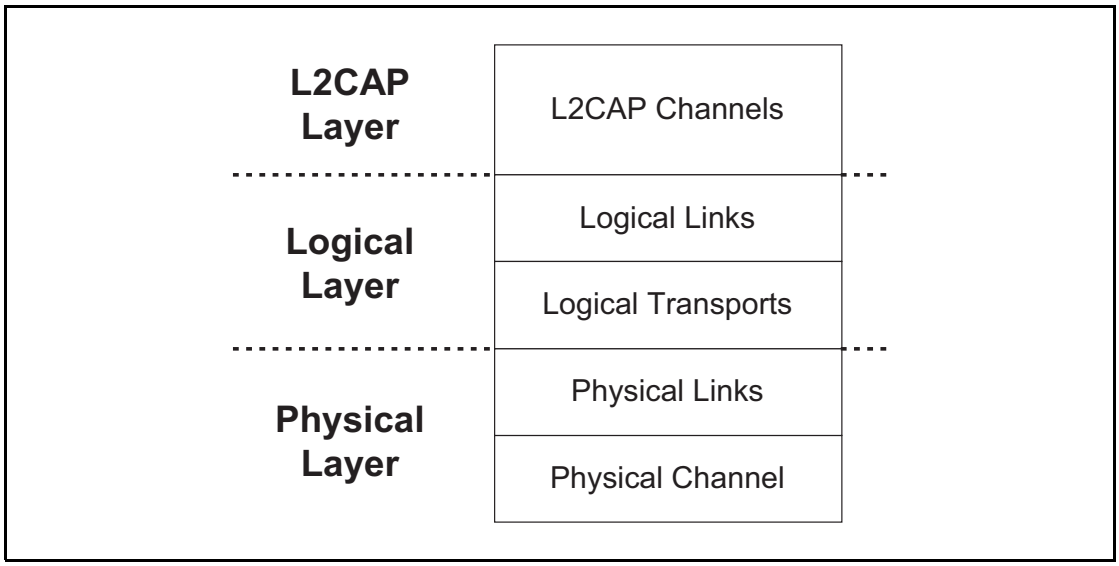


Figure 3.1: Bluetooth generic data transport architecture

For efficiency and legacy reasons, the Bluetooth transport architecture includes a sub-division of the logical layer, distinguishing between logical links and logical transports. This sub-division provides a general (and commonly understood) concept of a logical link that provides an independent transport between two or more devices. The logical transport sub-layer is required to describe the inter-dependence between some of the logical link types (mainly for reasons of legacy behavior.)

The Bluetooth 1.1 specification described the ACL and SCO links as physical links. With the addition of Extended SCO (eSCO) and for future expansion it is better to consider these as logical transport types, which more accurately encapsulates their purpose. However, they are not as independent as might be desired, due to their shared use of resources such as the LT_ADDR and acknowledgement/repeat request (ARQ) scheme. Hence the architecture is incapable of representing these logical transports with a single transport layer. The additional logical transport layer goes some way towards describing this behavior.

3.1 CORE TRAFFIC BEARERS

The Bluetooth core system provides a number of standard traffic bearers for the transport of service protocol and application data. These are shown in Figure 3.2 on page 28 below (for ease of representation this is shown with higher layers to the left and lower layers to the right).

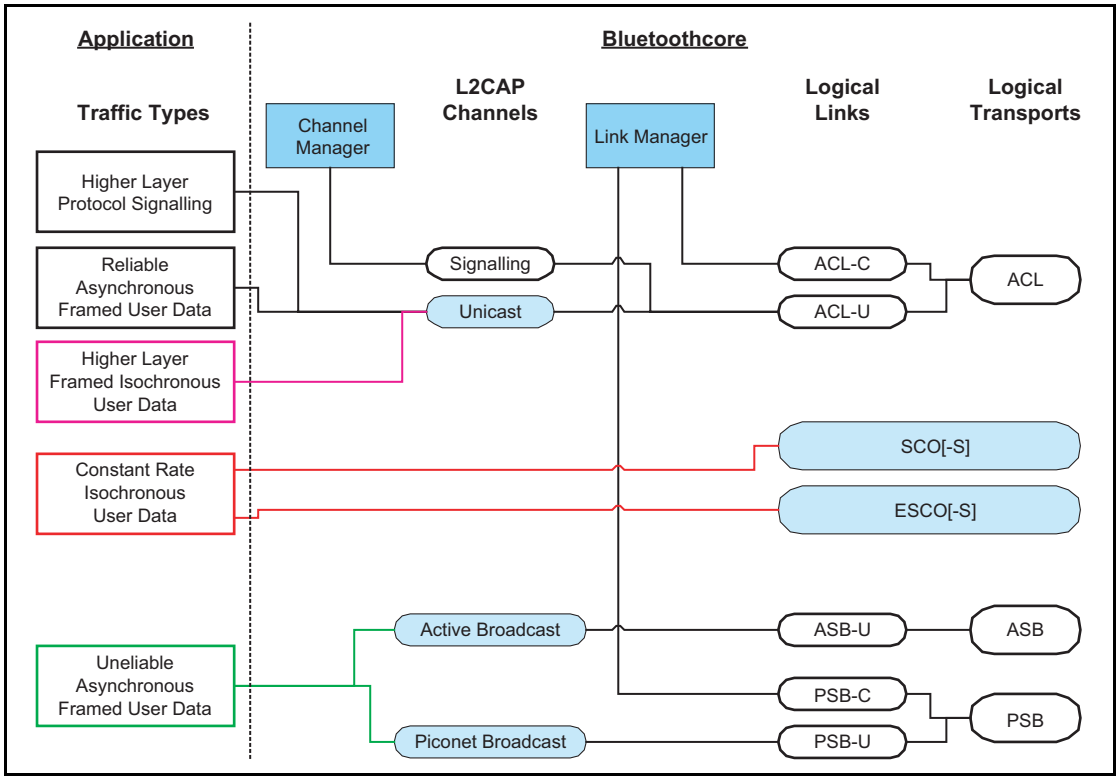


Figure 3.2: Bluetooth traffic bearers

The core traffic bearers that are available to applications are shown in Figure 3.2 on page 28 as the shaded rounded rectangles. The architectural layers that are defined to provide these services are described in Section 2 on page 21. A number of data traffic types are shown on the left of the diagram linked to the traffic bearers that are typically suitable for transporting that type of data traffic.

The logical links are named using the names of the associated logical transport and a suffix that indicates the type of data that is transported. (C for control links carrying LMP messages, U for L2CAP links carrying user data (L2CAP PDUs) and S for stream links carrying unformatted synchronous or isochronous data.) It is common for the suffix to be removed from the logical link without introducing ambiguity, thus a reference to the default ACL logical transport can be resolved to mean the ACL-C logical link in cases where the LMP protocol is being discussed, or the ACL-U logical link when the L2CAP layer is being discussed.

The mapping of application traffic types to Bluetooth core traffic bearers in Figure 3.2 on page 28 is based on matching the traffic characteristics with the

bearer characteristics. It is recommended to use these mappings as they provide the most natural and efficient method of transporting the data with its given characteristics.

However, an application (or an implementation of the Bluetooth core system) may choose to use a different traffic bearer, or a different mapping to achieve a similar result. For example, in a piconet with only one slave, the master may choose to transport L2CAP broadcasts over the ACL-U logical link rather than over the ASB-U or PSB-U logical links. This will probably be more efficient in terms of bandwidth (if the physical channel quality is not too degraded.) Use of alternative transport paths to those in Figure 3.2 on page 28 is only acceptable if the characteristics of the application traffic type are preserved.

Figure 3.2 on page 28 shows a number of application traffic types. These are used to classify the types of data that may be submitted to the Bluetooth core system. The original data traffic type may not be the same as the type that is submitted to the Bluetooth core system if an intervening process modifies it. For example, video data is generated at a constant rate but an intermediate coding process may alter this to variable rate, e.g. by MPEG4 encoding. For the purposes of the Bluetooth core system, only the characteristic of the submitted data is of interest.

3.1.1 Framed data traffic

The L2CAP layer services provide a frame-oriented transport for asynchronous and isochronous user data. The application submits data to this service in variable-sized frames (up to a negotiated maximum for the channel) and these frames are delivered in the same form to the corresponding application on the remote device. There is no requirement for the application to insert additional framing information into the data, although it may do so if this is required (such framing is invisible to the Bluetooth core system.)

Connection-oriented L2CAP channels may be created for transport of unicast (point-to-point) data between two Bluetooth devices. A connectionless L2CAP channel exists for broadcasting data. In the case of piconet topologies the master device is always the source of broadcast data and the slave device(s) are the recipients. Traffic on the broadcast L2CAP channel is uni-directional. Unicast L2CAP channels may be uni-directional or bi-directional.

L2CAP channels have an associated QoS setting that defines constraints on the delivery of the frames of data. These QoS settings may be used to indicate (for example) that the data is isochronous, and therefore has a limited lifetime after which it becomes invalid, or that the data should be delivered within a given time period, or that the data is reliable and should be delivered without error, however long this takes.

The L2CAP channel manager is responsible for arranging to transport the L2CAP channel data frames on an appropriate baseband logical link, possibly multiplexing this onto the baseband logical link with other L2CAP channels with similar characteristics.

3.1.2 Unframed data traffic

If the application does not require delivery of data in frames, possibly because it includes in-stream framing, or because the data is a pure stream, then it may avoid the use of L2CAP channels and make direct use of a baseband logical link.

The Bluetooth core system supports the direct transport of application data that is isochronous and of a constant rate (either bit-rate, or frame-rate for pre-framed data), using a SCO-S or eSCO-S logical link. These logical links reserve physical channel bandwidth and provide a constant rate transport locked to the piconet clock. Data is transported in fixed size packets at fixed intervals with both of these parameters negotiated during channel establishment. eSCO links provide a greater choice of bit-rates and also provide greater reliability by using limited retransmission in case of error. Enhanced Data Rate operation is supported for eSCO, but not for SCO logical transports. SCO and eSCO logical transports do not support multiplexed logical links or any further layering within the Bluetooth core. An application may choose to layer a number of streams within the submitted SCO/eSCO stream, provided that the submitted stream is, or has the appearance of being, a constant rate stream.

The application chooses the most appropriate type of logical link from those available at the baseband, and creates and configures it to transport the data stream, and releases it when completed. (The application will normally also use a framed L2CAP unicast channel to transport its C-plane information to the peer application on the remote device.)

If the application data is isochronous and of a variable rate, then this may only be carried by the L2CAP unicast channel, and hence will be treated as framed data.

3.1.3 Reliability of traffic bearers

Bluetooth is a wireless communications system. In poor RF environments, this system should be considered inherently unreliable. To counteract this the system provides levels of protection at each layer. The baseband packet header uses forward error correcting (FEC) coding to allow error correction by the receiver and a header error check (HEC) to detect errors remaining after correction. Certain Baseband packet types include FEC for the payload. Furthermore, some Baseband packet types include a cyclic redundancy error check (CRC).

On ACL logical transports the results of the error detection algorithm are used to drive a simple acknowledgement/repeat request (ARQ) protocol. This provides an enhanced reliability by re-transmitting packets that do not pass the receiver’s error checking algorithm. It is possible to modify this scheme to support latency-sensitive packets by discarding an unsuccessfully transmitted packet at the transmitter if the packet’s useful life has expired. eSCO links use

a modified version of this scheme to improve reliability by allowing a limited number of retransmissions.

The resulting reliability gained by this ARQ scheme is only as dependable as the ability of the HEC and CRC codes to detect errors. In most cases this is sufficient, however it has been shown that for the longer packet types the probability of an undetected error is too high to support typical applications, especially those with a large amount of data being transferred.

The L2CAP layer provides an additional level of error control that is designed to detect the occasional undetected errors in the baseband layer and request retransmission of the affected data. This provides the level of reliability required by typical Bluetooth applications.

Broadcast links have no feedback route, and are unable to use the ARQ scheme (although the receiver is still able to detect errors in received packets.) Instead each packet is transmitted several times in the hope that the receiver is able to receive at least one of the copies successfully. Despite this approach there are still no guarantees of successful receipt, and so these links are considered unreliable.

In summary, if a link or channel is characterized as reliable this means that the receiver is capable of detecting errors in received packets and requesting retransmission until the errors are removed. Due to the error detection system used some residual (undetected) errors may still remain in the received data. For L2CAP channels the level of these is comparable to other communication systems, although for logical links the residual error level is somewhat higher.

The transmitter may remove packets from the transmit queue such that the receiver does not receive all the packets in the sequence. If this happens detection of the missing packets is delegated to the L2CAP layer.

On an unreliable link the receiver is capable of detecting errors in received packets but cannot request retransmission. The packets passed on by the receiver may be without error, but there is no guarantee that all packets in the sequence are received. Hence the link is considered fundamentally unreliable. There are limited uses for such links, and these uses are normally dependent on the continuous repetition of data from the higher layers while it is valid.

Stream links have a reliability characteristic somewhere between a reliable and an unreliable link, depending on the current operating conditions.

3.2 TRANSPORT ARCHITECTURE ENTITIES

The Bluetooth transport architecture entities are shown in Figure 3.3 on page 32 and are described from the lowest layer upwards in the subsequent sections.

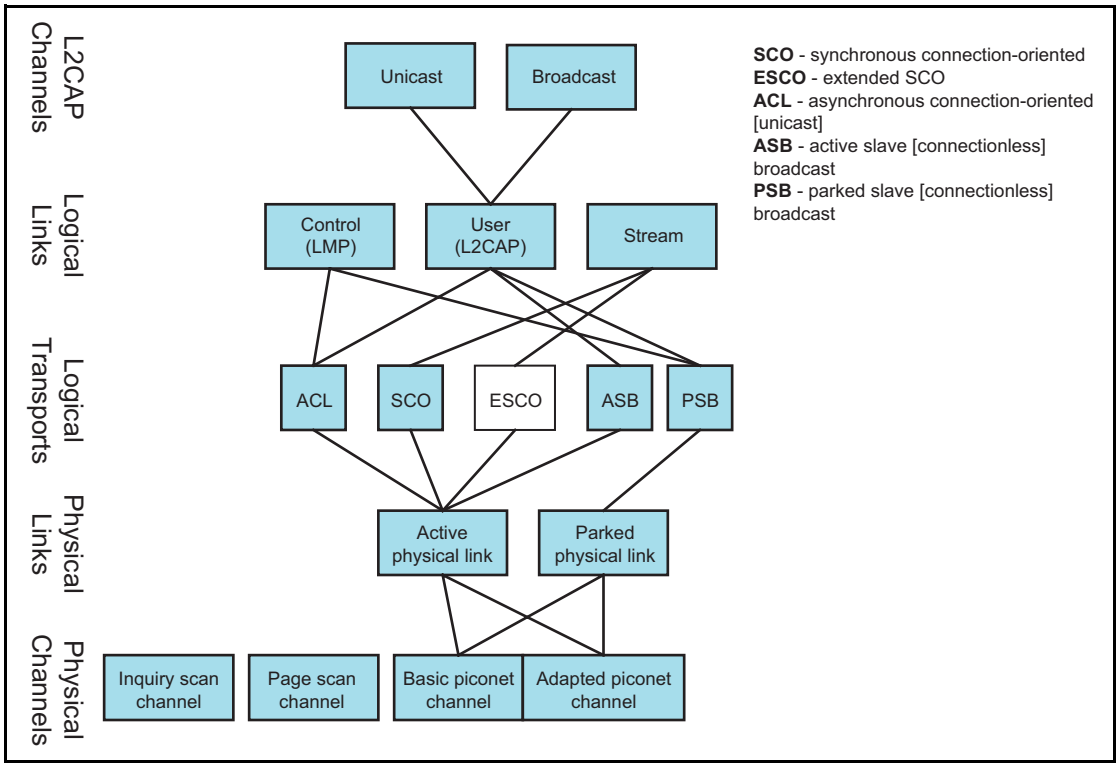


Figure 3.3: Overview of transport architecture entities and hierarchy

3.2.1 Bluetooth generic packet structure

The general packet structure nearly reflects the architectural layers found in the Bluetooth system. The packet structure is designed for optimal use in normal operation. It is shown in Figure 3.4 on page 32.

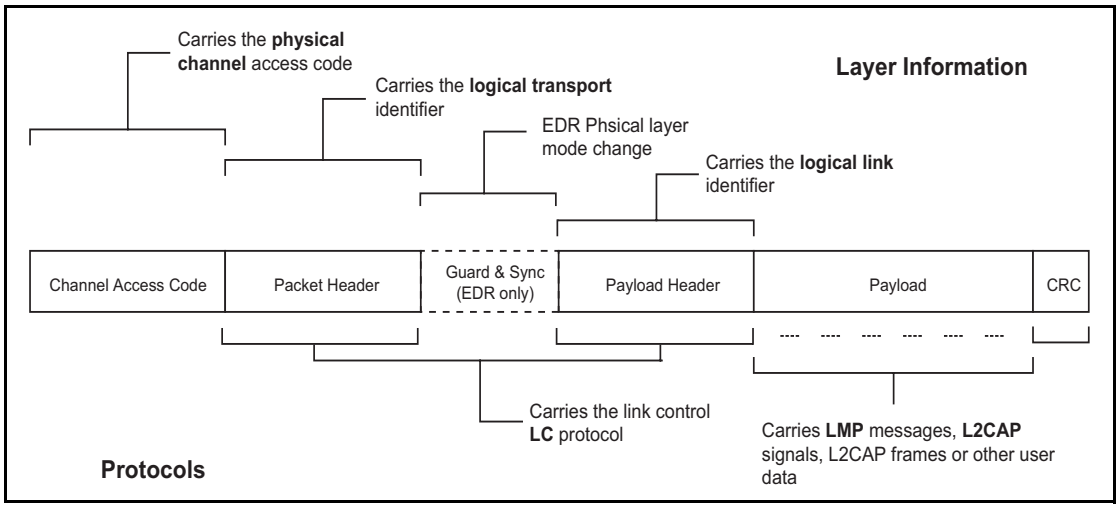


Figure 3.4: Bluetooth packet structure

Packets normally only include the fields that are necessary to represent the layers required by the transaction. Thus a simple inquiry request over an inquiry scan physical channel does not create or require a logical link or higher layer and therefore consists only of the channel access code (associated with the physical channel.) General communication within a piconet uses packets that include all of the fields, as all of the architectural layers are used.

All packets include the channel access code. This is used to identify communications on a particular physical channel, and to exclude or ignore packets on a different physical channel that happens to be using the same RF carrier in physical proximity.

There is no direct field within the Bluetooth packet structure that represents or contains information relating to physical links. This information is implied in the logical transport address (LT_ADDR) carried in the packet header.

Most packets include a packet header. The packet header is always present in packets transmitted on physical channels that support physical links, logical transports and logical links. The packet header carries the LT_ADDR, which is used by each receiving device to determine if the packet is addressed to the device and is used to route the packet internally.

The packet header also carries part of the link control (LC) protocol that is operated per logical transport (except for ACL and SCO transports that operate a shared LC protocol carried on either logical transport.)

The Enhanced Data Rate (EDR) packets have a guard time and synchronization sequence before the payload. This is a field used for physical layer change of modulation scheme.

The payload header is present in all packets on logical transports that support multiple logical links. The payload header includes a logical link identifier field used for routing the payload, and a field indicating the length of the payload. Some packet types also include a CRC after the packet payload that is used to detect most errors in received packets.

EDR packets have a trailer after the CRC.

The packet payload is used to transport the user data. The interpretation of this data is dependent on the logical transport and logical link identifiers. For ACL logical transports Link Manager Protocol (LMP) messages and L2CAP signals are transported in the packet payload, along with general user data from applications. For SCO and eSCO logical transports the payload contains the user data for the logical link.

3.3 PHYSICAL CHANNELS

The lowest architectural layer in the Bluetooth system is the physical channel. A number of types of physical channel are defined. All Bluetooth physical channels are characterized by an RF frequency combined with temporal parameters and restricted by spatial considerations. For the basic and adapted piconet physical channels frequency hopping is used to change frequency periodically to reduce the effects of interference and for regulatory reasons.

Two Bluetooth devices use a shared physical channel for communication. To achieve this their transceivers need to be tuned to the same RF frequency at the same time, and they need to be within a nominal range of each other.

Given that the number of RF carriers is limited and that many Bluetooth devices may be operating independently within the same spatial and temporal area there is a strong likelihood of two independent Bluetooth devices having their transceivers tuned to the same RF carrier, resulting in a physical channel collision. To mitigate the unwanted effects of this collision each transmission on a physical channel starts with an access code that is used as a correlation code by devices tuned to the physical channel. This channel access code is a property of the physical channel. The access code is always present at the start of every transmitted packet.

Four Bluetooth physical channels are defined. Each is optimized and used for a different purpose. Two of these physical channels (the basic piconet channel and adapted piconet channel) are used for communication between connected devices and are associated with a specific piconet. The remaining physical channels are used for discovering Bluetooth devices (the inquiry scan channel) and for connecting Bluetooth devices (the page scan channel.)

A Bluetooth device can only use one of these physical channels at any given time. In order to support multiple concurrent operations the device uses time-division multiplexing between the channels. In this way a Bluetooth device can appear to operate simultaneously in several piconets, as well as being discoverable and connectable.

Whenever a Bluetooth device is synchronized to the timing, frequency and access code of a physical channel it is said to be ‘connected’ to this channel (whether or not it is actively involved in communications over the channel.) The Bluetooth specification assumes that a device is only capable of connecting to one physical channel at any time. Advanced devices may capable of connecting simultaneously to more than one physical channel, but the specification does not assume that this is possible.

3.3.1 Basic piconet channel

3.3.1.1 Overview

The basic piconet channel is used for communication between connected devices during normal operation.

3.3.1.2 Characteristics

The basic piconet channel is characterized by a pseudo-random sequence hopping through the RF channels. The hopping sequence is unique for the piconet and is determined by the Bluetooth device address of the master. The phase in the hopping sequence is determined by the Bluetooth clock of the master. All Bluetooth devices participating in the piconet are time- and hop-synchronized to the channel.

The channel is divided into time slots where each slot corresponds to an RF hop frequency. Consecutive hops correspond to different RF hop frequencies. The time slots are numbered according to the Bluetooth clock of the piconet master. Packets are transmitted by Bluetooth devices participating in the piconet aligned to start at a slot boundary. Each packet starts with the channel's access code, which is derived from the Bluetooth device address of the piconet.

On the basic piconet channel the master controls access to the channel. The master starts its transmission in even-numbered time slots only. Packets transmitted by the master are aligned with the slot start and define the piconet timing. Packets transmitted by the master may occupy up to five time slots depending on the packet type.

Each master transmission is a packet carrying information on one of the logical transports. Slave devices may transmit on the physical channel in response. The characteristics of the response are defined by the logical transport that is addressed.

For example on the asynchronous connection-oriented logical transport the addressed slave device responds by transmitting a packet containing information for the same logical transport that is nominally aligned with the next (odd-numbered) slot start. Such a packet may occupy up to five time slots, depending on the packet type. On a broadcast logical transport no slaves are allowed to respond.

A special characteristic of the basic piconet physical channel is the use of some reserved slots to transmit a beacon train. The beacon train is only used if the piconet physical channel has parked slaves connected to it. In this situation the master transmits a packet in the reserved beacon train slots (these packets are used by the slave to resynchronize to the piconet physical channel.) The master may transmit packets from any logical transport onto these slots, providing there is a transmission starting in each of the slots. In the case where

there is information from the parked slave broadcast (PSB) logical transport to be transmitted then this is transmitted in the beacon train slots and takes priority over any other logical transport.

3.3.1.3 Topology

A basic piconet channel may be shared by any number of Bluetooth devices, limited only by the resources available on the piconet master device. Only one device is the piconet master, all others being piconet slaves. All communication is between the master and slave devices. There is no direct communication between slave devices on the piconet channel.

There is, however, a limitation on the number of logical transports that can be supported within a piconet. This means that although there is no theoretical limit to the number of Bluetooth devices that share a channel there is a limit to the number of these devices that can be actively involved in exchanging data with the master.

3.3.1.4 Supported layers

The basic piconet channel supports a number of physical links, logical transports, logical links and L2CAP channels used for general purpose communications.

3.3.2 Adapted piconet channel

3.3.2.1 Overview

The adapted piconet channel differs from the basic piconet channel in two ways. First the frequencies on which the slaves transmit are the same as the preceding master transmit frequency. In other words the frequency is not recomputed between master and subsequent slave packets. The second way in which the adapted piconet channel differs from the basic piconet channel is that the adapted type can be based on fewer than the full 79 frequencies. A number of frequencies may be excluded from the hopping pattern by being marked as “unused”. The remainder of the 79 frequencies are included. The two sequences are the same except that whenever the basic pseudo-random hopping sequence would have selected an unused frequency it is replaced with an alternative chosen from the used set.

Because the adapted piconet channel uses the same timing and access code as the basic piconet channel, the two channels are often coincident. This provides a deliberate benefit as it allows slaves in either the basic piconet channel or the adapted piconet channel to adjust their synchronization to the master.

The topology and supported layers of the adapted piconet physical channel are identical to the basic piconet physical channel.

3.3.3 Inquiry scan channel

3.3.3.1 Overview

In order for a device to be discovered an inquiry scan channel is used. A discoverable device listens for inquiry requests on its inquiry scan channel and then sends responses to these requests. In order for a device to discover other devices, it iterates (hops) through all possible inquiry scan channel frequencies in a pseudo-random fashion, sending an inquiry request on each frequency and listening for any response.

3.3.3.2 Characteristics

Inquiry scan channels follow a slower hopping pattern and use an access code to distinguish between occasional occupancy of the same radio frequency by two co-located devices using different physical channels.

The access code used on the inquiry scan channel is taken from a reserved set of inquiry access codes that are shared by all Bluetooth devices. One access code is used for general inquiries, and a number of additional access codes are reserved for limited inquiries. Each device has access to a number of different inquiry scan channels. As all of these channels share an identical hopping pattern, a device may concurrently occupy more than one inquiry scan channel if it is capable of concurrently correlating more than one access code.

A device using one of its inquiry scan channel remains passive until it receives an inquiry message on this channel from another Bluetooth device. This is identified by the appropriate inquiry access code. The inquiry scanning device will then follow the inquiry response procedure to return a response to the inquiring device.

In order for a device to discover other Bluetooth devices it uses the inquiry scan channel of these devices in order to send inquiry requests. As it has no prior knowledge of the devices to discover, it cannot know the exact characteristics of the inquiry scan channel.

The device takes advantage of the fact that inquiry scan channels have a reduced number of hop frequencies and a slower rate of hopping. The inquiring device transmits inquiry requests on each of the inquiry scan hop frequencies and listens for an inquiry response. This is done at a faster rate, allowing the inquiring device to cover all inquiry scan frequencies in a reasonably short time period.

3.3.3.3 Topology

Inquiring and discoverable devices use a simple exchange of packets to fulfil the inquiring function. The topology formed during this transaction is a simple and transient point-to-point connection.

3.3.3.4 Supported layers

During the exchange of packets between an inquiring and discoverable device it may be considered that a temporary physical link exists between these devices. However, the concept is quite irrelevant as it has no physical representation but is only implied by the brief transaction between the devices. No further architectural layers are considered to be supported.

3.3.4 Page scan channel

3.3.4.1 Overview

A connectable device (one that is prepared to accept connections) does so using an page scan channel. A connectable device listens for page requests on its page scan channel and enters into a sequence of exchanges with this device. In order for a device to connect to another device, it iterates (hops) through all page scan channel frequencies in a pseudo-random fashion, sending an page request on each frequency and listening for any response.

3.3.4.2 Characteristics

The page scan channel uses an access code derived from the scanning device's Bluetooth device address to identify communications on the channel. The page scan channel uses a slower hopping rate than the hop rate of the basic and adapted piconet channels. The hop selection algorithm uses the Bluetooth device clock of the scanning device as an input.

A device using its page scan channel remains passive until it receives a page request from another Bluetooth device. This is identified by the page scan channel access code. The two devices will then follow the page procedure to form a connection. Following a successful conclusion of the page procedure both devices switch to the basic piconet channel that is characterized by having the paging device as master.

In order for a device to connect to another Bluetooth device it uses the page scan channel of the target device in order to send page requests. If the paging device does not know the phase of the target device's page scan channel it therefore does not know the current hop frequency of the target device. The paging device transmits page requests on each of the page scan hop frequencies and listens for a page response. This is done at a faster hop rate, allowing

the paging device to cover all page scan frequencies in a reasonably short time period.

The paging device may have some knowledge of the target device's Bluetooth clock (indicated during a previous inquiry transaction between the two devices, or as a result of a previous involvement in a piconet with the device), in which case it is able to predict the phase of the target device's page scan channel. It may use this information to optimize the synchronization of the paging and page scanning process and speed up the formation of the connection.

3.3.4.3 Topology

Paging and connectable devices use a simple exchange of packets to fulfil the paging function. The topology formed during this transaction is a simple and transient point-to-point connection.

3.3.4.4 Supported layers

During the exchange of packets between an paging and connectable device it may be considered that a temporary physical link exists between these devices. However, the concept is quite irrelevant as it has no physical representation but is only implied by the brief transaction between the devices. No further architectural layers are considered to be supported.

3.4 PHYSICAL LINKS

A physical link represents a baseband connection between Bluetooth devices. A physical link is always associated with exactly one physical channel (although a physical channel may support more than one physical link.)

Within the Bluetooth system a physical link is a virtual concept that has no direct representation within the structure of a transmitted packet. The access code packet field, together with the clock and address of the master Bluetooth device are used to identify a physical channel. However there is no subsequent part of the packet that directly identifies the physical link. Instead the physical link may be identified by association with the logical transport, as each logical transport is only received on one physical link.

Some physical link types have properties that may be modified. An example of this is the transmit power for the link. Other physical link types have no such properties. In the case of physical links with modifiable properties the LM protocol is used to adapt these properties. As the LM protocol is supported at a higher layer (by a logical link) the appropriate physical link is identified by implication from the logical link that transports the LM signalling.

In the situation where a transmission is broadcasted over a number of different physical links, then the transmission parameters are selected to be suitable for all of the physical links.

3.4.1 Links supported by the basic and adapted piconet physical channel

The basic and adapted piconet physical channels support a physical link which may be active or parked. The physical link is a point-to-point link between the master and a slave. It is always present when the slave is synchronized in the piconet.

3.4.1.1 Active physical link

The physical link between a master and a slave device is active if a default ACL logical transport exists between the devices. Active physical links have no direct identification of their own, but are identified by association with the default ACL logical transport ID with which there is a one-to-one correspondence.

An active physical link has the associated properties of radio transmit power in each direction. Transmissions from slave devices are always directed over the active physical link to the master, and use the transmit power that is a property of this link in the slave to master direction. Transmissions from the master may be directed over a single active physical link (to a specific slave) or over a number of physical links (to a group of slaves in the piconet.) In the case of point-to-point transmissions the master uses the appropriate transmit power for the physical link in question. (In the case of point-to-multipoint transmissions the master uses a transmit power appropriate for the set of devices addressed.)

Active physical links may be placed into Hold or Sniff mode. The effect of these modes is to modify the periods when the physical link is active and may carry traffic. Logical transports that have defined scheduling characteristics are not affected by these modes and continue according to their pre-defined scheduling behavior. The default ACL logical transport and other links with undefined scheduling characteristics are subject to the mode of the active physical link.

3.4.1.2 Parked physical link

The physical link between a master and a slave device is parked when the slave remains synchronized in the piconet, but has no default ACL logical transport. Such a slave is also said to be parked. A beacon train is used to provide regular synchronization to all parked slaves connected to the piconet physical channel. A parked slave broadcast (PSB) logical transport is used to allow communication of a subset of LMP signalling and broadcast L2CAP to parked slaves. The PSB logical transport is closely associated with the beacon train.

A slave is parked (its active link is changed to a parked link) using the park procedure. The master is not allowed to park a slave that has any user created logical transport supported by the physical link. These logical transports are first removed, and any L2CAP channels that are built on these logical transports are also removed. The broadcast logical transport and default ACL logical transports are not considered as user created and are not explicitly removed. When the active link is replaced with a parked link the default ACL

logical transport is implicitly removed. The supported logical links and L2CAP channels remain in existence, but become suspended. It is not possible to use these links and L2CAP channels to transport signalling or data while the active link is absent.

A parked slave may become active using the unpark procedure. This procedure is requested by the slave at an access window and initiated by the master. Following the unpark procedure the parked physical link is changed to an active physical link and the default ACL logical transport is re-created. L2CAP channels that were suspended during the most recent park procedure are associated with the new default ACL logical transport and become active again.

Parked links do not support radio power control, as there is no feedback path from parked slaves to the piconet master that can be used to signal received signal strength at the slave or for the master to measure received signal strength from the slave. Transmissions are carried out at nominal power on parked links.

Parked links use the same physical channel as their associated active link. If a master manages a piconet that contains parked slaves using the basic piconet physical channel and also parked slaves using the adapted piconet physical channel then it must create a parked slave broadcast logical transport (and associated transport) for each of these physical channels.

A parked slave may use the inactive periods of the parked slave broadcast logical transport to save power, or it may carry out activities on other physical channels unrelated to the piconet within which it is parked.

3.4.2 Links supported by the scanning physical channels

In the case of the inquiry scan and page scan channels the physical link exists for a relatively short time and cannot be controlled or modified in any way. These types of physical link are not further elaborated.

3.5 LOGICAL LINKS AND LOGICAL TRANSPORTS

A variety of logical links are available to support different application data transport requirements. Each logical link is associated with a logical transport, which has a number of characteristics. These characteristics include flow control, acknowledgement/repeat mechanisms, sequence numbering and scheduling behavior. Logical transports are able to carry different types of logical links (depending on the type of the logical transport). In the case of some of the Bluetooth 1.1 logical links these are multiplexed onto the same logical transport. Logical transports may be carried by active physical links on either the basic or the adapted piconet physical channel.

Logical transport identification and real-time (link control) signalling are carried in the packet header, and for some logical links identification is carried in the

payload header. Control signalling that does not require single slot response times is carried out using the LMP protocol.

Table 3.1 on page 42 lists all of the logical transport types, the supported logical link types, which type of physical links and physical channels can support them, and a brief description of the purpose of the logical transport.

Logical transport	Links supported	Supported by	Overview
Asynchronous Connection-Oriented (ACL[1])	Control (LMP) ACL-C User (L2CAP) ACL-U	Active physical link, basic or adapted physical channel	Reliable or time-bounded, bi-directional, point-to-point.
Synchronous Connection-Oriented (SCO)	Stream (unframed) SCO-S	Active physical link, basic or adapted physical channel	Bi-directional, symmetric, point-to-point, AV channels. Used for 64Kb/s constant rate data.
Extended Synchronous Connection-Oriented (eSCO)	Stream (unframed) eSCO-S	Active physical link, basic or adapted physical channel	Bi-directional, symmetric or asymmetric, point-to-point, general regular data, limited retransmission. Used for constant rate data synchronized to the master Bluetooth clock.
Active slave broadcast (ASB)	User (L2CAP) ASB-U	Active physical link, basic or adapted physical channel.	Unreliable, uni-directional broadcast to any devices synchronized with the physical channel. Used for broadcast L2CAP groups.
Parked slave broadcast (PSB)	Control (LMP) PSB-C, User (L2CAP) PSB-U	Parked physical link, basic or adapted physical channel.	Unreliable, uni-directional broadcast to all piconet devices. Used for LMP and L2CAP traffic to parked devices, and for access requests from parked devices.

Table 3.1: Logical transport types.

1. It is clear that the most obvious abbreviation for Asynchronous Connection-Oriented logical transport is ACO. However, this acronym has an alternative meaning from the Bluetooth 1.1 specification. To avoid confusion between two possible meanings for ACO the decision was made to retain the ACL abbreviation for the Asynchronous Connection-oriented Logical transport.

The names given to the logical links and logical transports reflect some of the names used in Bluetooth 1.1, in order to provide some degree of familiarity and continuation. However these names do not reflect a consistent scheme, which is outlined below.

The classification of each link type follows from a selection procedure within three categories.

3.5.1 Casting

The first category is that of casting. This may be either unicast or broadcast. There are no multicast links defined in Bluetooth 1.2,

- *Unicast links.* Unicast links exist between exactly two endpoints. Traffic may be sent in either direction on unicast links. All unicast links are connection-oriented, meaning that a connection procedure takes place before the link may be used. In the case of the default ACL links, the connection procedure is an implicit step within the general paging procedure used to form ad-hoc piconets.
- *Broadcast links.* Broadcast links exist between one source device and zero or more receiver devices. Traffic is unidirectional, i e only sent from the source devices to the receiver devices. Broadcast links are connectionless, meaning that there is no procedure to create these links, and data may be sent over them at any time. Broadcast links are unreliable, and there is no guarantee that the data will be received.

3.5.2 Scheduling and acknowledgement scheme

The second category relates to the scheduling and acknowledgement scheme of the link, and implies the type of traffic that is supported by the link. These are synchronous, isochronous or asynchronous. There are no specific isochronous links defined in Bluetooth 1.2, though the default ACL link can be configured to operate in this fashion.

- *Synchronous links.* Synchronous links provide a method of associating the Bluetooth piconet clock with the transported data. This is achieved by reserving regular slots on the physical channel, and transmitting fixed size packets at these regular intervals. Such links are suitable for constant rate isochronous data.
- *Asynchronous links.* Asynchronous links provide a method for transporting data that has no time-based characteristics. The data is normally expected to be retransmitted until successfully received, and each data entity can be processed at any time after receipt, without reference to the time of receipt of any previous or successive entity in the stream (providing the ordering of data entities is preserved.)
- *Isochronous links.* Isochronous links provide a method for transporting data that has time-based characteristics. The data may be retransmitted until received or expired. The data rate on the link need not be constant (this being the main difference from synchronous links.)

3.5.3 Class of data

The final category is related to the class of data that is carried by the link. This is either control (LMP) data or user data. The user data category is sub-divided into L2CAP (or framed) data and stream (or unframed) data.

- *Control links.* Control links are only used for transporting LMP messages between two link managers. These links are invisible above the baseband layer, and cannot be directly instantiated, configured or released by applications, other than by the use of the connection and disconnection services that have this effect implicitly. Control links are always multiplexed with an equivalent L2CAP link onto an ACL logical transport. Subject to the rules defining the ARQ scheme, the control link traffic always takes priority over the L2CAP link traffic.
- *L2CAP links.* L2CAP links are used to transport L2CAP PDUs, which may carry the L2CAP signalling channel (on the default ACL-U logical link only) or framed user data submitted to user-instantiated L2CAP channels. L2CAP frames submitted to the baseband may be larger than the available baseband packets. A link control protocol embedded within the LLID field preserves the frame-start and frame-continuation semantics when the frame is transmitted in a number of fragments to the receiver.
- *Stream links.* Stream links are used to transport user data that has no inherent framing that should be preserved when delivering the data. Lost data may be replaced by padding at the receiver.

3.5.4 Asynchronous connection-oriented (ACL)

The asynchronous connection-oriented (ACL) logical transport is used to carry LMP and L2CAP control signalling and best effort asynchronous user data. The ACL logical transport uses a simple 1-bit ARQN/SEQN scheme to provide simple channel reliability. Every active slave device within a piconet has one ACL logical transport to the piconet master, known as the default ACL.

The default ACL is created between the master and the slave when a device joins a piconet (connects to the basic piconet physical channel). This default ACL is assigned a logical transport address (LT_ADDR) by the piconet master. This LT_ADDR is also used to identify the active physical link when required (or as a piconet active member identifier, effectively for the same purpose.)

The LT_ADDR for the default ACL is reused for synchronous connection-oriented logical transports between the same master and slave. (This is for reasons of compatibility with earlier Bluetooth specifications.) Thus the LT_ADDR is not sufficient on its own to identify the default ACL. However the packet types used on the ACL are different from those used on the synchronous connection-oriented logical transport. Therefore, the ACL logical transport can be identified by the LT_ADDR field in the packet header in combination with the packet type field.

The default ACL may be used for isochronous data transport by configuring it to automatically flush packets after the packets have expired.

If the default ACL is removed from the active physical link then all other logical transports that exist between the master and the slave are also removed. In the case of unexpected loss of synchronization to the piconet physical channel the physical link and all logical transports and logical links cease to exist at the time that this synchronization loss is detected.

A device may remove its default ACL (and by implication its active physical link) but remain synchronized to the piconet. This procedure is known as parking, and a device that is synchronized to the piconet, but has no active physical link is parked within that piconet.

When the device transitions to the parked state the default ACL logical links that are transported on the default ACL logical transport remain in existence, but become suspended. No data may be transferred across a suspended logical link. When the device transitions from the parked state back into active state, a new default ACL logical transport is created (it may have a different LT_ADDR from the previous one) and the suspended logical links are attached to this default ACL and become active once again.

3.5.5 Synchronous connection-oriented (SCO)

The synchronous connection-oriented (SCO) logical transport is a symmetric, point-to-point channel between the master and a specific slave. The SCO logical transport reserves slots on the physical channel and can therefore be considered as a circuit-switched connection between the master and the slave. SCO logical transports carry 64 kb/s of information synchronized with the piconet clock. Typically this information is an encoded voice stream. Three different SCO configurations exist, offering a balance between robustness, delay and bandwidth consumption.

Each SCO-S logical link is supported by a single SCO logical transport, which is assigned the same LT_ADDR as the default ACL logical transport between the devices. Therefore the LT_ADDR field is not sufficient to identify the destination of a received packet. Because the SCO links use reserved slots, a device uses a combination of the LT_ADDR, the slot numbers (a property of the physical channel) and the packet type to identify transmissions on the SCO link.

The reuse of the default ACL's LT_ADDR for SCO logical transports is due to legacy behavior from the Bluetooth 1.1 specification. In this earlier version of Bluetooth the LT_ADDR (then known as the active member address) was used to identify the piconet member associated with each transmission. This was not easily extensible for enabling more logical links, and so the purpose of this field was redefined for the new features. Some Bluetooth 1.1 features, however, do not cleanly fit into the more formally described architecture.

Although slots are reserved for the SCO, it is permissible to use a reserved slot for traffic from another channel that has a higher priority. This may be required as a result of QoS commitments, or to send LMP signalling on the default ACL when the physical channel bandwidth is fully occupied by SCOs. As SCOs carry different packet types to ACLs, the packet type is used to identify SCO traffic (in addition to the slot number and LT_ADDR.)

There are no further architectural layers defined by the Bluetooth core specification that are transported over an SCO link. A number of standard formats are defined for the 64 kb/s stream that is transported, or an unformatted stream is allowed where the application is responsible for interpreting the encoding of the stream.

3.5.6 Extended synchronous connection-oriented (eSCO)

The extended synchronous connection-oriented (eSCO) logical transport is a symmetric or asymmetric, point-to-point link between the master and a specific slave. The eSCO reserves slots on the physical channel and can therefore be considered as a circuit-switched connection between the master and the slave. eSCO links offer a number of extensions over the standard SCO links, in that they support a more flexible combination of packet types and selectable data contents in the packets and selectable slot periods, allowing a range of synchronous bit rates to be supported.

eSCO links also can offer limited retransmission of packets (unlike SCO links where there is no retransmission.) If these retransmissions are required they take place in the slots that follow the reserved slots, otherwise the slots may be used for other traffic.

Each eSCO-S logical link is supported by a single eSCO logical transport, identified by a LT_ADDR that is unique within the piconet for the duration of the eSCO. eSCO-S links are created using LM signalling and follow scheduling rules similar to SCO-S links.

There are no further architectural layers defined by the Bluetooth core specification that are transported over an eSCO-S link. Instead applications may use the data stream for whatever purpose they require, subject to the transport characteristics of the stream being suitable for the data being transported.

3.5.7 Active slave broadcast (ASB)

The active slave broadcast logical transport is used to transport L2CAP user traffic to all devices in the piconet that are currently connected to the physical channel that is used by the ASB. There is no acknowledgement protocol and the traffic is uni-directional from the piconet master to the slaves. The ASB channel may be used for L2CAP group traffic (a legacy of the 1.1 specification), and is never used for L2CAP connection-oriented channels, L2CAP control signalling or LMP control signalling.

The ASB logical transport is inherently unreliable because of the lack of acknowledgement. To improve the reliability, each packet is transmitted a number of times. An identical sequence number is used to assist with filtering retransmissions at the slave device.

The ASB logical transport is identified by a reserved LT_ADDR. (The reserved LT_ADDR address is also used by the PSB logical transport.) An active slave will receive traffic on both logical transports, and cannot readily distinguish between them. As the ASB logical transport does not carry LMP traffic an active slave can ignore packets received over the LMP logical link on the ASB logical transport. However L2CAP traffic transmitted over the PSB logical transport is also received by active slaves on the ASB logical transport and cannot be distinguished from L2CAP traffic sent on the ASB transport.

An ASB is implicitly created whenever a piconet exists, and there is always one ASB associated with each of the basic and adapted piconet physical channels that exist within the piconet. Because the basic and adapted piconet physical channels are mostly coincident a slave device cannot distinguish which of the ASB channels is being used to transmit the packets. This adds to the general unreliability of the ASB channel. (Although it is, perhaps, no more unreliable than general missed packets.)

A master device may decide to use only one of its two possible ASBs (when it has both a basic and adapted piconet physical channel), as with sufficient retransmissions it is possible to address both groups of slaves on the same ASB channel.

The ASB channel is never used to carry LMP or L2CAP control signals.

3.5.8 Parked slave broadcast (PSB)

The parked slave broadcast logical transport is used for communications between the master and slaves that are parked (have given up their default ACL logical transport.) The parked slave broadcast link is the only logical transport that exists between the piconet master and parked slaves.

The PSB logical transport is more complex than the other logical transports as it consists of a number of phases, each having a different purpose. These phases are the control information phase (used to carry the LMP logical link), the user information phase (used to carry the L2CAP logical link), and the access phase (carrying baseband signalling.) The control information and broadcast information phases are usually mutually exclusive as only one of them can be supported in a single beacon interval. (Even if there is no control or user information phase, the master is still required to transmit a packet in the beacon slots so that the parked slaves can resynchronize.) The access phase is normally present unless cancelled in a control information message.

The control information phase is used for the master to send information to the parked slaves containing modifications to the PSB transport attributes, modifi-

cations to the beacon train attributes, or a request for a parked slave to become active in the piconet (known as unparking). This control information is carried in LMP messages on the LMP logical link. (The control information phase is also present in the case of a user information phase where the user information requires more than one baseband packet.)

Packets in the control information phase are always transmitted in the physical channel beacon train slots, and cannot be transmitted on any other slots. The control information occupies a single DM1 packet and is repeated in every beacon train slot within a single beacon interval. (If there is no control information then there may be a user information phase that uses the beacon slots. If neither phase is used then the beacon slots are used for other logical transport traffic or for NULL packets.)

The user information phase is used for the master to send L2CAP packets that are destined for all piconet slaves. User information may occupy one or more baseband packets. If the user information occupies a single packet then the user information packet is repeated in each of the piconet channel beacon train slots.

If the user information occupies more than one baseband packet then it is transmitted in slots after the beacon train (the broadcast scan window) and the beacon slots are used to transmit a control information phase message that contains the timing attributes of this broadcast scan window. This is required so that the parked slaves remain connected to the piconet physical channel to receive the user information.

The access phase is normally present unless temporarily cancelled by a control message carried in the control information broadcast phase. The access window consists of a sequence of slots that follow the beacon train. In order for a parked slave to become active in the piconet, it may send such an access request to the piconet master during the access window. Each parked slave is allocated an access request address (not necessarily unique) that controls when during the access window the slave requests access.

The PSB logical transport is identified by the reserved LT_ADDR of 0. This reserved LT_ADDR address is also used by the ASB logical transport. Parked slaves are not normally confused by the duplicated use of the LT_ADDR as they are only connected to the piconet physical channel during the time that the PSB transport is being used.

3.5.9 Logical links

Some logical transports are capable of supporting different logical links, either concurrently multiplexed, or one of the choice. Within such logical transports, the logical link is identified by the logical link identifier (LLID) bits in the payload header of baseband packets that carry a data payload. The logical links distinguish between a limited set of core protocols that are able to transmit and receive data on the logical transports. Not all of the logical transports are able to carry all of the logical links (the supported mapping is shown in Figure 3.2 on

page 28.) In particular the SCO and eSCO logical transports are only able to carry constant data rate streams, and these are uniquely identified by the LT_ADDR. Such logical transports only use packets that do not contain a payload header, as their length is known in advance, and no LLID is necessary.

3.5.10 ACL Control Logical Link (ACL-C)

The ACL Control Logical Link (ACL-C) is used to carry LMP signalling between devices in the piconet. The control link is only carried on the default ACL logical transport and on the PSB logical transport (in the control information phase). The ACL-C link is always given priority over the ACL-U (see below) link when carried on the same logical transport.

3.5.11 User Asynchronous/Isochronous Logical Link (ACL-U)

The user asynchronous/isochronous logical link (ACL-U) is used to carry all asynchronous and isochronous framed user data. The ACL-U link is carried on all but the synchronous logical transports. Packets on the ACL-U link are identified by one of two reserved LLID values. One value is used to indicate whether the baseband packet contains the start of an L2CAP frame and the other indicates a continuation of a previous frame. This ensures correct synchronization of the L2CAP reassembler following flushed packets. The use of this technique removes the need for a more complex L2CAP header in every baseband packet (the header is only required in the L2CAP start packets), but adds the requirement that a complete L2CAP frame shall be transmitted before a new one is transmitted. (An exception to this rule being the ability to flush a partially transmitted L2CAP frame in favour of another L2CAP frame.)

3.5.12 User Synchronous/Extended Synchronous Logical Links (SCO-S/eSCO-S)

Synchronous (SCO-S) and extended synchronous (eSCO-S) logical links are used to support isochronous data delivered in a stream without framing. These links are associated with a single logical transport, where data is delivered in constant sized units at a constant rate. There is no LLID within the packets on these transports, as only a single logical link can be supported, and the packet length and scheduling period are pre-defined and remain fixed during the lifetime of the link.

Variable rate isochronous data cannot be carried by the SCO-S or eSCO-S logical links. In this case the data must be carried on ACL-U logical links, which use packets with a payload header. Bluetooth has some limitations when supporting variable-rate isochronous data concurrently with reliable user data.

3.6 L2CAP CHANNELS

L2CAP provides a multiplexing role allowing many different applications to share the resources of an ACL-U logical link between two devices. Applications and service protocols interface with L2CAP using a channel-oriented interface to create connections to equivalent entities on other devices.

L2CAP channel endpoints are identified to their clients by a Channel Identifier (CID). This is assigned by L2CAP, and each L2CAP channel endpoint on any device has a different CID.

L2CAP channels may be configured to provide an appropriate Quality of Service (QoS) to the application. L2CAP maps the channel onto the ACL-U logical link.

L2CAP supports channels that are connection-oriented and others that are group-oriented. Group-oriented channels may be mapped onto the ASB-U logical link, or implemented as iterated transmission to each member in turn over an ACL-U logical link.

Apart from the creation, configuration and dismantling of channels, the main role of L2CAP is to multiplex service data units (SDUs) from the channel clients onto the ACL-U logical links, and to carry out a simple level of scheduling, selecting SDUs according to relative priority.

L2CAP can provide a per channel flow control with the peer L2CAP layer. This option is selected by the application when the channel is established. L2CAP can also provide enhanced error detection and retransmission to (a) reduce the probability of undetected errors being passed to the application and (b) recover from loss of portions of the user data when the Baseband layer performs a flush on the ACL-U logical link.

In the case where an HCI is present, the L2CAP is also required to segment L2CAP SDUs into fragments that will fit into the baseband buffers, and also to operate a token based flow control procedure over the HCI, submitting fragments to the baseband only when allowed to do so. This may affect the scheduling algorithm.

4 COMMUNICATION TOPOLOGY

4.1 PICONET TOPOLOGY

Any time a Bluetooth link is formed it is within the context of a piconet. A piconet consists of two or more devices that occupy the same physical channel (which means that they are synchronized to a common clock and hopping sequence.) The common (piconet) clock is identical to the Bluetooth clock of one of the devices in the piconet, known as the master of the piconet, and the hopping sequence is derived from the master's clock and the master's Bluetooth device address. All other synchronized devices are referred to as slaves in the piconet. The terms master and slave are only used when describing these roles in a piconet.

Within a common location a number of independent piconets may exist. Each piconet has a different physical channel (that is a different master device and an independent piconet clock and hopping sequence.)

A Bluetooth device may participate concurrently in two or more piconets. It does this on a time-division multiplexing basis. A Bluetooth device can never be a master of more than one piconet. (Since the piconet is defined by synchronization to the master's Bluetooth clock it is impossible to be the master of two or more piconets.) A Bluetooth device may be a slave in many independent piconets.

A Bluetooth device that is a member of two or more piconets is said to be involved in a scatternet. Involvement in a scatternet does not necessarily imply any network routing capability or function in the Bluetooth device. The Bluetooth core protocols do not, and are not intended to offer such functionality, which is the responsibility of higher level protocols and is outside the scope of the Bluetooth core specification.

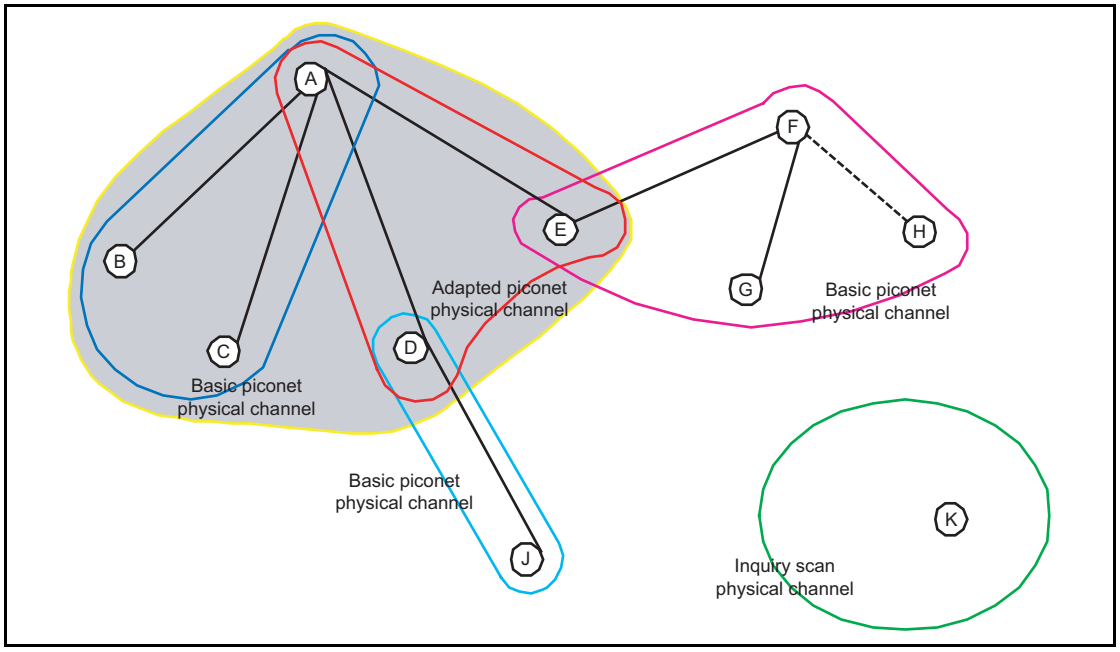


Figure 4.1: Example Bluetooth topology

In Figure 4.1 on page 52 an example topology is shown that demonstrates a number of the architectural features described below. Device A is a master in a piconet (represented by the shaded area, and known as piconet A) with devices B, C, D and E as slaves. Two other piconets are shown: a) one piconet with device F as master (known as piconet F) and devices E, G and H as slaves and b) one piconet with device D as master (known as piconet D) and device J as slave.

In piconet A there are two physical channels. Devices B and C are using the basic piconet physical channel (represented by the blue enclosure) as they do not support adaptive frequency hopping. Devices D and E are capable of supporting adaptive frequency hopping, and are using the adapted piconet physical channel (represented by the red enclosure.) Device A is capable of adaptive frequency hopping, and operates in a TDM basis on both physical channels according to which slave is being addressed.

Piconet D and piconet F are both using only a basic piconet physical channel (represented by the cyan and magenta enclosures respectively.) In the case of piconet D this is because device J does not support the adaptive hopping mode. Although device D supports adaptive hopping it cannot use it in this piconet. In piconet F device F does not support adaptive hopping, and therefore it cannot be used in this piconet.

Device K is shown in the same locality as the other devices. It is not currently a member of a piconet, but has services that it offers to other Bluetooth devices. It is currently listening on its inquiry scan physical channel (represented by the green enclosure), awaiting an inquiry request from another device.

Physical links (one per slave device) are represented in the diagram by lines connecting the devices. The solid lines represent an active physical link, and the dashed line represents a parked physical link. Device H is parked, and hence the physical link between the master (F) and the slave (H) is shown as parked.

Logical transports, logical links and L2CAP channels are used to provide capabilities for the transport of data, but are not shown on this diagram. They are described in more detail in Section 3.5 on page 41 and Section 3.6 on page 50.

4.2 OPERATIONAL PROCEDURES AND MODES

The typical operational mode of a Bluetooth device is to be connected to other Bluetooth devices (in a piconet) and exchanging data with that Bluetooth device. As Bluetooth is an ad-hoc wireless communications technology there are also a number of operational procedures that enable piconets to be formed so that the subsequent communications can take place. Procedures and modes are applied at different layers in the architecture and therefore a device may be engaged in a number of these procedures and modes concurrently.

4.2.1 Inquiry (Discovering) Procedure

Bluetooth devices use the inquiry procedure to discover nearby devices, or to be discovered by devices in their locality.

The inquiry procedure is asymmetrical. A Bluetooth device that tries to find other nearby devices is known as an inquiring device and actively sends inquiry requests. Bluetooth devices that are available to be found are known as discoverable devices and listen for these inquiry requests and send responses. The inquiry procedure uses a special physical channel for the inquiry requests and responses.

Both inquiring and discoverable devices may already be connected to other Bluetooth devices in a piconet. Any time spent inquiring or occupying the inquiry scan physical channel needs to be balanced with the demands of the QoS commitments on existing logical transports.

The inquiry procedure does not make use of any of the architectural layers above the physical channel, although a transient physical link may be considered to be present during the exchange of inquiry and inquiry response information.

4.2.2 Paging (Connecting) Procedure

The procedure for forming connections is asymmetrical and requires that one Bluetooth device carries out the page (connection) procedure while the other Bluetooth device is connectable (page scanning.) The procedure is targeted, so that the page procedure is only responded to by one specified Bluetooth device.

The connectable device uses a special physical channel to listen for connection request packets from the paging (connecting) device. This physical channel has attributes that are specific to the connectable device, hence only a paging device with knowledge of the connectable device is able to communicate on this channel.

Both paging and connectable devices may already be connected to other Bluetooth devices in a piconet. Any time spent paging or occupying the page scan physical channel needs to be balanced with the demands of the QoS commitments on existing logical transports.

4.2.3 Connected mode

After a successful connection procedure, the devices are physically connected to each other within a piconet. This means that there is a piconet physical channel to which they are both connected, there is a physical link between the devices, and there are default ACL-C and ACL-U logical links. When in the connected mode it is possible to create and release additional logical links, and to change the modes of the physical and logical links while remaining connected to the piconet physical channel. It is also possible for the device to carry out inquiry, paging or scanning procedures or to be connected to other piconets without needing to disconnect from the original piconet physical channel.

Additional logical links are created using the Link Manager that exchanges Link Manager Protocol messages with the remote Bluetooth device to negotiate the creation and settings for these links. Default ACL-C and ACL-U logical links are always created during the connection process, and these are used for LMP messages and the L2CAP signalling channel respectively.

It is noted that two default logical links are created when two units are initially connected. One of these links (ACL-C) transports the LMP control protocol and is invisible to the layers above the Link Manager. The other link (ACL-U) transports the L2CAP signalling protocol and any multiplexed L2CAP best-effort channels. It is common to refer to a default ACL logical transport, which can be resolved by context, but typically refers to the default ACL-U logical link. Also note that these two logical links share a logical transport.

During the time that a slave device is actively connected to a piconet there is always a default ACL logical transport between the slave and the master device. There are two methods of deleting the default ACL logical transport. The first method is to detach the device from the piconet physical channel, at

which time the entire hierarchy of L2CAP channels, logical links and logical transports between the devices is deleted.

The second method is to place the physical link to the slave device in the park state, at which time it gives up its default ACL logical transport. This is only allowed if all other logical transports have been deleted (except for the ASB logical transport that cannot be explicitly created or deleted.) It is not allowed to park a device while it has any logical transports other than the default ACL and ASB logical transports.

When the slave device physical link is parked, its default ACL logical transport is released and the ASB logical transport is replaced with a PSB logical transport. The ACL-C and ACL-U logical links that are multiplexed onto the default ACL logical transport remain in existence but cannot be used for the transport of data. The Link Manager on the master device restricts itself to the use of LMP messages that are allowed to be transported over the PSB-C logical link. The Channel Manager and L2CAP Resource Manager ensure that no L2CAP unicast data traffic is submitted to the controller while the device is parked. The Channel Manager may decide to manage the parking and unparking of the device as necessary to allow data to be transported.

4.2.4 Hold mode

Hold mode is not a general device mode, but applies to unreserved slots on the physical link. When in this mode, the physical link is only active during slots that are reserved for the operation of the synchronous link types SCO and eSCO. All asynchronous links are inactive. Hold modes operate once for each invocation and are then exited when complete, returning to the previous mode.

4.2.5 Sniff mode

Sniff mode is not a general device mode, but applies to the default ACL logical transports. When in this mode the availability of these logical transports is modified by defining a duty cycle consisting of periods of presence and absence. Devices that have their default ACL logical transports in sniff mode may use the absent periods to engage in activity on another physical channel, or to enter reduced power mode. Sniff mode only affects the default ACL logical transports (i.e. their shared ACL logical transport), and does not apply to any additional SCO or eSCO logical transports that may be active. The periods of presence and absence of the physical link on the piconet physical channel is derived as a union of all logical transports that are built on the physical link.

Note that broadcast logical transports have no defined expectations for presence or absence. A master device should aim to schedule broadcasts to coincide with periods of physical link presence within the piconet physical channel, but this may not always be possible or practical. Repetition of broadcasts is defined to improve the possibilities for reaching multiple slaves without overlapping presence periods. However, broadcast logical transports cannot be considered to be reliable.

4.2.6 Parked state

A slave device may remain connected to a piconet but have its physical link in the parked state. In this state the device cannot support any logical links to the master with the exception of the PSB-C and PSB-U logical links that are used for all communication between the piconet master and the parked slave.

When the physical link to a slave device is parked this means that there are restrictions on when the master and slave may communicate, defined by the PSB logical transport parameters. During times when the PSB logical transport is inactive (or absent) then the devices may engage in activity on other physical channels, or enter reduced power mode.

4.2.7 Role switch procedure

The role switch procedure is a method for swapping the roles of two devices connected in a piconet. The procedure involves moving from the physical channel that is defined by the original master device to the physical channel that is defined by the new master device. In the process of swapping from one physical channel to the next, the hierarchy of physical links and logical transports are removed and rebuilt, with the exception of the ASB and PSB logical transports that are implied by the topology and are not preserved. After the role switch, the original piconet physical channel may cease to exist or may be continued if the original master had other slaves that are still connected to the channel.

The procedure only copies the default ACL logical links and supporting layers to the new physical channel. Any additional logical transports are not copied by this procedure, and if required this must be carried out by higher layers. The LT_ADDRs of any affected transports may not be preserved as the values may already be in use on the new physical channel.

If there are any QoS commitments or modes such as sniff mode on the original logical transports, then these are not preserved after a role switch. These must be renegotiated after the role switch has completed.

4.2.8 Enhanced Data Rate

Enhanced Data Rate is a method of extending the capacity and types of Bluetooth packets for the purposes of increasing the maximum throughput, providing better support for multiple connections, and lowering power consumption, while the remainder of the architecture is unchanged.

Enhanced Data Rate may be selected as a mode that operates independently on each logical transport. Once enabled, the packet type bits in the packet header are interpreted differently from their meaning in Basic Rate mode. This different interpretation is clarified in conjunction with the logical transport address field in the header. The result of this interpretation allows the packet payload header and payload to be received and demodulated according to the packet type. Enhanced Data Rate can be enabled only for ACL-U, eSCO-S logical transports and cannot be enabled for ACL-C, SCO-S, and the broadcast logical transports.

Acronyms & Abbreviations

1 LIST OF ACRONYMS AND ABBREVIATIONS

Acronym or abbreviation	Writing out in full	Which means
8DPSK	8 phase Differential Phase Shift Keying	3 Mbps modulation type used by Enhanced Data rate
π/4 DQPSK	pi/4 Rotated Differential Quaternary Phase Shift Keying	2Mbps modulation type used by Enhanced Data Rate
A		
ACK	Acknowledge	
ACL	Asynchronous Connection-oriented [logical transport]	Reliable or time-bounded, bi-directional, point-to-point.
ACL-C	ACL Control [logical link] (LMP)	
ACL-U	ACL User [logical link] (L2CAP)	
ACO	Authenticated Ciphering Offset	
AFH	Adaptive Frequency Hopping	
AG	Audio Gateway	
AHS	Adapted Hop Sequence	
AR_ADDR	Access Request Address	
ARQ	Automatic Repeat reQuest	
ASB	Active Slave Broadcast [logical transport]	Unreliable, uni-directional broadcast to any devices synchronized with the physical channel.
ASB-U	ASB User [logical link] (L2CAP)	
B		
BB	BaseBand	
BCH	Bose, Chaudhuri & Hocquenghem	Type of code The persons who discovered these codes in 1959 (H) and 1960 (B&C)
BD_ADDR	Bluetooth Device Address	
BER	Bit Error Rate	
BT	Bandwidth Time	
BT	Bluetooth	
C		
CAC	Channel Access Code	

Table 1.1: Acronyms and Abbreviations.

Acronym or abbreviation	Writing out in full	Which means
CC	Call Control	
CL	Connectionless	
CODEC	COder DECoder	
COF	Ciphering Offset	
CRC	Cyclic Redundancy Check	
CVSD	Continuous Variable Slope Delta Modulation	
D		
DAC	Device Access Code	
DCE	Data Communication Equipment	
DCE	Data Circuit-Terminating Equipment	In serial communications, DCE refers to a device between the communication endpoints whose sole task is to facilitate the communications process; typically a modem
DCI	Default Check Initialization	
DEVM	Differential Error Vector Magnitude	Measure of modulation error used for Enhanced Data Rate transmitter testing
DH	Data-High Rate	Data packet type for high rate data
DIAC	Dedicated Inquiry Access Code	
DM	Data - Medium Rate	Data packet type for medium rate data
DPSK	Differential Phase Shift Keying	Generic description of Enhanced Data Rate modulation
DQPSK	Differential Quarternary Phase Shift Keying	Modulation type used by Enhanced Data Rate
DTE	Data Terminal Equipment	In serial communications, DTE refers to a device at the endpoint of the communications path; typically a computer or terminal.
DTMF	Dual Tone Multiple Frequency	
DUT	Device Under Test	
DV	Data Voice	Data packet type for data and voice

Table 1.1: Acronyms and Abbreviations.

Acronym or abbreviation	Writing out in full	Which means
E		
EDR	Enhanced Data Rate	
EIRP	Effective Isotropic Radiated Power	Equivalent power that an isotropic antenna must transmit to provide the same field power density
eSCO	Extended Synchronous Connection Oriented [logical transport]	Bi-directional, symmetric or asymmetric, point-to-point, general regular data, limited retransmission.
eSCO-S	Stream eSCO (unframed)	used to support isochronous data delivered in a stream without framing
ETSI	European Telecommunications Standards Institute	
EUT	Equipment Under Test	
F		
FCC	Federal Communications Commission	
FEC	Forward Error Correction code	
FH	Frequency Hopping	
FHS	Frequency Hop Synchronization	
FIFO	First In First Out	
FM	Frequency Modulation	Modulation Type
FW	Firmware	
G		
GFSK	Gaussian Frequency Shift Keying	
GIAC	General Inquiry Access Code	
GM	Group Management	
H		
HCI	Host Controller Interface	
HEC	Header-Error-Check	
HID	Human Interface Device	
HV	High quality Voice	e.g. HV1 packet
HW	Hardware	

Table 1.1: Acronyms and Abbreviations.

Acronym or abbreviation	Writing out in full	Which means
I		
IAC	Inquiry Access Code	
IEEE	Institute of Electronic and Electrical Engineering	
IETF	Internet Engineering Task Force	
IP	Internet Protocol	
IrDA	Infra-red Data Association	
IrMC	Ir Mobile Communications	
ISDN	Integrated Services Digital Net-works	
ISM	Industrial, Scientific, Medical	
IUT	Implementation Under Test	
L		
L2CAP	Logical Link Control and Adaptation Protocol	
LAP	Lower Address Part	
LC	Link Controller	Link Controller (or baseband) part of the Bluetooth protocol stack. Low level Baseband protocol handler
LC	Link Control [logical link]	The control logical links LC and ACL-C are used at the link control level and link manager level, respectively.
LCP	Link Control Protocol	
LCSS	Link Controller Service Signalling	
LFSR	Linear Feedback Shift Register	
LLID	Logical Link Identifier	
LM	Link Manager	
LMP	Link Manager Protocol	For LM peer to peer communica-tion
LSB	Least Significant Bit	
LT_ADDR	Logical Transport ADDRess	
M		
M	Master or Mandatory	

Table 1.1: Acronyms and Abbreviations.

Acronym or abbreviation	Writing out in full	Which means
MAC	Medium Access Control	
MAPI	Messaging Application Procedure Interface	
Mbps	Million (Mega) bits per second	
MMI	Man Machine Interface	
MS	Mobile Station	
MS	Multiplexing sublayer	
MSB	Most Significant Bit	
MSC	Message Sequence Chart	
MTU	Maximum Transmission Unit	
N		
NAK	Negative Acknowledge	
NAP	Non-significant Address Part	
O		
O	Optional	
OBEX	OBject EXchange protocol	
OCF	OpCode Command Field	
OGF	OpCode Group Field	
P		
PCM	Pulse Coded Modulation	
PCMCIA	Personal Computer Memory Card International Association	
PDU	Protocol Data Unit	a message
PIN	Personal Identification Number	
PM_ADDR	Parked Member Address	
PN	Pseudo-random Noise	
PPM	Part Per Million	
PPP	Point-to-Point Protocol	
PRBS	Pseudo Random Bit Sequence	
PRNG	Pseudo Random Noise Generation	

Table 1.1: Acronyms and Abbreviations.

Acronym or abbreviation	Writing out in full	Which means
PSB	Parked Slave Broadcast [logical transport]	Unreliable, uni-directional broadcast to all piconet devices.
PSB-C	PSB Control [logical link] (LMP)	
PSB-U	PSB User [logical link] (L2CAP)	
PSK	Phase Shift Keying	Class of modulation types
PSTN	Public Switched Telephone Network	
ptt	Packet Type Table	The ptt parameter is used to select the logical transport types via LMP.
Q		
QoS	Quality of Service	
QPSK	Quarternary Phase Shift Keying	Modulation type used by Enhanced Data Rate
R		
RAND	Random number	
RF	Radio Frequency	
RFC	Request For Comments	
RFCMode	Retransmission and Flow Control Mode	
RFCOMM		Serial cable emulation protocol based on ETSI TS 07.10
RMS	Root Mean Square	
RSSI	Received Signal Strength Indication	
RX	Receive	
S		
S	Slave	
SAP	Service Access Points	
SAR	Segmentation and Reassembly	
SCO	Synchronous Connection-Oriented [logical transport]	Bi-directional, symmetric, point-to-point, AV channels.
SCO-S	Stream SCO (unframed)	
SCO-S	Synchronous logical link	used to support isochronous data delivered in a stream without framing

Table 1.1: Acronyms and Abbreviations.

Acronym or abbreviation	Writing out in full	Which means
SD	Service Discovery	
SDDB	Service Discovery Database	
SDP	Service Discovery Protocol	
SDU	Service Data Unit	
SEQN	Sequential Numbering scheme	
SRES	Signed Response	
SS	Supplementary Services	
SSI	Signal Strength Indication	
SUT	System Under Test	
SW	Software	
T		
TBD	To Be Defined	
TC	Test Control	Test Control layer for the test interface
TCI	Test Control Interface	
TCP/IP	Transport Control Protocol/Internet Protocol	
TCS	Telephony Control protocol Specification	
TDD	Time-Division Duplex	
TX	Transmit	
U		
UAP	Upper Address Part	
UART	Universal Asynchronous receiver Transmitter	
UI	User Interface	
UI	Unnumbered Information	
ULAP	Upper and Lower Address Parts	
USB	Universal Serial Bus	
UUID	Universal Unique Identifier	
W		
WAP	Wireless Application Protocol	
WUG	Wireless User Group	

Table 1.1: Acronyms and Abbreviations.

2 ABBREVIATIONS OF THE SPECIFICATION NAMES

Acronym or abbreviation	Writing out in full	Placement in Specification
A2DP	Advanced Audio Distribution Profile Specification	vol 10 part C
AVCTP	A/V Control Transport Protocol Specification	vol 10 part F
AVDTP	A/V Distribution Transport Profile Specification	vol 10 part A
AVRCP	A/V Remote Control Profile Specification	vol 10 part G
BB	Baseband Specification	vol 2 part B
BIP	Basic Imaging Profile	vol 8 part E
BNEP	Bluetooth Network Encapsulation Protocol Specification	vol 6 part A
BPP	Basic Printing Profile Specification	vol 8 part F
CIP	Common ISDN Access Profile Specification	vol 12 part A
CTP	Cordless Telephony Profile Specification	vol 9 part B
DUN	Dial-Up Networking Profile Specification	vol 7 part C
EDR	Enhanced Data Rate	vol 2 part A-C, H
ESDP / UPNP	Extended Service Discovery Profile	vol 6 part D
FAX	Fax Profile Specification	vol 7 part D
FTP	File Transfer Profile Specification	vol 8 part C
GAP	Generic Access Profile Specification	vol 3 part C
GAVDP	Generic A/V Distribution Profile Specification	vol 10 part B
GOEP	Generic Object Exchange Profile Specification	vol 8 part A
HCI (1)	Host Controller Interface Functional Specification	vol 2 part E
HCI (2)	Host Controller Interface Transport Layers Specification	vol 4 part A-C
HCRP	Hardcopy Cable Replacement Profile Specification	vol 11 part B
HFP	Hands-Free Profile Specification	vol 7 part E
HID	Human Interface Device Profile Specification	vol 11 part A
HSP	Headset Profile Specification	vol 7 part F
ICP	Intercom Profile Specification	vol 9 part C
L2CAP	Logical Link Control and Adaptation Protocol Specification	vol 3 part A

Table 2.1: Abbreviations of the specification names.

Acronym or abbreviation	Writing out in full	Placement in Specification
LAP	LAN Access Profile Specification	deprecated
LMP	Link Manager Protocol Specification	vol 2 part C
MSC	Message Sequence Charts	vol 2 part F
OPP	Object Push Profile Specification	vol 8 part B
PAN	Personal Area Networking Profile Specification	vol 6 part B
RF	Radio Specification	vol 2 part A
RFCOMM	RFCOMM with TS 07.10	vol 7 part A
SAP	SIM Access Profile Specification	vol 12 part C
SDAP	Service Discovery Application Profile Specification	vol 5 part B
SDP (1)	Service Discovery Protocol Specification (server)	vol 3 part B
SDP (2)	Service Discovery Protocol Specification (client)	vol 5 part A
SPP	Serial Port Profile Specification	vol 7 part B
Synch	Synchronization Profile Specification	vol 8 part D
TCI	Test Control Interface	vol 3 part D, section 2
TCP	Telephony Control Protocol Specification	vol 9 part A
UDI	Unrestricted Digital Information Profile Specification	vol 12 part B

Table 2.1: Abbreviations of the specification names.

Architecture & Terminology Overview
Part C

Core Specification Change History

CONTENTS

1 CHANGES FROM V1.1 TO V1.2

1.1 NEW FEATURES

Several new features are introduced in the Bluetooth Core Specification 1.2. The major areas of improvement are:

- Architectural overview
- Faster connection
- Adaptive frequency hopping
- Extended SCO links
- Enhanced error detection and flow control
- Enhanced synchronization capability
- Enhanced flow specification

The feature descriptions are incorporated into the existing text in different core parts, described in vol2 and vol3.

1.2 STRUCTURE CHANGES

The Bluetooth Core Specification 1.2 has been significantly restructured for better consistency and readability. The most important structure changes have been performed in Baseband, LMP, HCI and L2CAP. The text in these sections has been rearranged to provide:

- Presentation of the information in a more logical progression
- Removal of redundant text and requirements
- Consolidation of baseband related requirements (for example, the *Baseband Timers* and *Bluetooth Audio* sections into the Baseband Specification)
- Alignment of the specification with the new architecture and terminology presented in the Architecture Overview (see Part A, IEEE Language)

In addition, new text and requirements have been added for the new features as well as many changes throughout the specification to update the text to use IEEE language (see Part E, IEEE Language).

1.3 DEPRECATED SPECIFICATIONS

As the Bluetooth Specification continues to evolve, some features, protocols, and profiles are replaced with new ways of performing the same function. Often these changes reflect the evolution of the communications industry. Some of the changes merely reflect an evolved understanding of the Bluetooth environment itself.

Those functions no longer recommended are being deprecated. The term *deprecation* does not mean that they are no longer allowed, but that they are no longer recommended as the best way of performing a given function.

Specifications that have been deprecated will not be included in the updated Bluetooth Specifications. Deprecation also discontinues all further maintenance of specific associated Test Specifications, PICS and TCRL-records. After deprecation, qualification remains enabled, however, towards a requirement and document set that no longer is maintained, but the earlier versions are still available and may be used according to the rules set forth in the Qualification Policy (PRD).

1.4 DEPRECATED FEATURES

Features deprecated in version 1.2 are:

- The use of Unit Keys for security
- Optional Paging schemes
- 23 channel hopping sequence
- Page scan period mode

1.5 CHANGES IN WORDING

Two general classes of changes to the wording of the Bluetooth Specification have been done for version 1.2. They are a formalization of the language by using conventions established by the Institute of Electrical and Electronic Engineers (IEEE), and a regularization of Bluetooth wireless technology-specific terms. Many portions of the version 1.1 specification use imprecise or inaccurate terms to describe attributes of the protocol. A more accurate terminology described in Part E has been introduced into the version 1.2 specification and will be applied in future versions.

1.6 NOMENCLATURE CHANGES

The nomenclature in Bluetooth 1.2 has also been updated due to new concepts that are introduced together with the new features and the new architecture (see [Part A] Section 1.2 on page 15).

2 CHANGES FROM V1.2 TO V2.0 + EDR

2.1 NEW FEATURES

The Bluetooth Core Specification version 2.0 + EDR introduces Enhanced Data Rate (EDR). EDR provides a set of additional packet types that use the new 2 Mbps and 3 Mbps modes.

In addition to EDR a set of errata provided in ESR for Core and HCI Transports C1.2V10r00 has been incorporated into this version and revised to include changes caused by the addition of EDR.

These additions are incorporated into the existing text in different core parts described in Volumes 2 and 3.

2.2 DEPRECATED FEATURES

The only feature deprecated in version 2.0 + EDR is the Page Scan Period Mode and associated commands (based on Erratum 694 which is also included in ESR for Core and HCI Transports C1.2V10r00.

Mixing of Specification Versions

CONTENTS

1 MIXING OF SPECIFICATION VERSIONS

This part describes how different versions of the Core System Packages can be mixed in Bluetooth implementations. The Core System Packages consist of a Controller Package (see volume 2) and a Host Package (see volume 3).

In order to describe how these packages can be mixed, one needs to distinguish between four categories of features specified in the different specification versions. The four categories are:

Category	Description
Type 1	Feature that exists below HCI and cannot be configured via HCI
Type 2	Feature that exists below HCI and can be configured/enabled via HCI
Type 3	Feature that exists below and above HCI and requires HCI command/ events to function
Type 4	Feature that exists only above HCI

The outcome of mixing different core system packages are derived from the feature definitions in the table above:

- If an implementation contains features of type 1 or type 4, these features can function with any combination of Host Package and- Controller Package versions.
- If an implementation contains features of type 2, these features can only be used under a default condition if a Host Package of an older version is mixed with a Controller Package of this version.
- In order to fully use the feature under all conditions, the Host Package and Controller Package must be of the same version.
- If an implementation contains features of type 3, these features can only function with a Host Package and a Controller Package both in the same version.

1.1 FEATURES AND THEIR TYPES

The following table lists the features and their types.

Feature	Version	Type
Basic AFH operation	V1.2	1
Enhanced inquiry	V1.2	1
Configuration of AFH (setting channels and enabling/disabling channel assessment)	V1.2	2
Enhanced synchronization capability	V1.2	2
Interlaced inquiry scan	V1.2	2
Interlaced page scan	V1.2	2
Broadcast encryption	V1.2	2
Enhanced flow specification and flush time-out	V1.2	3
Extended SCO links	V1.2	3
Inquiry Result with RSSI	V1.2	3
L2CAP flow and error control	V1.2	4
2Mbps EDR	V2.0 + EDR	2
3Mbps EDR	V2.0 + EDR	2
3 slot packets in EDR	V2.0 + EDR	2
5 slot packets in EDR	V2.0 + EDR	2
2 Mbps eSCO	V2.0 + EDR	2*
3 Mbps eSCO	V2.0 + EDR	2*
3 slot packets for EDR eSCO	V2.0 + EDR	2*

The EDR eSCO options are marked as 2* because eSCO requires profile support, but if a product includes the eSCO option from V1.2, then EDR eSCO will be supported without any new support above HCI.

Architecture & Terminology Overview
Part E

IEEE Language

CONTENTS

1 USE OF IEEE LANGUAGE

One of the purposes of this terminology is to make it easy for the reader to identify text that diatribes requirements as opposed to background information. The general term for text that describes attributes that are required for proper implementation of Bluetooth wireless technology is normative. The general term for language that provides background and context for normative text is informative. These terms are used in various sections to clarify implementation requirements.

Many portions of the Bluetooth Specification use imprecise or inaccurate terms to describe attributes of the protocol. This subsection describes the correct usage of key terms that indicate degree of requirements for processes and data structures. The information here was derived from the Institute of Electrical and Electronic Engineers (IEEE) Style Guide, see http://standards.ieee.org/guides/style/.

The following list is a summary of the terms to be discussed in more detail below:

shall	is required to – used to define requirements
must	is a natural consequence of -- used only to describe unavoidable situations
will	it is true that -- only used in statements of fact
should	is recommended that – used to indicate that among several possibilities one is recommended as particularly suitable, but not required
may	is permitted to – used to allow options
can	is able to – used to relate statements in a causal fashion
is	is defined as – used to further explain elements that are previously required or allowed
note	<informational text ONLY>

For clarity of the definition of those terms, the following sections document why and how they are used. For these sections only, the IEEE terms are italicized to indicate their use as a noun. Uses and examples of the use of the terms in this section are underlined.

1.1 SHALL

The word *shall* is used to indicate mandatory requirements that shall be followed in order to conform to the specification and from which no deviation is permitted.

There is a strong implication that the presence of the word *shall* indicates a testable requirement. All testable requirements shall be reflected in the Protocol Implementation Conformance Statement (PICS). In turn, all PICS indicators should be reflected in the Test Cases (TCs) either directly or indirectly.

A direct reference is a specific test for the attribute cited in the text. For example, a minimum value for a given parameter may be an entry in the TCs. Indirect test coverage may be appropriate if the existence of the attribute is requisite for passing a higher level test.

1.2 MUST

The word *must* shall not be used when stating mandatory requirements. *Must* is used only to describe unavoidable situations and is seldom appropriate for the text of a Specification.

An example of an appropriate use of the term *must* is: “the Bluetooth radios must be in range of each other to communicate”.

1.3 WILL

The use of the word *will* shall not be used when stating mandatory requirements. The term *will* is only used in statements of fact. As with the term *must*, *will* is not generally applicable to the description of a protocol. An example of appropriate use of will is: “when power is removed from the radio, it can be assumed that communications will fail”

1.4 SHOULD

Should equals *is recommended that*. The word *should* is used to indicate that among several possibilities one is recommended as particularly suitable without mentioning or excluding others. Alternatively it may indicate that a certain course of action is preferred but not necessarily required. Finally, in the negative form, it indicates a certain course of action is deprecated but not prohibited.

In the Bluetooth Specification the term designates an optional attribute that may require an entry in the PICS.

Explicit specification of alternatives should be done when using *should*.

1.5 MAY

The word *may* is used to indicate a course of action permissible within the limits of the specification. The term *may* equals *is permitted*. This is generally used when there is a single, optional attribute described, but multiple alternatives may be cited.

The use of *may* implies an optional condition in the PICS and therefore may need to be reflected in the corresponding test cases.

1.6 CAN

The word *can* is used for statements of possibility and capability, whether material, physical, or causal. The term *can* equals *is able to*.

The term *can* shall be used only in informative text. It describes capabilities by virtue of the rules established by normative text.

Specification of the Bluetooth System

Wireless connections made easy

Core System Package

[Controller volume]

Covered Core Package version:
2.0 + EDR
Current Master TOC issued:
4 November 2004

Bluetooth®

Revision History

The Revision History is shown in the “Appendix” on page 51[vol. 0].

Contributors

The persons who contributed to this specification are listed in the “Appendix” on page 51[vol. 0].

Web Site

This specification can also be found on the official Bluetooth web site:
http://www.bluetooth.com

Disclaimer and Copyright Notice

Part A
RADIO SPECIFICATION

Contents

Part B
BASEBAND SPECIFICATION

Part E

HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

Part H
SECURITY SPECIFICATION

Core System Package [Controller volume]
Part A

RADIO SPECIFICATION

CONTENTS

1 SCOPE

Bluetooth devices operate in the unlicensed 2.4 GHz ISM (Industrial Scientific Medical) band. A frequency hop transceiver is applied to combat interference and fading.

Two modulation modes are defined. A mandatory mode, called Basic Rate, uses a shaped, binary FM modulation to minimize transceiver complexity. An optional mode, called Enhanced Data Rate, uses PSK modulation and has two variants: π/4-DQPSK and 8DPSK. The symbol rate for all modulation schemes is 1 Ms/s. The gross air data rate is 1 Mbps for Basic Rate, 2 Mbps for Enhanced Data Rate using π/4-DQPSK and 3 Mbps for Enhanced Data Rate using 8DPSK.

For full duplex transmission, a Time Division Duplex (TDD) scheme is used in both modes. This specification defines the requirements for a Bluetooth radio for the Basic Rate and Enhanced Data Rate modes.

Requirements are defined for two reasons:

- Provide compatibility between radios used in the system
- Define the quality of the system

The Bluetooth radio shall fulfil the stated requirements under the operating conditions specified in Appendix A and Appendix B. The radio parameters shall be measured according to the methods described in the RF Test Specification.

This specification is based on the established regulations for Europe, Japan and North America. The standard documents listed below are only for information, and are subject to change or revision at any time.

Europe:

Approval Standards: European Telecommunications Standards Institute, ETSI
Documents: EN 300 328, ETS 300-826
Approval Authority: National Type Approval Authorities

Japan:

Approval Standards: Association of Radio Industries and Businesses, ARIB
Documents: ARIB STD-T66
Approval Authority: Ministry of Post and Telecommunications, MPT.

North America:

Approval Standards: Federal Communications Commission, FCC, USA
Documents: CFR47, Part 15, Sections 15.205, 15.209, 15.247 and 15.249

Approval Standards: Industry Canada, IC, Canada
Documents: GL36

Approval Authority: FCC (USA), Industry Canada (Canada)

2 FREQUENCY BANDS AND CHANNEL ARRANGEMENT

The Bluetooth system operates in the 2.4 GHz ISM band. This frequency band is 2400 - 2483.5 MHz.

Regulatory Range	RF Channels
2.400-2.4835 GHz	f=2402+k MHz, k=0,…,78

Table 2.1: Operating frequency bands

RF channels are spaced 1 MHz and are ordered in channel number k as shown in Table 2.1. In order to comply with out-of-band regulations in each country, a guard band is used at the lower and upper band edge.

Lower Guard Band	Upper Guard Band
2 MHz	3.5 MHz

Table 2.2: Guard Bands

3 TRANSMITTER CHARACTERISTICS

The requirements stated in this section are given as power levels at the antenna connector of the Bluetooth device. If the device does not have a connector, a reference antenna with 0 dBi gain is assumed.

Due to difficulty in measurement accuracy in radiated measurements, systems with an integral antenna should provide a temporary antenna connector during type approval.

If transmitting antennas of directional gain greater than 0 dBi are used, the applicable paragraphs in EN 300 328, EN 301 489-17and FCC part 15 shall be compensated for.

The device is classified into three power classes.

Power Class	Maximum Output Power (Pmax)	Nominal Output Power	Minimum Output Power[1]	Power Control
1	100 mW (20 dBm)	N/A	1 mW (0 dBm)	Pmin<+4 dBm to Pmax Optional: Pmin[2] to Pmax
2	2.5 mW (4 dBm)	1 mW (0 dBm)	0.25 mW (-6 dBm)	Optional: Pmin[2)] to Pmax
3	1 mW (0 dBm)	N/A	N/A	Optional: Pmin[2)] to Pmax

Table 3.1: Power classes

1. Minimum output power at maximum power setting.
2. The lower power limit Pmin<-30dBm is suggested but is not mandatory, and may be chosen according to application needs.

Power class 1 device shall implement power control. The power control is used for limiting the transmitted power over +4 dBm. Power control capability under +4 dBm is optional and could be used for optimizing the power consumption and overall interference level. The power steps shall form a monotonic sequence, with a maximum step size of 8 dB and a minimum step size of 2 dB. A class 1 device with a maximum transmit power of +20 dBm shall be able to control its transmit power down to 4 dBm or less.

Devices with power control capability optimizes the output power in a physical link with LMP commands (see Link Manager Protocol). It is done by measuring RSSI and reporting back if the transmission power shall be increased or decreased if possible.

In a connection, the output power shall not exceed the maximum output power of power class 2 for transmitting packets if the receiving device does not support the necessary messaging for sending the power control messages, see

Link Manager Protocol Section 4.1.3 on page 235. In this case, the transmitting device shall comply with the rules of a class 2 or class 3 device.

If a class 1 device is paging or inquiring very close to another device, the input power can be larger than the requirement in Section 4.1.5 on page 43. This can cause the receiving device to fail to respond. It may therefore be useful to page at Class 2 or 3 power in addition to paging at power class 1.

Devices shall not exceed the maximum allowed transmit power levels set by controlling regulatory bodies. The maximum allowed transmit power level may depend upon the modulation mode.

3.1 BASIC RATE

3.1.1 Modulation Characteristics

The Modulation is GFSK (Gaussian Frequency Shift Keying) with a bandwidth-bit period product BT=0.5. The Modulation index shall be between 0.28 and 0.35. A binary one shall be represented by a positive frequency deviation, and a binary zero shall be represented by a negative frequency deviation. The symbol timing shall be less than ±20 ppm.

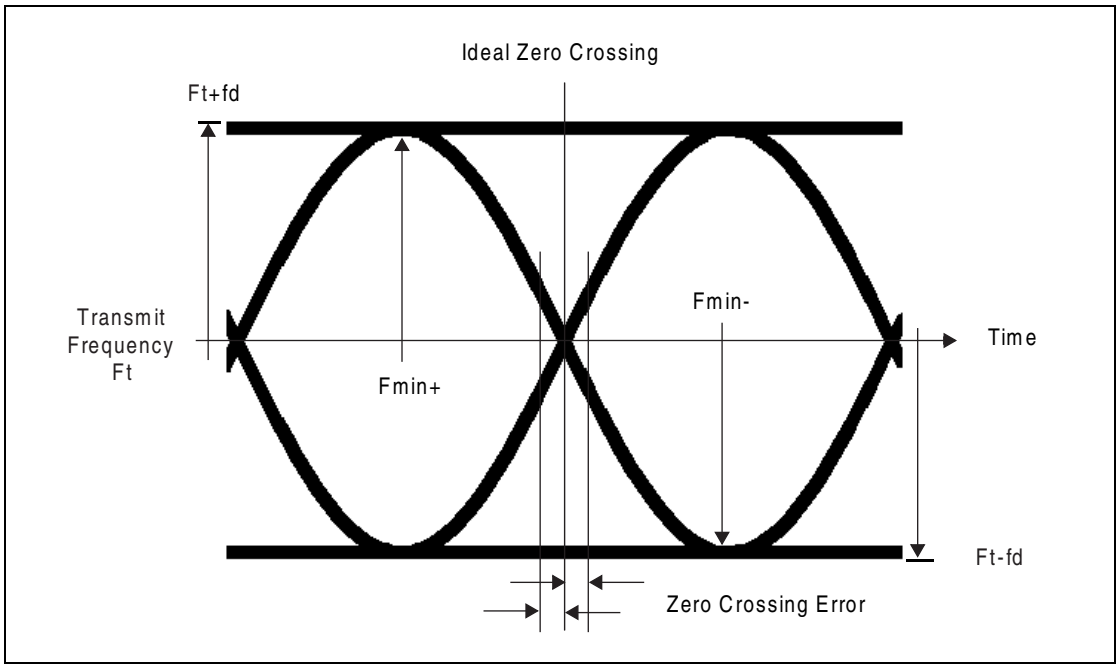


Figure 3.1: GFSK parameters definition.

For each transmission, the minimum frequency deviation, Fmin = min{|Fmin+|, Fmin-}, which corresponds to 1010 sequence shall be no smaller than ±80% of the frequency deviation (fd) with respect to the transmit frequency Ft, which corresponds to a 00001111 sequence.

In addition, the minimum frequency deviation shall never be smaller than 115 kHz. The data transmitted has a symbol rate of 1 Ms/s.

The zero crossing error is the time difference between the ideal symbol period and the measured crossing time. This shall be less than ± 1/8 of a symbol period.

3.1.2 Spurious Emissions

In-band spurious emissions shall be measured with a frequency hopping radio transmitting on one RF channel and receiving on a second RF channel; this means that the synthesizer may change RF channels between reception and transmission, but always returns to the same transmit RF channel. There will be no reference in this document to out of ISM band spurious emissions; the equipment manufacturer is responsible for compliance in the intended country of use.

3.1.2.1 In-band spurious emission

Within the ISM band the transmitter shall pass a spectrum mask, given in Table 3.2. The spectrum shall comply with the 20dB bandwidth definition in FCC part 15.247 and shall be measured accordingly. In addition to the FCC requirement an adjacent channel power on adjacent channels with a difference in RF channel number of two or greater is defined. This adjacent channel power is defined as the sum of the measured power in a 1 MHz RF channel. The transmitted power shall be measured in a 100 kHz bandwidth using maximum hold. The device shall transmit on RF channel M and the adjacent channel power shall be measured on RF channel number N. The transmitter shall transmit a pseudo random data pattern in the payload throughout the test.

Frequency offset	Transmit Power
± 500 kHz	-20 dBc
2MHz (|M-N| = 2)	-20 dBm
3MHz or greater (|M-N| ≥ 3)	-40 dBm

Table 3.2: Transmit Spectrum mask.

Note: If the output power is less than 0dBm then, wherever appropriate, the FCC's 20 dB relative requirement overrules the absolute adjacent channel power requirement stated in the above table.

Exceptions are allowed in up to three bands of 1 MHz width centered on a frequency which is an integer multiple of 1 MHz. They shall comply with an absolute value of –20 dBm.

3.1.3 Radio Frequency Tolerance

The transmitted initial center frequency shall be within ±75 kHz from F_C. The initial frequency accuracy is defined as being the frequency accuracy before any packet information is transmitted. Note that the frequency drift requirement is not included in the ±75 kHz.

The limits on the transmitter center frequency drift within a packet are specified in Table 3.3. The different packets are defined in the Baseband Specification.

Duration of Packet	Frequency Drift
Max length one slot packet	±25 kHz
Max length three slot packet	±40 kHz
Max length five slot packet	±40 kHz
Maximum drift rate[1]	400 Hz/μs

Table 3.3: Maximum allowable frequency drifts in a packet.

1. The maximum drift rate is allowed anywhere in a packet.

3.2 ENHANCED DATA RATE

A key characteristic of the Enhanced Data Rate mode is that the modulation scheme is changed within the packet. The access code and packet header, as defined in Table 6.1 in the Baseband Specification, are transmitted with the Basic Rate 1 Mbps GFSK modulation scheme, whereas the subsequent synchronization sequence, payload, and trailer sequence are transmitted using the Enhanced Data Rate PSK modulation scheme.

3.2.1 Modulation Characteristics

During access code and packet header transmission the Basic Rate GFSK modulation scheme shall be used. During the transmission of the synchronization sequence, payload, and trailer sequence a PSK type of modulation with a data rate of 2 Mbps or optionally 3 Mbps shall be used. The following subsections specify the PSK modulation for this transmission.

3.2.1.1 Modulation Method Overview

The PSK modulation format defined for the 2 Mbps transmission shall be π/4 rotated differential encoded quaternary phase shift keying (π/4-DQPSK).

The PSK modulation format defined for the 3 Mbps transmission shall be differential encoded 8-ary phase shift keying (8DPSK).

The modulation shall employ square-root raised cosine pulse shaping to generate the equivalent lowpass information-bearing signal v(t). The output of the transmitter shall be a bandpass signal that can be represented as

$$S(t) = Re\left[v(t)e^{j2\pi F_c t}\right] \quad \text{(EQ 1)}$$

with F_c denoting the carrier frequency.

3.2.1.2 Differential Phase Encoding

For the M-ary modulation, the binary data stream $\{b_n\}$, n=1,2,3, …N, shall be mapped onto a corresponding sequence {Sk}, k=1,2, …$N/\log_2(M)$ of complex valued signal points. M=4 applies for 2 Mbps and M=8 applies for 3 Mbps. Gray coding shall be applied as shown in Table 3.4 and Table 3.5. In the event that the length of the binary data stream N is not an integer multiple of log2(M), the last symbol of the sequence $\{S_k\}$ shall be formed by appending data zeros to the appropriate length. The signal points S_k shall be defined by:

$$S_k = S_{k-1}e^{j\varphi_k} \qquad k = 1, 2, ..N/\log_2(M) \quad \text{(EQ 2)}$$

$$S_0 = e^{j\phi} \qquad \text{with } \phi \in [0, 2\pi) \quad \text{(EQ 3)}$$

The relationship between the binary input b_k and the phase ϕ_k shall be as defined in Table 3.4 for the 2 Mbps transmission and in Table 3.5 for the 3 Mbps transmission.

b_{2k-1}	b_{2k}	φ_k
0	0	$\pi/4$
0	1	$3\pi/4$
1	1	$-3\pi/4$
1	0	$-\pi/4$

Table 3.4: $\pi/4$-DQPSK mapping.

b_{3k-2}	b_{3k-1}	b_{3k}	φ_k
0	0	0	0
0	0	1	π/4
0	1	1	π/2
0	1	0	3π/4
1	1	0	π
1	1	1	-3π/4
1	0	1	-π/2
1	0	0	-π/4

Table 3.5: 8DPSK mapping.

3.2.1.3 Pulse Shaping

The lowpass equivalent information-bearing signal v(t) shall be generated according to

$$v(t) = \sum_k S_k p(t - kT) \qquad \text{(EQ 4)}$$

in which the symbol period T shall be 1μs.

The frequency spectrum P(f), which corresponds to the square-root raised cosine pulse p(t) of the pulse shaping filter is:

$$|P(f)| = \begin{cases} 1 & 0 \le |f| \le \frac{1-\beta}{2T} \\ \sqrt{\frac{1}{2}\left(1 - \sin\left(\frac{\pi(2fT-1)}{2\beta}\right)\right)} & \frac{1-\beta}{2T} \le |f| \le \frac{1+\beta}{2T} \\ 0 & \text{elsewhere} \end{cases} \qquad \text{(EQ 5)}$$

The roll off factor β shall be 0.4.

3.2.1.4 Modulation Accuracy

The measurement of modulation accuracy utilizes differential error vector magnitude (DEVM) with tracking of the carrier frequency drift. The definition of DEVM and the derivation of the RMS and peak measures of DEVM are given in 51.

The DEVM shall be measured over the synchronization sequence and payload portions of the packet, but not the trailer symbols. For each modulation method and each measurement carrier frequency, DEVM measurement is made over a total of 200 non-overlapping blocks, where each block contains 50 symbols.

The transmitted packets shall be the longest supported packet type for each modulation method, as defined in Table 6.9 and Table 6.10 in the Baseband part.

The DEVM is measured using a square-root raised cosine filter, with a roll-off of 0.4 and a 3 dB bandwidth of ±500 kHz.

3.2.1.4.1 RMS DEVM

The RMS DEVM for any of the measured blocks shall not exceed 0.20 for π/4-DQPSK and 0.13 for 8DPSK.

3.2.1.4.2 99% DEVM

The 99% DEVM (defined as the DEVM value for which 99% of measured symbols have a lower DEVM) shall not exceed 0.30 for π/4-DQPSK and 0.20 for 8DPSK.

3.2.1.4.3 Peak DEVM

The Peak DEVM shall not exceed 0.35 for π/4-DQPSK and 0.25 for 8DPSK.

3.2.2 Spurious Emissions

In-band spurious emissions shall be measured with a frequency hopping radio transmitting on one RF channel and receiving on a second RF channel; this means that the synthesizer may change RF channels between reception and transmission, but always returns to the same transmit RF channel. There will be no reference in this document to out of ISM band spurious emissions; the equipment manufacturer is responsible for compliance in the intended country of use.

3.2.2.1 In-band Spurious Emission

Within the ISM band the power spectral density of the transmitter shall comply with the following requirements when sending pseudo random data. All power measurements shall use a 100 kHz bandwidth with maximum hold. The power measurements between 1 MHz and 1.5 MHz from the carrier shall be at least 26 dB below the maximum power measurement up to 500 kHz from the carrier. The adjacent channel power for channels at least 2 MHz from the carrier is defined as the sum of the power measurements over a 1 MHz channel and shall not exceed -20 dBm for the second adjacent channels and -40 dBm for the third and subsequent adjacent channels. These requirements shall apply to

the transmitted signal from the start of the guard time to the end of the power down ramp. The spectral mask is illustrated in Figure 3.2.

Exceptions are allowed in up to 3 bands of 1 MHz width centered on a frequency which is an integer multiple of 1 MHz. They shall comply with an absolute value of –20 dBm.

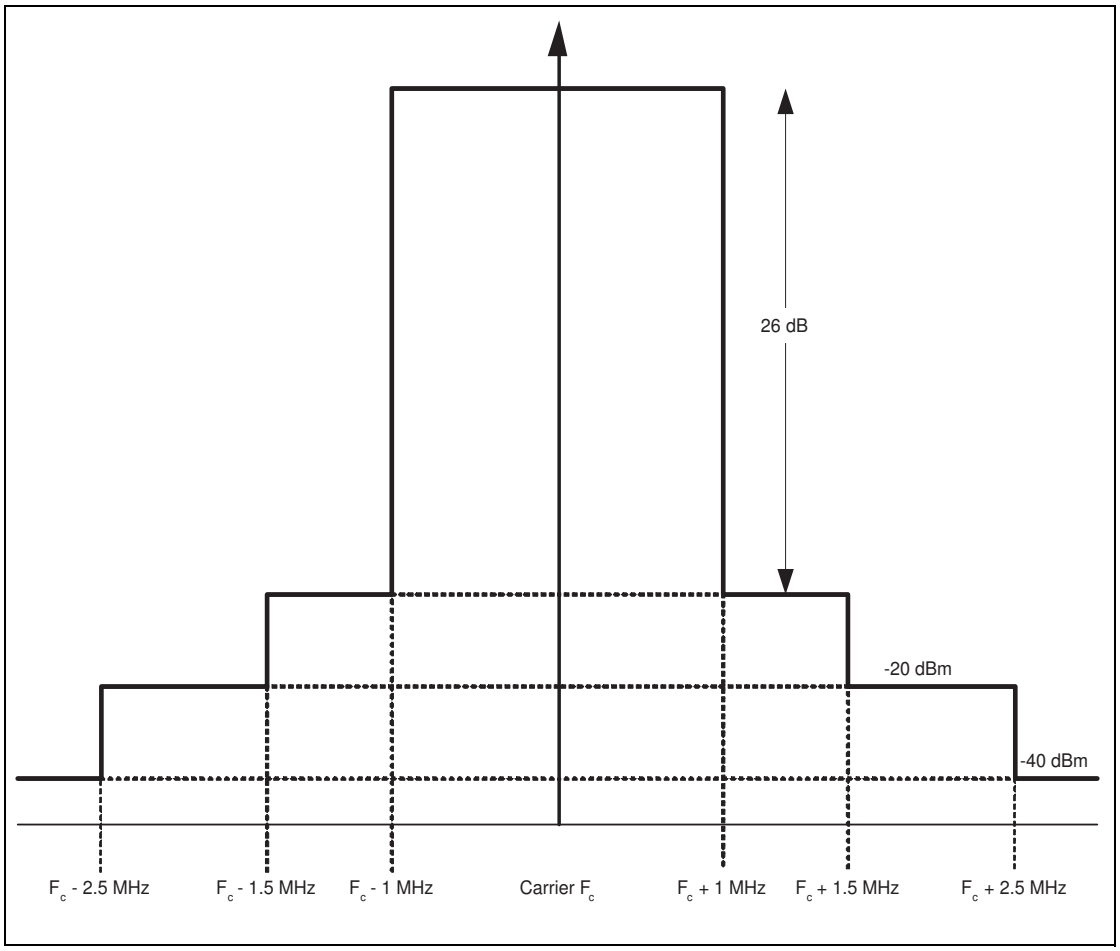


Figure 3.2: Transmitter spectrum mask

3.2.3 Radio Frequency Tolerance

The same carrier frequencies F_c as used for Basic Rate transmissions shall be used for the Enhanced Data Rate transmissions. The transmitted initial center frequency accuracy shall be within ±75 kHz from Fc. The maximum excursion from Fc (frequency offset plus drift) shall not exceed ±75 kHz.

The initial frequency accuracy is defined as being the frequency accuracy before any information is transmitted.

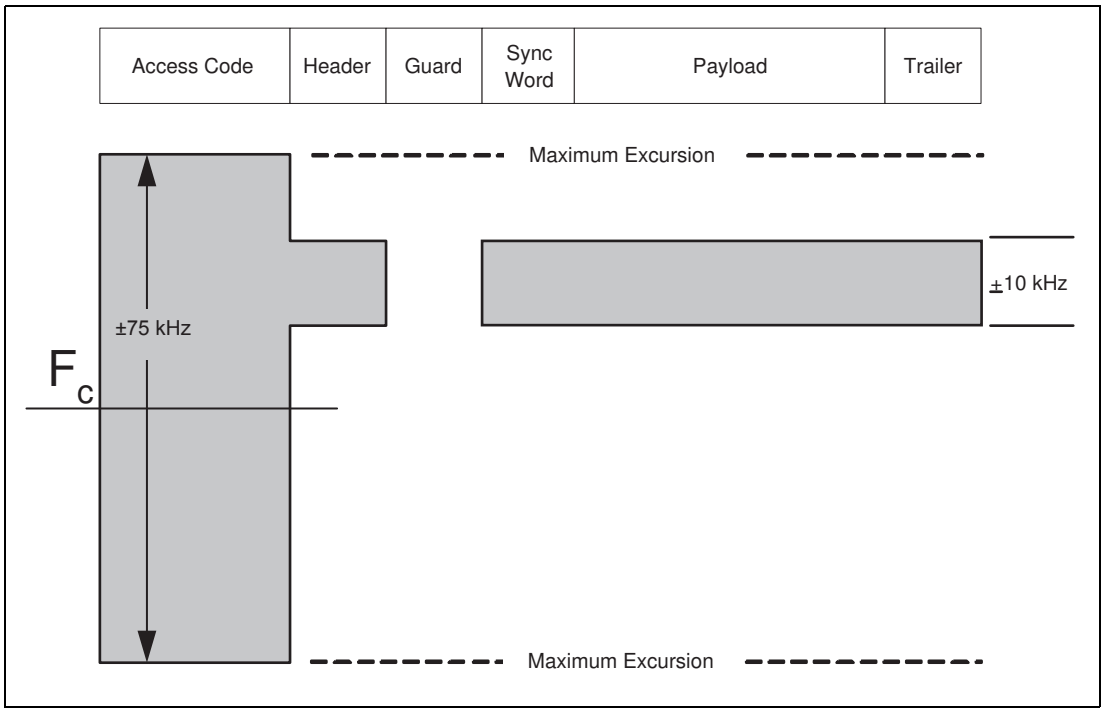


Figure 3.3: Carrier frequency mask

The requirements on accuracy and stability are illustrated by Figure 3.3 for the Enhanced Data Rate packet format defined in 55. The higher frequency accuracy requirement shall start at the first symbol of the header. The maximum drift over the header, synchronization sequence and payload shall be ±10 kHz.

3.2.4 Relative Transmit Power

The requirement on the relative power of the GFSK and PSK portions of the Enhanced Data Rate packet is defined as follows. The average power level during the transmission of access code and header is denoted as P_{GFSK} and the average power level during the transmission of the synchronization sequence and the payload is denoted as P_{DFSK}.The following inequalities shall be satisfied independently for every Enhanced Data Rate packet transmitted:

$(P_{GFSK} - 4\ dB) < P_{DPSK} < (P_{GFSK} + 1\ dB)$

4 RECEIVER CHARACTERISTICS

The receiver characteristics shall be measured using loopback as defined in Section 1, “Test Methodology,” on page 231.

The reference sensitivity level referred to in this chapter is -70 dBm.

4.1 BASIC RATE

4.1.1 Actual Sensitivity Level

The actual sensitivity level is defined as the input level for which a raw bit error rate (BER) of 0.1% is met. The receiver sensitivity shall be below or equal to –70 dBm with any Bluetooth transmitter compliant to the transmitter specification specified in Section 3 on page 31.

4.1.2 Interference Performance

The interference performance on Co-channel and adjacent 1 MHz and 2 MHz shall be measured with the wanted signal 10 dB over the reference sensitivity level. For interference performance on all other RF channels the wanted signal shall be 3 dB over the reference sensitivity level. If the frequency of an interfering signal is outside of the band 2400-2483.5 MHz, the out-of-band blocking specification (see Section 4.1.3 on page 42) shall apply. The interfering signal shall be Bluetooth-modulated (see Section 4.1.7 on page 43). The BER shall be ≤0.1% for all the signal-to-interference ratios listed in Table 4.1:

Frequency of Interference	Ratio
Co-Channel interference, $C/I_{co\text{-}channel}$	11 dB
Adjacent (1 MHz) interference, C/I_{1MHz}	0 dB
Adjacent (2 MHz) interference, C/I_{2MHz}	-30 dB
Adjacent (≥3 MHz) interference, $C/I_{\geq 3MHz}$	-40 dB
Image frequency Interference[1] [2], C/I_{Image}	-9 dB
Adjacent (1 MHz) interference to in-band image frequency, $C/I_{Image\pm 1MHz}$	-20 dB

Table 4.1: Interference performance

1. In-band image frequency
2. If the image frequency ≠ n*1 MHz, then the image reference frequency is defined as the closest n*1 MHz frequency.

   If two adjacent channel specifications from Table 4.1 are applicable to the same channel, the more relaxed specification applies.

These specifications are only to be tested at nominal temperature conditions with a device receiving on one RF channel and transmitting on a second RF channel;

this means that the synthesizer may change RF channels between reception and transmission, but always returns to the same receive RF channel.

RF channels where the requirements are not met are called spurious response RF channels. Five spurious response RF channels are allowed at RF channels with a distance of ≥2 MHz from the wanted signal. On these spurious response RF channels a relaxed interference requirement C/I = -17 dB shall be met.

4.1.3 Out-of-Band Blocking

The out-of-band suppression (or rejection) shall be measured with the wanted signal 3 dB over the reference sensitivity level. The interfering signal shall be a continuous wave signal. The BER shall be ≤ 0.1%. The out-of-band blocking shall fulfil the following requirements:

Interfering Signal Frequency	Interfering Signal Power Level
30 MHz - 2000 MHz	-10 dBm
2000 - 2399 MHz	-27 dBm
2484 – 3000 MHz	-27 dBm
3000 MHz – 12.75 GHz	-10 dBm

Table 4.2: Out-of-band suppression (or rejection) requirements.

24 exceptions are permitted which are dependent upon the given RF channel and are centered at a frequency which is an integer multiple of 1 MHz. For at least 19 of these spurious response frequencies, a reduced interference level of at least -50dBm is allowed in order to achieve the required BER=0.1% performance, whereas for a maximum of 5 of the spurious frequencies the interference level may be assumed arbitrarily lower.

4.1.4 Intermodulation Characteristics

The reference sensitivity performance, BER = 0.1%, shall be met under the following conditions:

- The wanted signal shall be at frequency f_0 with a power level 6 dB over the reference sensitivity level.
- A static sine wave signal shall be at a frequency f_1 with a power level of –39 dBm.
- A Bluetooth modulated signal (see Section 4.1.7 on page 43) shall be at f_2 with a power level of -39 dBm.

Frequencies f0, f1 and f2 shall be chosen such that $f_0=2f_1-f_2$ and $|f_2-f_1|=n*1$ MHz, where n can be 3, 4, or 5. The system shall fulfill at least one of the three alternatives (n=3, 4, or 5).

4.1.5 Maximum Usable Level

The maximum usable input level that the receiver operates at shall be greater than -20 dBm. The BER shall be less than or equal to 0.1% at –20 dBm input power.

4.1.6 Receiver Signal Strength Indicator

If a device supports Receive Signal Strength Indicator (RSSI) the accuracy shall be +/- 6 dBm.

4.1.7 Reference Signal Definition

A Bluetooth modulated interfering signal shall be defined as:

Modulation = GFSK

Modulation index = 0.32±1%

BT= 0.5±1%

Bit Rate = 1 Mbps ±1 ppm

Modulating Data for wanted signal = PRBS9

Modulating Data for interfering signal = PRBS 15

Frequency accuracy better than ±1 ppm.

4.2 ENHANCED DATA RATE

4.2.1 Actual Sensitivity Level

The actual sensitivity level shall be defined as the input level for which a raw bit error rate (BER) of 0.01% is met. The requirement for a Bluetooth π/4-DQPSK and 8DPSK Enhanced Data Rate receiver shall be an actual sensitivity level of –70 dBm or better. The receiver shall achieve the –70 dBm sensitivity level with any Bluetooth transmitter compliant to the Enhanced Data Rate transmitter specification as defined in Section 3.2.

4.2.2 BER Floor Performance

The receiver shall achieve a BER less than 0.001% at 10 dB above the reference sensitivity level.

4.2.3 Interference Performance

The interference performance for co-channel and adjacent 1 MHz and 2 MHz channels shall be measured with the wanted signal 10 dB above the reference sensitivity level. On all other frequencies the wanted signal shall be 3 dB above the reference sensitivity level. The requirements in this section shall only apply if the frequency of the interferer is inside of the band 2400-2483.5 MHz.

The interfering signal for co-channel interference shall be similarly modulated as the desired signal. The interfering signal for other channels shall be equivalent to a nominal Bluetooth Basic Rate GFSK transmitter. The interfering signal shall carry random data.

A BER of 0.1% or better shall be achieved for the signal to interference ratios defined in Table 4.3.

Frequency of Interference	**π/4-DQPSK Ratio**	**8DPSK Ratio**
Co-Channel interference, $C/I_{co\text{-}channel}$	13 dB	21 dB
Adjacent (1 MHz) interference[1], C/I_{1MHz}	0 dB	5 dB
Adjacent (2MHz) interference[1], C/I_{2MHz}	-30 dB	-25 dB
Adjacent (≥3MHz) interference[1]	-40 dB	-33 dB
Image frequency interference[1,2,3], C/I_{Image}	-7 dB	0 dB
Adjacent (1 MHz) interference to in-band image frequency[1,2,3], $C/I_{Image \pm 1MHz}$	-20 dB	-13 dB

Table 4.3: Interference Performance

1. If two adjacent channel specifications from Table 4.3 are applicable to the same channel, the more relaxed specification applies.
2. In-band image frequency.
3. If the image frequency is not equal to n*1 MHz, then the image reference frequency is defined as the closest n*1 MHz frequency.

These specifications are only to be tested at nominal temperature conditions with a receiver hopping on one frequency; this means that the synthesizer may change frequency between receive slot and transmit slot, but always returns to the same receive frequency.

Frequencies where the requirements are not met are called spurious response frequencies. Five spurious response frequencies are allowed at frequencies with a distance of ≥2 MHz from the wanted signal. On these spurious response frequencies a relaxed interference requirement C/I = -15 dB for π/4-DQPSK and C/I = -10 dB for 8DPSK shall be met.

4.2.4 Maximum Usable Level

The maximum usable input level that the receiver operates at shall be greater than -20 dBm. The BER shall be less than or equal to 0.1% at -20 dBm input power.

4.2.5 Out-of-Band and Intermodulation Characteristics

Note: The Basic Rate out-of-band blocking and intermodulation requirements ensure adequate Enhanced Data Rate performance, and therefore there are no specific requirements for Enhanced Data Rate.

4.2.6 Reference Signal Definition

A 2 Mbps Bluetooth signal used as "wanted" or "interfering signal" is defined as:

Modulation = π/4-DQPSK
Symbol Rate = 1 Msym/s ± 1 ppm
Frequency accuracy better than ±1 ppm
Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
RMS Differential Error Vector Magnitude < 5%
Average power over the GFSK and DPSK portions of the packet shall be equal to within +/- 1 dB

A 3 Mbps Bluetooth signal used as "wanted" or "interfering signal" is defined as:

Modulation = 8DPSK
Symbol Rate = 1 Msym/s ± 1 ppm
Frequency accuracy better than ±1 ppm
Modulating Data for wanted signal = PRBS9
Modulating Data for interfering signal = PRBS15
RMS Differential Error Vector Magnitude < 5%
Average power over the GFSK and DPSK portions of the packet shall be equal to within +/- 1 dB

5 APPENDIX A

5.1 NOMINAL TEST CONDITIONS

5.1.1 Nominal temperature

The nominal temperature conditions for tests shall be +15 to +35 ^{o}C. When it is impractical to carry out the test under this condition a note to this effect, stating the ambient temperature, shall be recorded. The actual value during the test shall be recorded in the test report.

5.1.2 Nominal power source

5.1.2.1 Mains voltage

The nominal test voltage for equipment to be connected to the mains shall be the nominal mains voltage. The nominal voltage shall be the declared voltage or any of the declared voltages for which the equipment was designed. The frequency of the test power source corresponding to the AC mains shall be within 2% of the nominal frequency.

5.1.2.2 Lead-acid battery power sources used in vehicles

When radio equipment is intended for operation from the alternator-fed lead-acid battery power sources which are standard in vehicles, then the nominal test voltage shall be 1.1 times the nominal voltage of the battery (6V, 12V, etc.).

5.1.2.3 Other power sources

For operation from other power sources or types of battery (primary or secondary), the nominal test voltage shall be as declared by the equipment manufacturer. This shall be recorded in the test report.

5.2 EXTREME TEST CONDITIONS

5.2.1 Extreme temperatures

The extreme temperature range shall be the largest temperature range given by the combination of:

- The minimum temperature range 0 °C to +35 °C
- The product operating temperature range declared by the manufacturer.

This extreme temperature range and the declared operating temperature range shall be recorded in the test report.

5.2.2 Extreme power source voltages

Tests at extreme power source voltages specified below are not required when the equipment under test is designed for operation as part of and powered by another system or piece of equipment. Where this is the case, the limit values of the host system or host equipment shall apply. The appropriate limit values shall be declared by the manufacturer and recorded in the test report.

5.2.2.1 Mains voltage

The extreme test voltage for equipment to be connected to an AC mains source shall be the nominal mains voltage ±10%.

5.2.2.2 Lead-acid battery power source used on vehicles

When radio equipment is intended for operation from the alternator-fed lead-acid battery power sources which are standard in vehicles, then extreme test voltage shall be 1.3 and 0.9 times the nominal voltage of the battery (6V, 12V etc.)

5.2.2.3 Power sources using other types of batteries

The lower extreme test voltage for equipment with power sources using the following types of battery, shall be

a) for Leclanché, alkaline, or lithium type battery: 0.85 times the nominal voltage of the battery

b) for mercury or nickel-cadmium types of battery: 0.9 times the nominal voltage of the battery.

In both cases, the upper extreme test voltage shall be 1.15 times the nominal voltage of the battery.

5.2.2.4 Other power sources

For equipment using other power sources, or capable of being operated from a variety of power sources (primary or secondary), the extreme test voltages shall be those declared by the manufacturer. These shall be recorded in the test report.

6 APPENDIX B

The Basic Rate radio parameters shall be tested in the following conditions

Parameter	Temperature	Power source
Output Power	ETC	ETC
Power control	NTC	NTC
Modulation index	ETC	ETC
Initial Carrier Frequency accuracy	ETC	ETC
Carrier Frequency drift	ETC	ETC
Conducted in-band spurious emissions	ETC	ETC
Radiated in-band emissions	NTC	NTC
Sensitivity	ETC	ETC
Interference Performance	NTC	NTC
Intermodulation Characteristics	NTC	NTC
Out-of-band blocking	NTC	NTC
Maximum Usable Level	NTC	NTC
Receiver Signal Strength Indicator	NTC	NTC

ETC = Extreme Test Conditions
NTC = Nominal Test Conditions

The Enhanced Data Rate radio parameters shall be tested in the following conditions

Parameter	Temperature	Power source
Modulation accuracy	ETC	ETC
Carrier frequency stability	ETC	ETC
In-band spurious emissions	ETC	ETC
Relative transmit power	ETC	ETC
Sensitivity	ETC	ETC
BER floor sensitivity	NTC	NTC
Interference Performance	NTC	NTC
Maximum usable level	NTC	NTC

ETC = Extreme Test Conditions
NTC = Nominal Test Conditions

7 APPENDIX C

7.1 ENHANCED DATA RATE MODULATION ACCURACY

The Enhanced Data Rate modulation accuracy is defined by the differential error vector, being the difference between the vectors representing consecutive symbols of the transmitted signal, after passing the signal through a specified measurement filter, sampling it at the symbol rate with an optimum sampling phase and compensating it for carrier frequency error and for the ideal carrier phase changes. The magnitude of the normalized differential error vector is called the Differential Error Vector Magnitude (DEVM). The objective of the DEVM is to estimate the modulation errors that would be perceived by a differential receiver.

In an ideal transmitter, the input bit sequence $\{b_j\}$ is mapped onto a complex valued symbol sequence $\{S_k\}$. Subsequently, this symbol sequence is transformed into a baseband signal S(t) by means of a pulse-shaping filter.

In an actual transmitter implementation, the bit sequence $\{b_j\}$ generates a baseband equivalent transmitted signal Y(t). This signal Y(t) contains, besides the desired component S(t), multiple distortion components. This is illustrated in Figure 7.1.

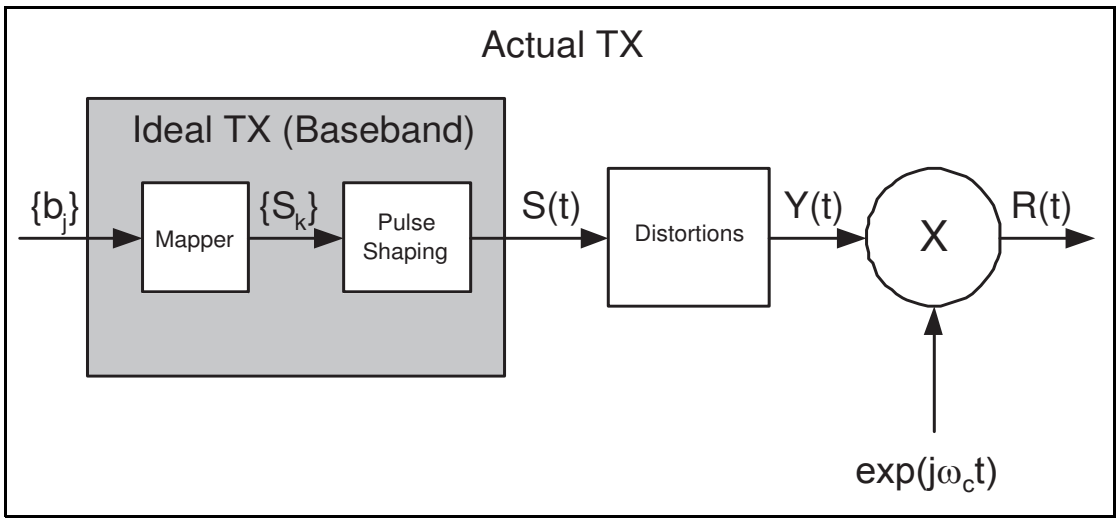


Figure 7.1: TX model.

Let Z(t) be the output of the measurement filter after first compensating the received signal for the initial center frequency error, ω_i, of the received packet, i.e. the output after down conversion and filtering the transmit signal R(t) (see Figure 7.2).The measurement filter is defined by a square-root raised cosine shaping filter with a roll-off factor equal to 0.4 and 3 dB bandwidth of ±500 kHz.

Let $\{Z_k(\varepsilon)\}$ be the sequence of samples obtained by sampling the signal Z(t) with a sampling period equal to the symbol period T and a sampling phase equal to ε such that $Z_k(\varepsilon)=Z((k+\varepsilon)T)$. Note that this sequence $\{Z_k(\varepsilon)\}$ would coincide with the symbol sequence $\{S_k\}$ if no distortion is present and the correct timing phase ε is chosen.

To reflect the behavior of a typical differential receiver, the sample sequence $\{Z_k(\varepsilon)\}$ should be compensated for carrier frequency drift. Therefore, the sequence $\{Z_k(\varepsilon)\}$ is multiplied by a factor W^{-k} in which $W = e^{j\omega T}$ accounts for the frequency offset ω. A constant value of ω is used for each DEVM block of N = 50 symbols, but ω may vary between DEVM blocks (note that the values of ω can be used to measure carrier frequency drift.

In addition, $\{Z_k(\varepsilon)\}$ is compensated for the ideal phase changes between symbols by multiplying it with the complex conjugate of the symbol sequence $\{S_k\}$. However, it is not necessary to compensate $\{Z_k(\varepsilon)\}$ for initial carrier phase or output power of the transmitter.

Let $\{Q_k(\varepsilon,\omega)\}$ denote the compensated sequence $\{Z_k(\varepsilon)\}$, where the ideal phase changes have been removed and ε and ω are chosen optimally to minimize the DEVM, (i.e. remove time and frequency uncertainty). For a transmitter with no distortions other than a constant frequency error, $\{Q_k(\varepsilon,\omega)\}$ is a complex constant that depends on the initial carrier phase and the output power of the transmitter.

The differential error sequence $\{E_k(\varepsilon,\omega)\}$ is defined as the difference between $\{Q_k(\varepsilon,\omega)\}$ and $\{Q_{k-1}(\varepsilon,\omega)\}$. This reflects the modulation errors that would be perceived by a differential receiver. For a transmitter with no distortions other than a constant frequency error, $\{E_k(\varepsilon,\omega)\}$ is zero.

The definitions of the DEVM measures are based upon this differential error sequence $\{E_k(\varepsilon,\omega)\}$. The generation of the error sequence is depicted in Figure 7.2.

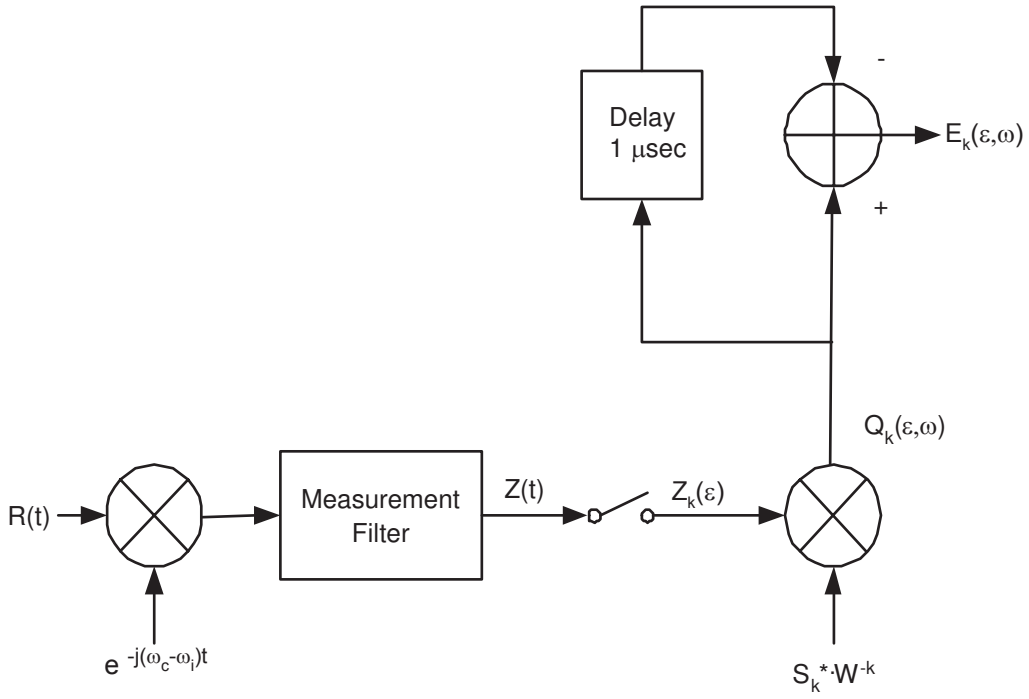


Figure 7.2: Error sequence generation.

RMS DEVM

The root mean squared DEVM (RMS DEVM) computed over N = 50 symbols is defined as:

$$RMS\ DEVM = \min_{\varepsilon,\omega} \left\{ \sqrt{\sum_{k=1}^{N} \left|E_k(\varepsilon,\omega)\right|^2 \Big/ \sum_{k=1}^{N} \left|Q_k(\varepsilon,\omega)\right|^2} \right\}$$

As can be seen from the expression above, the RMS DEVM is the square-root of the normalized power of the error.

Peak DEVM

The DEVM at symbol k is defined as

$$DEVM\,(k) = \sqrt{\frac{\left|E_k\,(\varepsilon_0, \omega_0)\right|^2}{\sum_{j=1}^{N}\left|Q_j\,(\varepsilon_0, \omega_0)\right|^2 / N}} \qquad \text{(EQ 6)}$$

where ε_0 and ω_0 are the values for ε and ω used to calculate the RMS DEVM.

The peak DEVM is defined as:

$$Peak\ DEVM = \max_{k}\ \{\mathrm{DEVM}\ (k)\} \qquad \text{(EQ 7)}$$

Core System Package [Controller volume]
Part B

BASEBAND SPECIFICATION

This document describes the specification of the Bluetooth link controller which carries out the baseband protocols and other low-level link routines.

CONTENTS

1 GENERAL DESCRIPTION

This part specifies the normal operation of a Bluetooth baseband.

The Bluetooth system provides a point-to-point connection or a point-to-multi-point connection, see (a) and (b) in Figure 1.1 on page 63. In a point-to-point connection the physical channel is shared between two Bluetooth devices. In a point-to-multipoint connection, the physical channel is shared among several Bluetooth devices. Two or more devices sharing the same physical channel form a *piconet*. One Bluetooth device acts as the master of the piconet, whereas the other device(s) act as slave(s). Up to seven slaves can be active in the piconet. Additionally, many more slaves can remain connected in a parked state. These parked slaves are not active on the channel, but remain synchronized to the master and can become active without using the connection establishment procedure. Both for active and parked slaves, the channel access is controlled by the master.

Piconets that have common devices are called a *scatternet,* see (c) in Figure 1.1 on page 63. Each piconet only has a single master, however, slaves can participate in different piconets on a time-division multiplex basis. In addition, a master in one piconet can be a slave in other piconets. Piconets shall not be frequency synchronized and each piconet has its own hopping sequence.

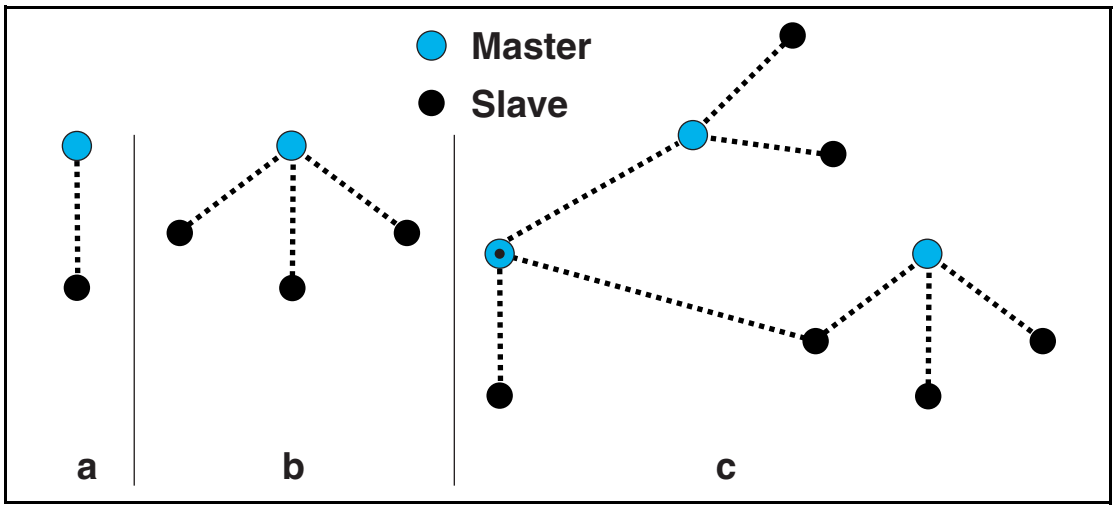


Figure 1.1: Piconets with a single slave operation (a), a multi-slave operation (b) and a scatternet operation (c).

Data is transmitted over the air in packets. Two modulation modes are defined: a mandatory mode called Basic Rate and an optional mode called Enhanced Data Rate. The symbol rate for all modulation schemes is 1 Ms/s. The gross air data rate is 1 Mbps for Basic Rate. Enhanced Data Rate has a primary modulation mode that provides a gross air data rate of 2 Mbps, and a secondary modulation mode that provides a gross air data rate of 3 Mbps.

The general Basic Rate packet format is shown in Figure 1.2. Each packet consists of 3 entities: the access code, the header, and the payload.

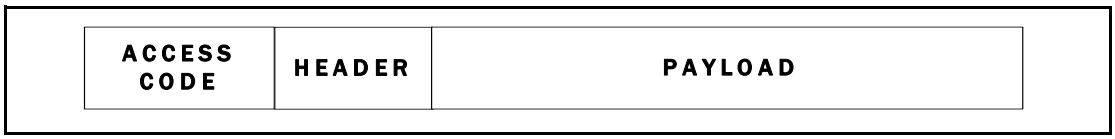


Figure 1.2: Standard Basic Rate packet format.

The general Enhanced Data Rate packet format is shown in Standard Enhanced Data Rate packet format. Each packet consists of 6 entities: the access code, the header, the guard period, the synchronization sequence, the Enhanced Data Rate payload and the trailer. The access code and header use the same modulation scheme as for Basic Rate packets while the synchronization sequence, the Enhanced Data Rate payload and the trailer use the Enhanced Data Rate modulation scheme. The guard time allows for the transition between the modulation schemes.

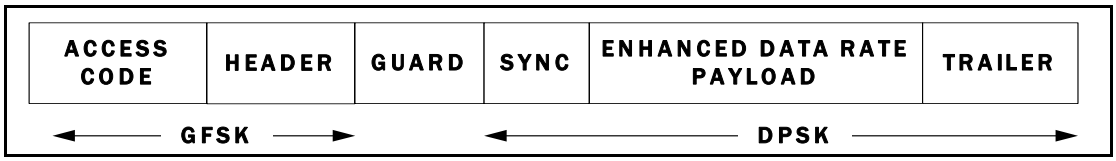


Figure 1.3: Standard Enhanced Data Rate packet format

1.1 BLUETOOTH CLOCK

Every Bluetooth device shall have a native clock that shall be derived from a free running system clock. For synchronization with other devices, offsets are used that, when added to the native clock, provide temporary Bluetooth clocks that are mutually synchronized. It should be noted that the Bluetooth clock has no relation to the time of day; it may therefore be initialized to any value. The clock has a cycle of about a day. If the clock is implemented with a counter, a 28-bit counter is required that shall wrap around at 2^{28}-1. The least significant bit (LSB) shall tick in units of 312.5 μs (i.e. half a time slot), giving a clock rate of 3.2 kHz.

The clock determines critical periods and triggers the events in the device. Four periods are important in the Bluetooth system: 312.5 μs, 625 μs, 1.25 ms, and 1.28 s; these periods correspond to the timer bits CLK_0, CLK_1, CLK_2, and CLK_{12}, respectively, see Figure 1.4 on page 65.

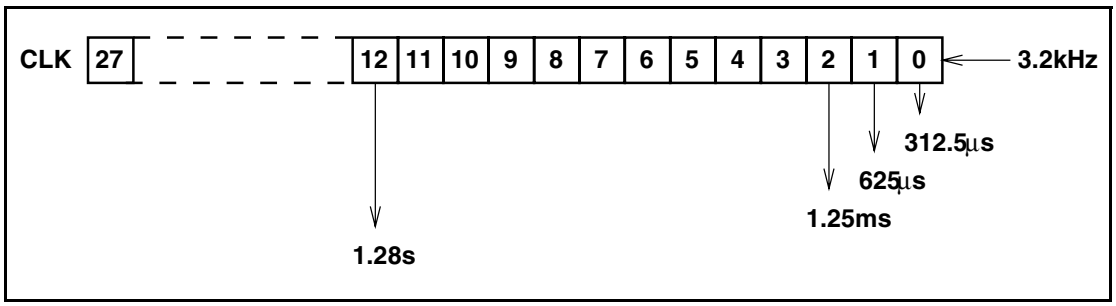


Figure 1.4: Bluetooth clock.

In the different modes and states a device can reside in, the clock has different appearances:

- CLKN native clock
- CLKE estimated clock
- CLK master clock

CLKN is the native clock and shall be the reference to all other clock appearances. In STANDBY and in Park, Hold and Sniff mode the native clock may be driven by a low power oscillator (LPO) with worst case accuracy (+/-250ppm). Otherwise, the native clock shall be driven by the reference crystal oscillator with worst case accuracy of +/-20ppm.

See Section 2.2.4 on page 72 for the definition of CLK and Section 2.4.1 on page 76 for the definition of CLKE.

The master shall never adjust its native clock during the existence of the piconet.

1.2 BLUETOOTH DEVICE ADDRESSING

Each Bluetooth device shall be allocated a unique 48-bit Bluetooth device address (BD_ADDR). This address shall be obtained from the IEEE Registration Authority. The address is divided into the following three fields:

- LAP field: lower address part consisting of 24 bits
- UAP field: upper address part consisting of 8 bits
- NAP field: non-significant address part consisting of 16 bits

The LAP and UAP form the significant part of the BD_ADDR. The bit pattern in Figure 1.5 is an example BD_ADDR.

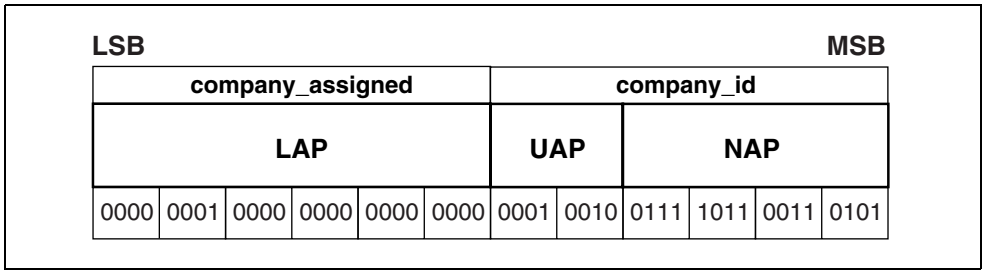


Figure 1.5: Format of BD_ADDR.

The BD_ADDR may take any values except the 64 reserved LAP values for general and dedicated inquiries (see Section 1.2.1 on page 66).

1.2.1 Reserved addresses

A block of 64 contiguous LAPs is reserved for inquiry operations; one LAP common to all devices is reserved for general inquiry, the remaining 63 LAPs are reserved for dedicated inquiry of specific classes of devices (see Assigned Numbers on the web site[1]). The same LAP values are used regardless of the contents of UAP and NAP. Consequently, none of these LAPs can be part of a user BD_ADDR.

The reserved LAP addresses are 0x9E8B00-0x9E8B3F. The general inquiry LAP is 0x9E8B33. All addresses have the LSB at the rightmost position, hexadecimal notation. The default check initialization (DCI) is used as the UAP whenever one of the reserved LAP addresses is used. The DCI is defined to be 0x00 (hexadecimal).

1. https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers

1.3 ACCESS CODES

In the Bluetooth system all transmissions over the physical channel begin with an access code. Three different access codes are defined, see also Section 6.3.1 on page 111:

- device access code (DAC)
- channel access code (CAC)
- inquiry access code (IAC)

All access codes are derived from the LAP of a device address or an inquiry address. The device access code is used during **page**, **page scan** and **page response** substates and shall be derived from the paged device's BD_ADDR. The channel access code is used in the **CONNECTION** state and forms the beginning of all packets exchanged on the piconet physical channel. The channel access code shall be derived from the LAP of the master's BD_ADDR. Finally, the inquiry access code shall be used in the **inquiry** substate. There is one general IAC (GIAC) for general inquiry operations and there are 63 dedicated IACs (DIACs) for dedicated inquiry operations.

The access code also indicates to the receiver the arrival of a packet. It is used for timing synchronization and offset compensation. The receiver correlates against the entire synchronization word in the access code, providing very robust signalling.

2 PHYSICAL CHANNELS

The lowest architectural layer in the Bluetooth system is the physical channel. A number of types of physical channels are defined. All Bluetooth physical channels are characterized by the combination of a pseudo-random frequency hopping sequence, the specific slot timing of the transmissions, the access code and packet header encoding. These aspects, together with the range of the transmitters, define the signature of the physical channel. For the basic and adapted piconet physical channels frequency hopping is used to change frequency periodically to reduce the effects of interference and to satisfy local regulatory requirements.

Two devices that wish to communicate use a shared physical channel for this communication. To achieve this, their transceivers must be tuned to the same RF frequency at the same time, and they must be within a nominal range of each other.

Given that the number of RF carriers is limited and that many Bluetooth devices may be operating independently within the same spatial and temporal area there is a strong likelihood of two independent Bluetooth devices having their transceivers tuned to the same RF carrier, resulting in a physical channel collision. To mitigate the unwanted effects of this collision each transmission on a physical channel starts with an access code that is used as a correlation code by devices tuned to the physical channel. This channel access code is a property of the physical channel. The access code is always present at the start of every transmitted packet.

Four Bluetooth physical channels are defined. Each is optimized and used for a different purpose. Two of these physical channels (the basic piconet channel and adapted piconet channel) are used for communication between connected devices and are associated with a specific piconet. The remaining physical channels are used for discovering (the inquiry scan channel) and connecting (the page scan channel) Bluetooth devices.

A Bluetooth device can only use one of these physical channels at any given time. In order to support multiple concurrent operations the device uses time-division multiplexing between the channels. In this way a Bluetooth device can appear to operate simultaneously in several piconets, as well as being discoverable and connectable.

Whenever a Bluetooth device is synchronized to the timing, frequency and access code of a physical channel it is said to be 'connected' to this channel (whether or not it is actively involved in communications over the channel.) At a minimum, a device need only be capable of connection to one physical channel at a time, however, advanced devices may be capable of connecting simultaneously to more than one physical channel, but the specification does not assume that this is possible.

2.1 PHYSICAL CHANNEL DEFINITION

Physical channels are defined by a pseudo-random RF channel hopping sequence, the packet (slot) timing and an access code. The hopping sequence is determined by the UAP and LAP of a Bluetooth device address and the selected hopping sequence. The phase in the hopping sequence is determined by the Bluetooth clock. All physical channels are subdivided into time slots whose length is different depending on the physical channel. Within the physical channel, each reception or transmission event is associated with a time slot or time slots. For each reception or transmission event an RF channel is selected by the hop selection kernel (see Section 2.6 on page 82).The maximum hop rate is 1600 hops/s in the **CONNECTION** state and the maximum is 3200 hops/s in the **inquiry** and **page** substates.

The following physical channels are defined:

- basic piconet physical channel
- adapted piconet physical channel
- page scan physical channel
- inquiry scan physical channel

2.2 BASIC PICONET PHYSICAL CHANNEL

During the **CONNECTION** state the basic piconet physical channel is used by default. The adapted piconet physical channel may also be used. The adapted piconet physical channel is identical to the basic piconet physical channel except for the differences listed in Section 2.3 on page 75.

2.2.1 Master-slave definition

The basic piconet physical channel is defined by the master of the piconet. The master controls the traffic on the piconet physical channel by a polling scheme. (see Section 8.5 on page 167)

By definition, the device that initiates a connection by paging is the master. Once a piconet has been established, master-slave roles may be exchanged. This is described in Section 8.6.5 on page 175.

2.2.2 Hopping characteristics

The basic piconet physical channel is characterized by a pseudo-random hopping through all 79 RF channels. The frequency hopping in the piconet physical channel is determined by the Bluetooth clock and BD_ADDR of the master. When the piconet is established, the master clock is communicated to the slaves. Each slave shall add an offset to its native clock to synchronize with the master clock. Since the clocks are independent, the offsets must be updated regularly. All devices participating in the piconet are time-synchronized and hop-synchronized to the channel.

The basic piconet physical channel uses the basic channel hopping sequence and is described in Section 2.6 on page 82.

2.2.3 Time slots

The basic piconet physical channel is divided into time slots, each 625 μs in length. The time slots are numbered according to the most significant 27 bits of the Bluetooth clock CLK_{28-1} of the piconet master. The slot numbering ranges from 0 to 2^{27}-1 and is cyclic with a cycle length of 2^{27}. The time slot number is denoted as k.

A TDD scheme is used where master and slave alternatively transmit, see Figure 2.1 on page 71. The packet start shall be aligned with the slot start. Packets may extend over up to five time slots.

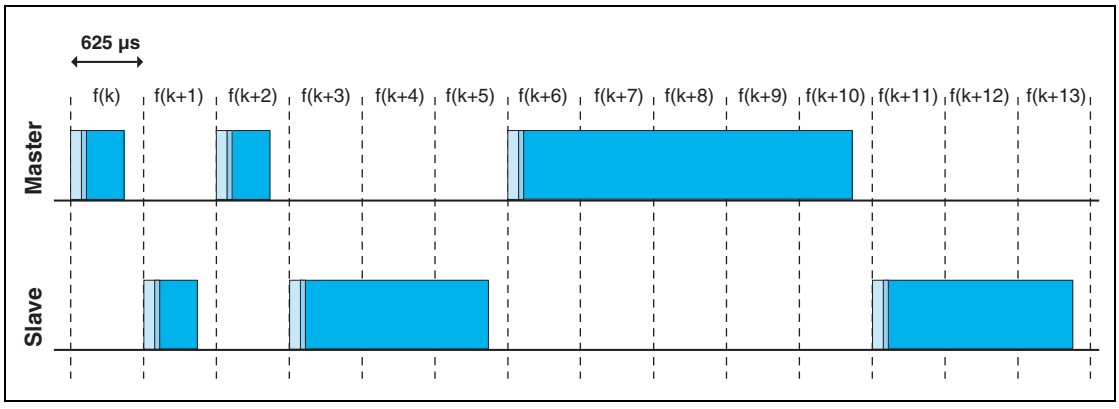


Figure 2.1: Multi-slot packets

The term *slot pairs* is used to indicate two adjacent time slots starting with a master-to-slave transmission slot.

2.2.4 Piconet clocks

CLK is the master clock of the piconet. It shall be used for all timing and scheduling activities in the piconet. All devices shall use the CLK to schedule their transmission and reception. The CLK shall be derived from the native clock CLKN (see Section 1.1 on page 64) by adding an offset, see Figure 2.2 on page 72. The offset shall be zero for the master since CLK is identical to its own native clock CLKN. Each slave shall add an appropriate offset to its CLKN such that the CLK corresponds to the CLKN of the master. Although all CLKNs in the devices run at the same nominal rate, mutual drift causes inaccuracies in CLK. Therefore, the offsets in the slaves must be regularly updated such that CLK is approximately CLKN of the master.

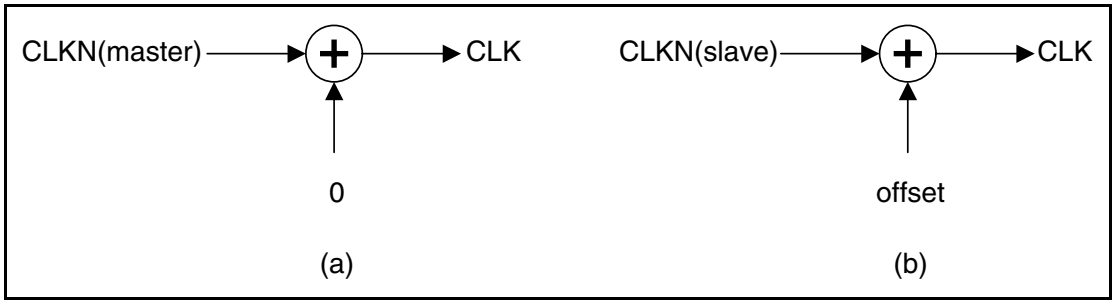


Figure 2.2: Derivation of CLK in master (a) and in slave (b).

2.2.5 Transmit/receive timing

The master transmission shall always start at even numbered time slots (CLK_1=0) and the slave transmission shall always start at odd numbered time slots (CLK_1=1). Due to packet types that cover more than a single slot, master transmission may continue in odd numbered slots and slave transmission may continue in even numbered slots, see Figure 2.1 on page 71.

All timing diagrams shown in this chapter are based on the signals as present at the antenna. The term “exact” when used to describe timing refers to an ideal transmission or reception and neglects timing jitter and clock frequency imperfections.

The average timing of packet transmission shall not drift faster than 20 ppm relative to the ideal slot timing of 625 µs. The instantaneous timing shall not deviate more than 1 µs from the average timing. Thus, the absolute packet transmission timing t_k of slot boundary k shall fulfill the equation:

$$t_k = \left(\sum_{i=1}^{k} (1 + d_i) T_N \right) + j_k + \text{offset}, \qquad \text{(EQ 1)}$$

where T_N is the nominal slot length (625 µs), j_k denotes jitter ($|j_k| \leq 1$ µs) at the start of slot k, and, d_k, denotes the drift ($|d_k| \leq 20$ ppm) within slot k. The jitter and drift may vary arbitrarily within the given limits for every slot, while *offset* is an arbitrary but fixed constant. For hold, park and sniff the drift and jitter parameters specified in Link Manager Protocol [Part C] Section 4.3.1 on page 262 apply.

2.2.5.1 Piconet physical channel timing

In the figures, only single-slot packets are shown as an example.

The master TX/RX timing is shown in Figure 2.3 on page 73. In Figure 2.3 and Figure 2.4 the channel hopping frequencies are indicated by *f(k)* where k is the time slot number. After transmission, a return packet is expected N × 625 µs after the start of the TX packet where N is an odd, integer larger than 0. N depends on the type of the transmitted packet.

To allow for some time slipping, an uncertainty window is defined around the exact receive timing. During normal operation, the window length shall be 20 µs, which allows the RX packet to arrive up to 10 µs too early or 10 µs too late. It is recommended that slaves implement variable sized windows or time tracking to accommodate a master's absence of more than 250ms.

During the beginning of the RX cycle, the access correlator shall search for the correct channel access code over the uncertainty window. If an event trigger does not occur the receiver may go to sleep until the next RX event. If in the course of the search, it becomes apparent that the correlation output will never exceed the final threshold, the receiver may go to sleep earlier. If a trigger event occurs, the receiver shall remain open to receive the rest of the packet unless the packet is for another device, a non-recoverable header error is detected, or a non-recoverable payload error is detected.

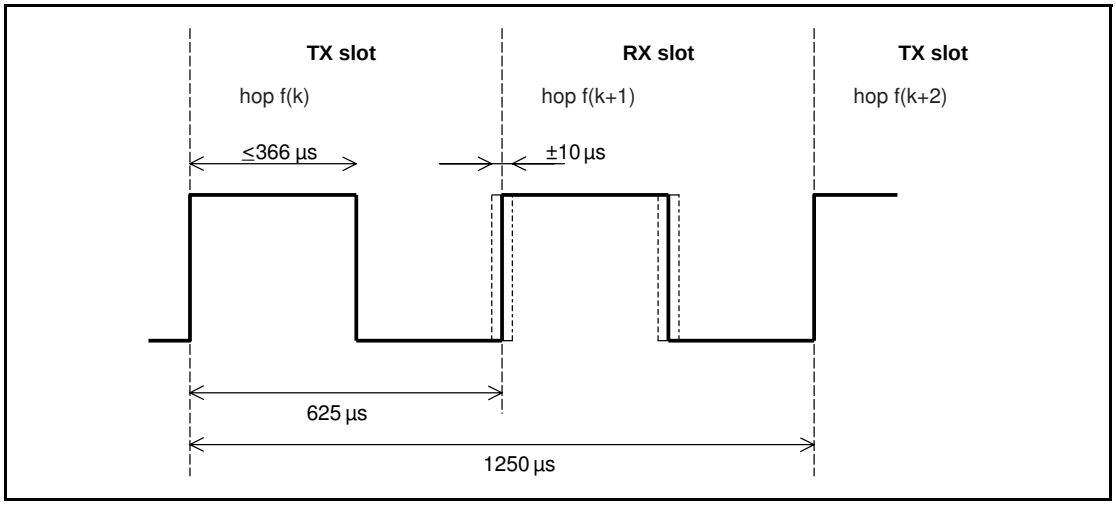


Figure 2.3: RX/TX cycle of master transceiver in normal mode for single-slot packets.

Each master transmission shall be derived from bit 2 of the Master's native Bluetooth clock, thus the current transmission will be scheduled Mx1250µs after the start of the previous master TX burst where M depends on the transmitted and received packet type and is an even, integer larger than 0. The master TX timing shall be derived from the master's native Bluetooth clock, and thus it will not be affected by time drifts in the slave(s).

Slaves maintain an estimate of the master's native clock by adding a timing offset to the slave's native clock (see Section 2.2.4 on page 72). This offset shall be updated each time a packet is received from the master. By comparing the exact RX timing of the received packet with the estimated RX timing, slaves shall correct the offset for any timing misalignments. Since only the channel access code is required to synchronize the slave, slave RX timing can be corrected with any packet sent in the master-to-slave transmission slot.

The slave's TX/RX timing is shown in Figure 2.4 on page 74. The slave's transmission shall be scheduled N × 625µs after the start of the slave's RX packet where N is an odd, positive integer larger than 0. If the slave's RX timing drifts, so will its TX timing. During periods when a slave is in the active mode (see Section 8.6 on page 168) and is not able to receive any valid channel access codes from the master, the slave may increase its receive uncertainty window and/or use predicted timing drift to increase the probability of receiving the master's bursts when reception resumes.

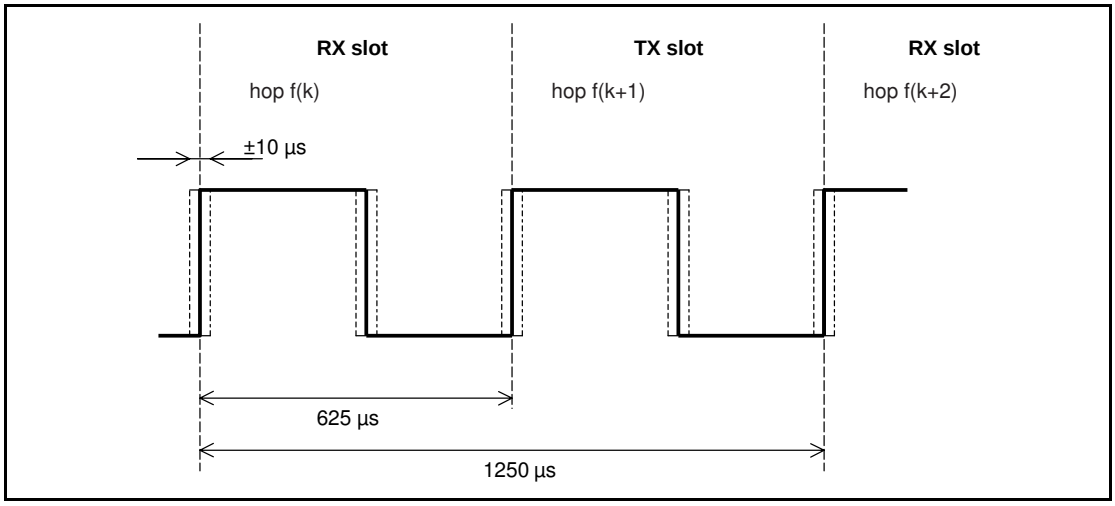


Figure 2.4: RX/TX cycle of slave transceiver in normal mode for single-slot packets.

2.2.5.2 Piconet physical channel re-synchronization

In the piconet physical channel, a slave may loose synchronization if it does not receive a packet from the master at least every 250ms (or less if the low power clock is used). This may occur in sniff, hold, park, in a scatternet or due to interference. When re-synchronizing to the piconet physical channel a slave device shall listen for the master before it may send information. In this case, the length of the search window in the slave device may be increased from 20 µs to a larger value X µs as illustrated in Figure 2.5 on page 75. Note that only RX hop frequencies are used. The hop frequency used in the master-to-slave (RX) slot shall also be used in the uncertainty window, even when it is extended into the preceding time interval normally used for the slave-to-master (TX) slot.

If the length of search window, X, exceeds 1250 μs, consecutive windows shall avoid overlapping search windows. Consecutive windows should instead be centered at *f(k), f(k+4),... f(k+4i)* (where 'i' is an integer), which gives a maximum value X=2500 μs, or even at *f(k), f(k+6),...f(k+6i)* which gives a maximum value X=3750 μs. The RX hop frequencies used shall correspond to the master-to-slave transmission slots.

It is recommended that single slot packets are transmitted by the master during slave re-synchronization.

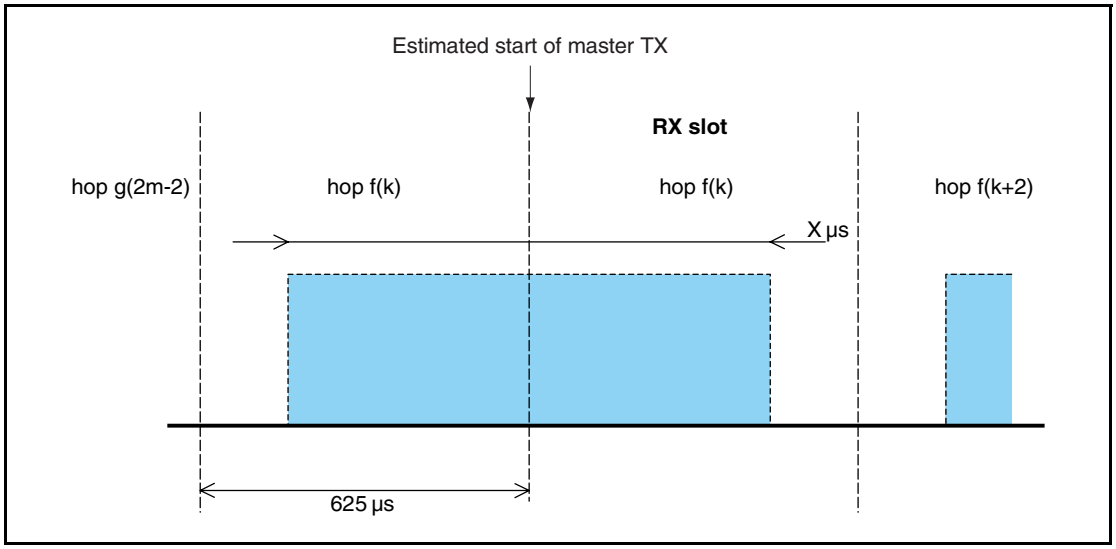


Figure 2.5: RX timing of slave returning from hold mode.

2.3 ADAPTED PICONET PHYSICAL CHANNEL

2.3.1 Hopping characteristics

The adapted piconet physical channel shall use at least N_{min} RF channels (where N_{min} is 20).

The adapted piconet physical channel uses the adapted channel hopping sequence described in Section 2.6 on page 82.

Adapted piconet physical channels can be used for connected devices that have adaptive frequency hopping (AFH) enabled. There are two distinctions between basic and adapted piconet physical channels. The first is that the same channel mechanism that makes the slave frequency the same as the preceding master transmission. The second aspect is that the adapted piconet physical channel may be based on less than the full 79 frequencies of the basic piconet physical channel.

2.4 PAGE SCAN PHYSICAL CHANNEL

Although master and slave roles are not defined prior to a connection, the term *master* is used for the paging device (that becomes a master in the **CONNECTION** state) and *slave* is used for the page scanning device (that becomes a slave in the **CONNECTION** state).

2.4.1 Clock estimate for paging

A paging device uses an estimate of the native clock of the page scanning device, CLKE; i.e. an offset shall be added to the CLKN of the pager to approximate the CLKN of the recipient, see Figure 2.6 on page 76. CLKE shall be derived from the reference CLKN by adding an offset. By using the CLKN of the recipient, the pager might be able to speed up the connection establishment.

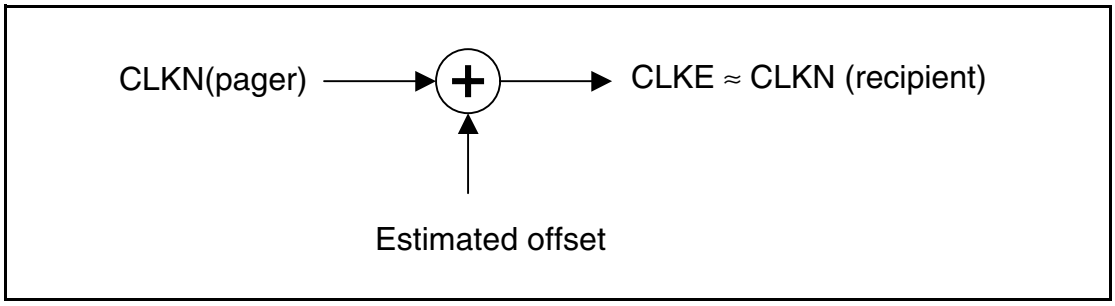


Figure 2.6: Derivation of CLKE.

2.4.2 Hopping characteristics

The page scan physical channel follows a slower hopping pattern than the basic piconet physical channel and is a short pseudo-random hopping sequence through the RF channels. The timing of the page scan channel shall be determined by the native Bluetooth clock of the scanning device. The frequency hopping sequence is determined by the Bluetooth address of the scanning device.

The page scan physical channel uses the page, master page response, slave page response, and page scan hopping sequences specified in Section 2.6 on page 82.

2.4.3 Paging procedure timing

During the paging procedure, the master shall transmit paging messages (see Table 8.3 on page 159) corresponding to the slave to be connected. Since the paging message is a very short packet, the hop rate is 3200 hops/s. In a single TX slot interval, the paging device shall transmit on two different hop frequencies. In Figure 2.7 through Figure 2.11, *f(k)* is used for the frequencies of the page hopping sequence and *f'(k)* denotes the corresponding page response sequence frequencies. The first transmission starts where CLK_0 = 0 and the second transmission starts where CLK_0 = 1.

In a single RX slot interval, the paging device shall listen for the slave page response message on two different hop frequencies. Similar to transmission, the nominal reception starts where CLK_0 = 0 and the second reception nominally starts where CLK_0 = 1; see Figure 2.7 on page 77. During the TX slot, the paging device shall send the paging message at the TX hop frequencies *f(k)* and *f(k+1)*. In the RX slot, it shall listen for a response on the corresponding RX hop frequencies *f'(k)* and *f'(k+1)*. The listening periods shall be exactly timed 625 µs after the corresponding paging packets, and shall include a ±10 µs uncertainty window.

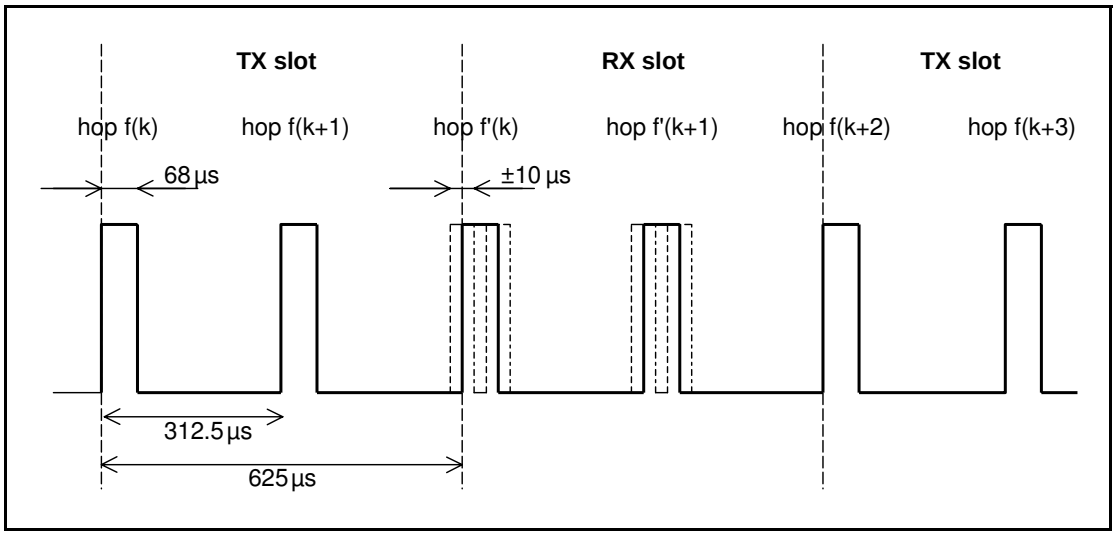


Figure 2.7: RX/TX cycle of transceiver in PAGE mode.

2.4.4 Page response timing

At connection setup a master page response packet is transmitted from the master to the slave (see Table 8.3 on page 159). This packet establishes the timing and frequency synchronization. After the slave device has received the page message, it shall return a response message that consists of the slave page response packet and shall follow 625 μs after the receipt of the page message. The master shall send the master page response packet in the TX slot following the RX slot in which it received the slave response, according to the RX/TX timing of the master. The time difference between the slave page response and master page response message will depend on the timing of the page message the slave received. In Figure 2.8 on page 78, the slave receives the paging message sent **first** in the master-to-slave slot. It then responds with a first slave page response packet in the first half of the slave-to-master slot. The timing of the master page response packet is based on the timing of the page message sent first in the preceding master-to-slave slot: there is an exact 1250 μs delay between the first page message and the master page response packet. The packet is sent at the hop frequency *f(k+1)* which is the hop frequency following the hop frequency *f(k)* the page message was received in.

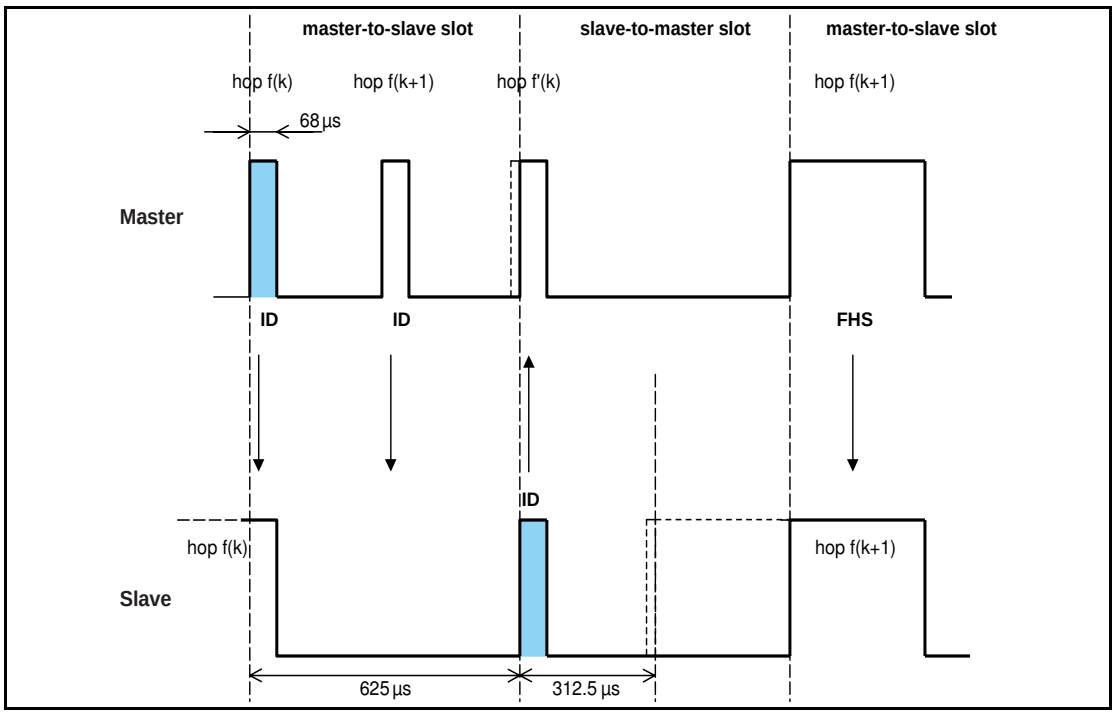


Figure 2.8: Timing of page response packets on successful page in first half slot

In Figure 2.9 on page 79, the slave receives the paging message sent **second** in the master-to-slave slot. It then responds with a slave page response packet in the second half of the slave-to-master slot exactly 625 μs after the receipt of the page message. The timing of the master page response packet is still based on the timing of the page message sent **first** in the preceding master-to-slave slot: there is an exact 1250 μs delay between the **first** page message and the master page response packet. The packet is sent at the hop frequency *f(k+2)* which is the hop frequency following the hop frequency *f(k+1)* the page message was received in.

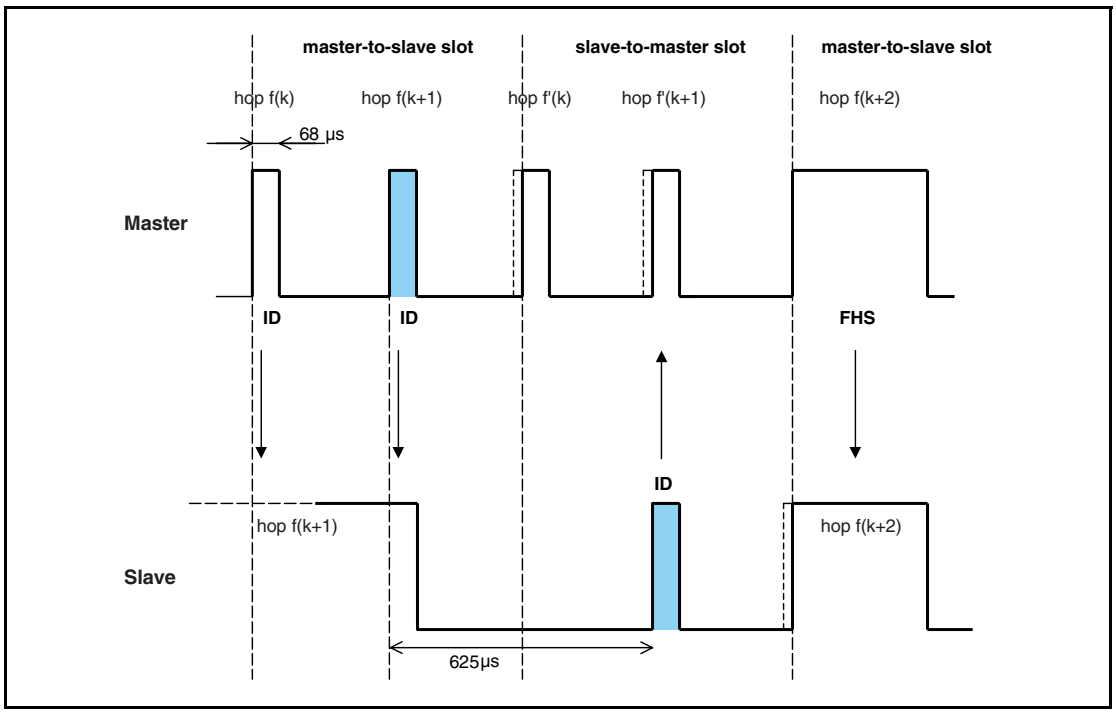


Figure 2.9: Timing of page response packets on successful page in second half slot

The slave shall adjust its RX/TX timing according to the reception of the master page response packet (and not according to the reception of the page message). That is, the second slave page response message that acknowledges the reception of the master page response packet shall be transmitted 625 µs after the start of the master page response packet.

2.5 INQUIRY SCAN PHYSICAL CHANNEL

Although master and slave roles are not defined prior to a connection, the term *master* is used for the inquiring device and *slave* is used for the inquiry scanning device.

2.5.1 Clock for inquiry

The clock used for inquiry and inquiry scan shall be the device's native clock.

2.5.2 Hopping characteristics

The inquiry scan channel follows a slower hopping pattern than the piconet physical channel and is a short pseudo-random hopping sequence through the RF channels. The timing of the inquiry scan channel is determined by the native Bluetooth clock of the scanning device while the frequency hopping sequence is determined by the general inquiry access code.

The inquiry scan physical channel uses the inquiry, inquiry response, and inquiry scan hopping sequences described in Section 2.6 on page 82.

2.5.3 Inquiry procedure timing

During the inquiry procedure, the master shall transmit inquiry messages with the general or dedicated inquiry access code. The timing for inquiry is the same as for paging (see Section 2.4.3 on page 77).

2.5.4 Inquiry response timing

An inquiry response packet is transmitted from the slave to the master after the slave has received an inquiry message (see Table 8.5 on page 167). This packet contains information necessary for the inquiring master to page the slave (see definition of the FHS packet in Section 6.5.1.4 on page 120) and follows 625μs after the receipt of the inquiry message. In Figure 2.10 and Figure 2.11, *f(k)* is used for the frequencies of the inquiry hopping sequence and f'(k) denotes the corresponding inquiry response sequence frequency. The packet is received by the master at the hop frequency *f'(k)* when the inquiry message received by the slave was first in the master-to-slave slot.

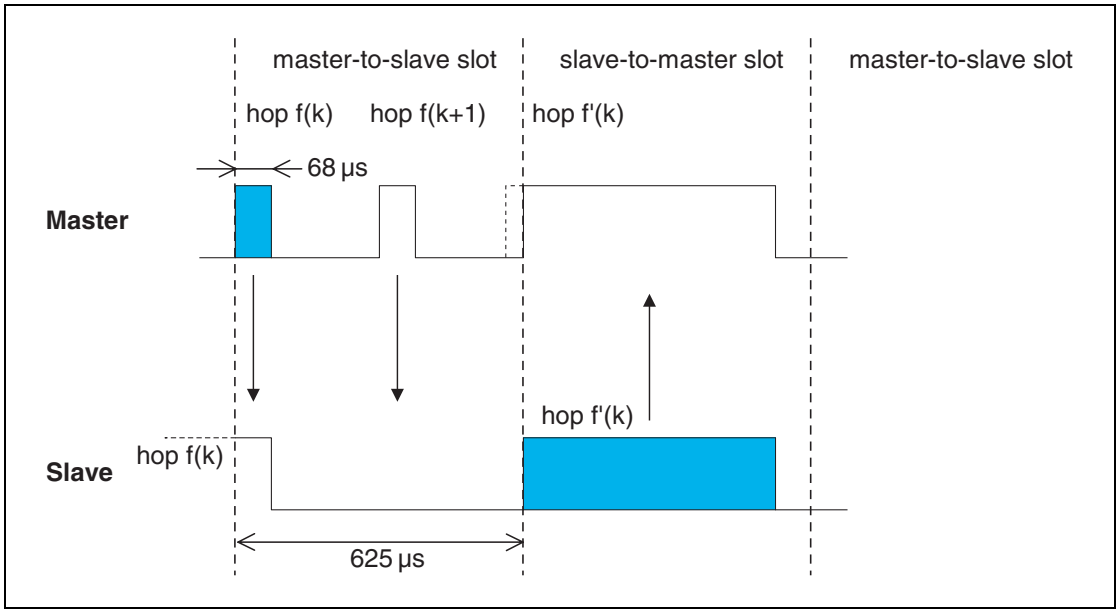


Figure 2.10: Timing of inquiry response packet on successful inquiry in first half slot

When the inquiry message received by the slave was the second in the master-to-slave slot the packet is received by the master at the hop frequency *f'(k+1)*.

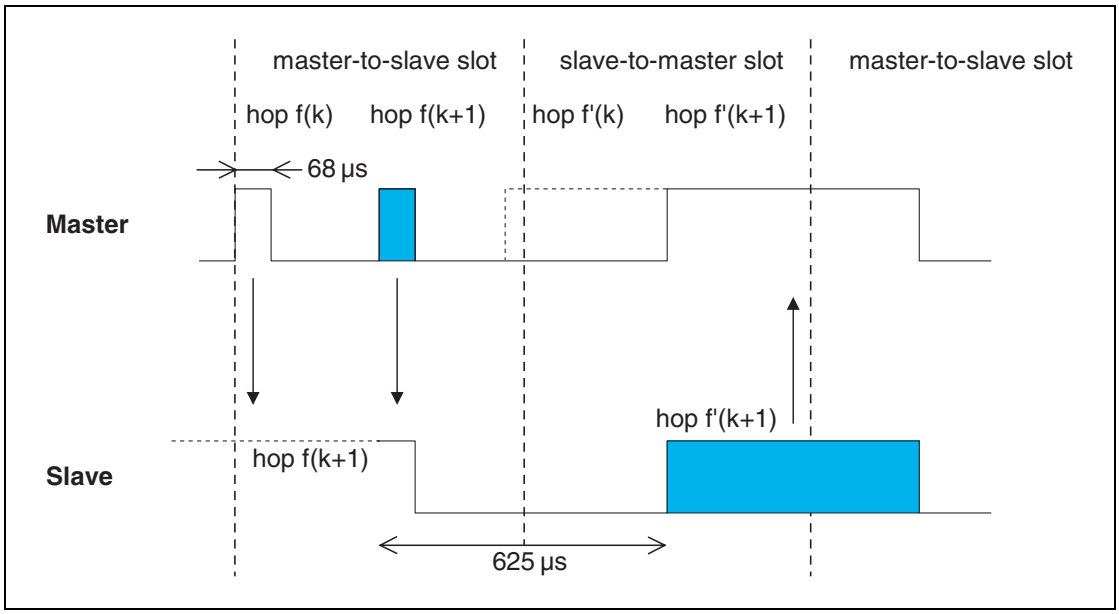


Figure 2.11: Timing of inquiry response packet on successful inquiry in second half slot

2.6 HOP SELECTION

Bluetooth devices shall use the hopping kernel as defined in the following sections.

In total, six types of hopping sequence are defined – five for the basic hop system and one for an adapted set of hop locations used by adaptive frequency hopping (AFH). These sequences are:

- A **page hopping sequence** with 32 wake-up frequencies distributed equally over the 79 MHz, with a period length of 32;
- A **page response hopping sequence** covering 32 response frequencies that are in a one-to-one correspondence to the current page hopping sequence. The master and slave use different rules to obtain the same sequence;
- An **inquiry hopping sequence** with 32 wake-up frequencies distributed equally over the 79 MHz, with a period length of 32;
- An **inquiry response hopping sequence** covering 32 response frequencies that are in a one-to-one correspondence to the current inquiry hopping sequence.
- A **basic channel hopping sequence** which has a very long period length, which does not show repetitive patterns over a short time interval, and which distributes the hop frequencies equally over the 79 MHz during a short time interval.
- An **adapted channel hopping sequence** derived from the basic channel hopping sequence which uses the same channel mechanism and may use fewer than 79 frequencies. The adapted channel hopping sequence is only used in place of the basic channel hopping sequence. All other hopping sequences are not affected by hop sequence adaptation.

2.6.1 General selection scheme

The selection scheme consists of two parts:

- selecting a sequence;
- mapping this sequence onto the hop frequencies;

The general block diagram of the hop selection scheme is shown in Figure 2.12 on page 83. The mapping from the input to a particular RF channel index is performed in the selection box.

The inputs to the selection box are the selected clock, frozen clock, N, k_{offset}, address, sequence selection and AFH_channel_map. The source of the clock input depends on the hopping sequence selected. Additionally, each hopping sequence uses different bits of the clock (see Table 2.2 on page 91). N and k_{offset} are defined in Section 2.6.4 on page 90.

The *sequence selection* input can be set to the following values:

- page scan
- inquiry scan
- page
- inquiry
- master page response
- slave page response
- inquiry response
- basic channel
- adapted channel

The address input consists of 28 bits including the entire LAP and the 4 LSBs of the UAP. This is designated as the UAP/LAP. When the basic or adapted channel hopping sequence is selected, the Bluetooth device address of the master (BD_ADDR) shall be used. When the page, master page response, slave page response, or page scan hopping sequences are selected the BD_ADDR given by the Host of the paged device shall be used (see HCI Create Connection Command [Part E] Section 7.1.5 on page 406). When the inquiry, inquiry response, or inquiry scan hopping sequences are selected, the UAP/LAP corresponding to the GIAC shall be used even if it concerns a DIAC. Whenever one of the reserved BD_ADDRs (see Section 1.2.1 on page 66) is used for generating a frequency hop sequence, the UAP shall be replaced by the default check initialization (DCI, see Section 7.1 on page 138). The hopping sequence is selected by the sequence selection input to the selection box.

When the adapted channel hopping sequence is selected, the *AFH_channel_map* is an additional input to the selection box. The *AFH_channel_map* indicates which channels shall be *used* and which shall be *unused*. These terms are defined in Section 2.6.3 on page 89.

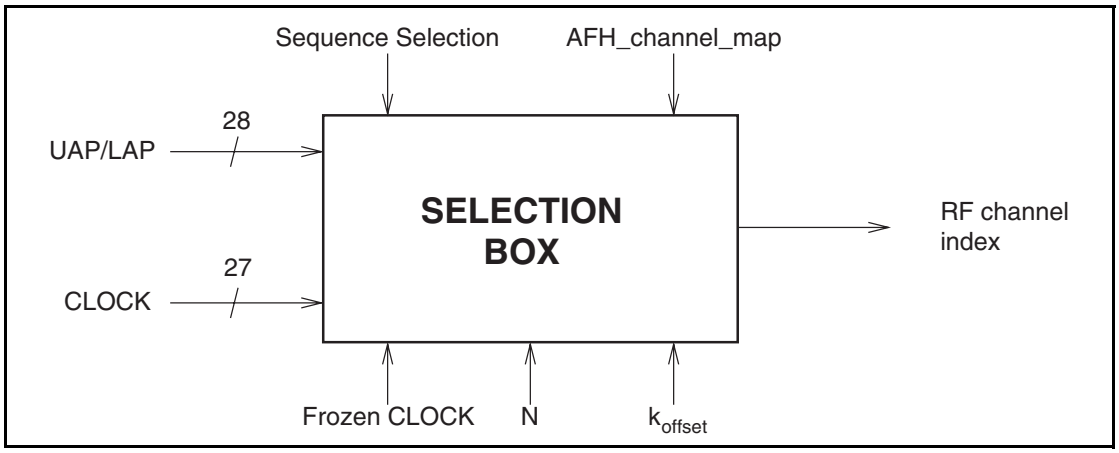


Figure 2.12: General block diagram of hop selection scheme.

The output, *RF channel index*, constitutes a pseudo-random sequence. The RF channel index is mapped to RF channel frequencies using the equation in Table 2.1 on page 29 in the Radio Specification.

The selection scheme chooses a segment of 32 hop frequencies spanning about 64 MHz and visits these hops in a pseudo-random order. Next, a different 32-hop segment is chosen, etc. In the page, master page response, slave page response, page scan, inquiry, inquiry response and inquiry scan hopping sequences, the same 32-hop segment is used all the time (the segment is selected by the address; different devices will have different paging segments). When the basic channel hopping sequence is selected, the output constitutes a pseudo-random sequence that slides through the 79 hops. The principle is depicted in Figure 2.13 on page 84.

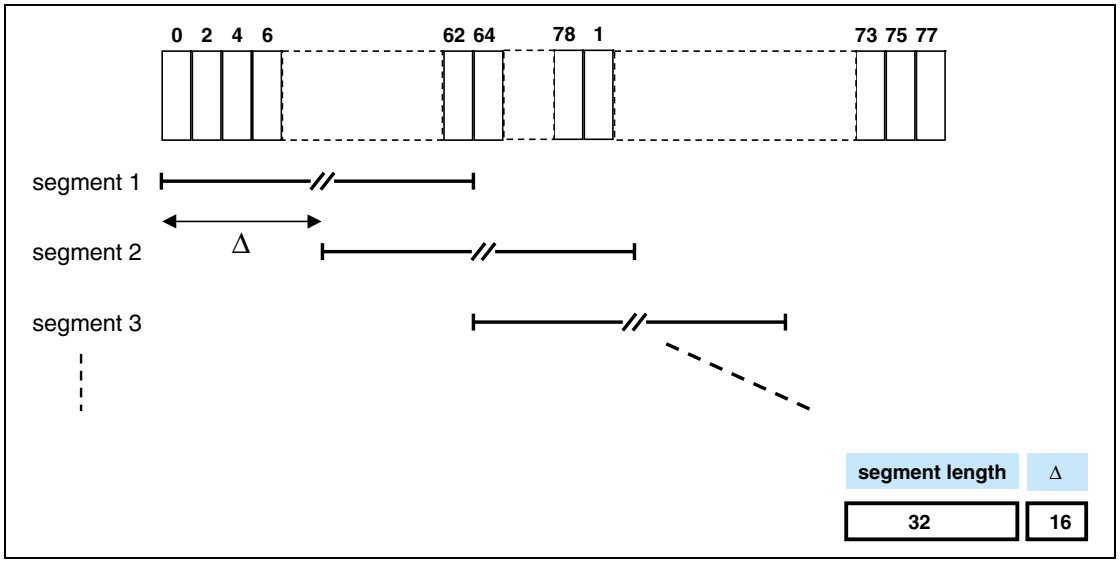


Figure 2.13: Hop selection scheme in CONNECTION state.

The RF frequency shall remain fixed for the duration of the packet. The RF frequency for the packet shall be derived from the Bluetooth clock value in the first slot of the packet. The RF frequency in the first slot after a multi-slot packet shall use the frequency as determined by the Bluetooth clock value for that slot. Figure 2.14 on page 85 illustrates the hop definition on single- and multi-slot packets.

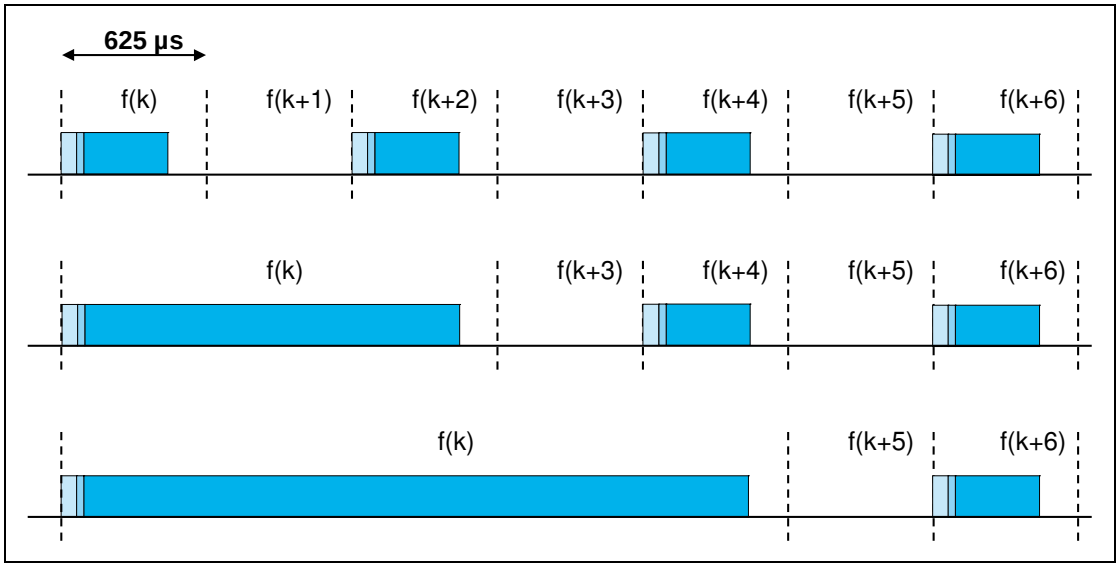


Figure 2.14: Single- and multi-slot packets.

When the adapted channel hopping sequence is used, the pseudo-random sequence contains only frequencies that are in the RF channel set defined by the *AFH_channel_map* input. The adapted sequence has similar statistical properties to the non-adapted hop sequence. In addition, the slave responds with its packet on the same RF channel that was used by the master to address that slave (or would have been in the case of a synchronous reserved slot without a validly received master-to-slave transmission). This is called the *same channel mechanism* of AFH. Thus, the RF channel used for the master to slave packet is also used for the immediately following slave to master packet. An example of the same channel mechanism is illustrated in Figure 2.15 on page 85. The same channel mechanism shall be used whenever the adapted channel hopping sequence is selected.

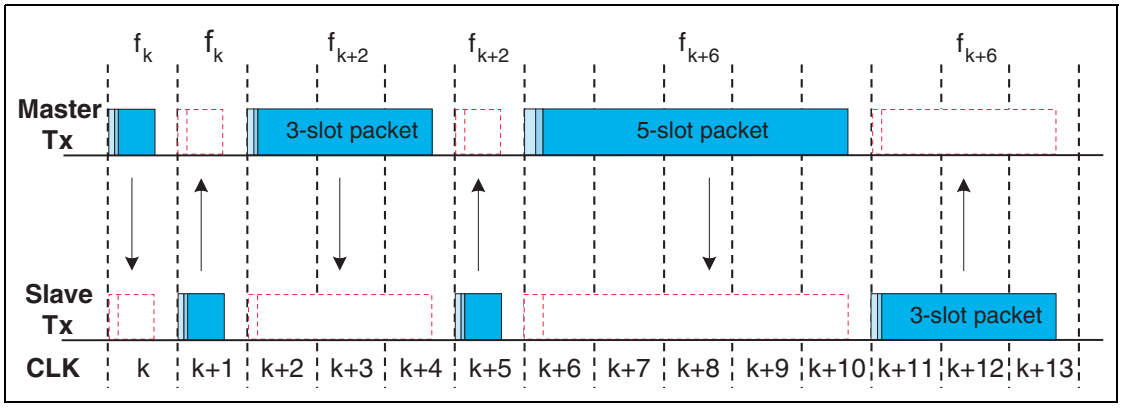


Figure 2.15: Example of the same channel mechanism.

2.6.2 Selection kernel

The basic hop selection kernel shall be as shown in Figure 2.16 on page 86 and is used for the page, page response, inquiry, inquiry response and basic channel hopping selection kernels. In these substates the AFH_channel_map input is unused. The adapted channel hopping selection kernel is described in Section 2.6.3 on page 89. The X input determines the phase in the 32-hop segment, whereas Y1 and Y2 selects between master-to-slave and slave-to-master. The inputs A to D determine the ordering within the segment, the inputs E and F determine the mapping onto the hop frequencies. The kernel addresses a register containing the RF channel indices. This list is ordered so that first all even RF channel indices are listed and then all odd hop frequencies. In this way, a 32-hop segment spans about 64 MHz.

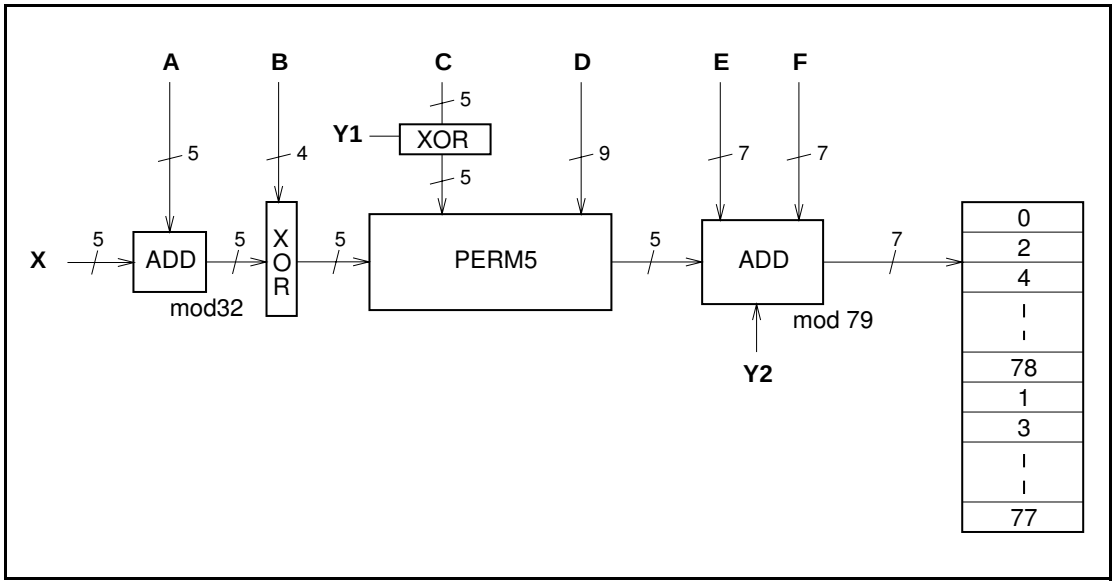


Figure 2.16: Block diagram of the basic hop selection kernel for the hop system.

The selection procedure consists of an addition, an XOR operation, a permutation operation, an addition, and finally a register selection. In the remainder of this chapter, the notation A_i is used for bit i of the BD_ADDR.

2.6.2.1 First addition operation

The first addition operation only adds a constant to the phase and applies a modulo 32 operation. For the page hopping sequence, the first addition is redundant since it only changes the phase within the segment. However, when different segments are concatenated (as in the basic channel hopping sequence), the first addition operation will have an impact on the resulting sequence.

2.6.2.2 XOR operation

Let Z' denote the output of the first addition. In the XOR operation, the four LSBs of Z' are modulo-2 added to the address bits $A_{22\text{-}19}$. The operation is illustrated in Figure 2.17 on page 87.

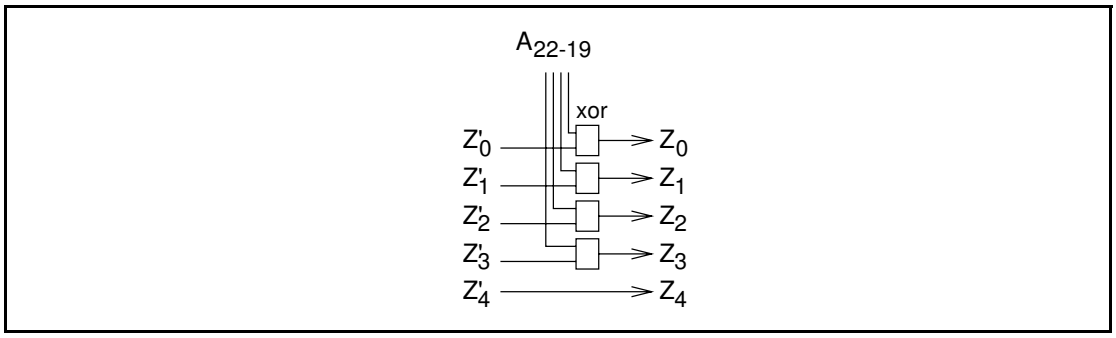


Figure 2.17: XOR operation for the hop system.

2.6.2.3 Permutation operation

The permutation operation involves the switching from 5 inputs to 5 outputs for the hop system, controlled by the control word. The permutation or switching box shall be as shown in Figure 2.18 on page 88. It consists of 7 stages of butterfly operations. The control of the butterflies by the control signals P is shown in Table 2.1. $P_{0\text{-}8}$ corresponds to $D_{0\text{-}8}$, and, P_{i+9} corresponds to $C_i \oplus Y1$ for $i = 0 \ldots 4$ in Figure 2.16.

Control signal	Butterfly		Control signal	Butterfly
P_0	$\{Z_0,Z_1\}$		P_8	$\{Z_1,Z_4\}$
P_1	$\{Z_2,Z_3\}$		P_9	$\{Z_0,Z_3\}$
P_2	$\{Z_1,Z_2\}$		P_{10}	$\{Z_2,Z_4\}$
P_3	$\{Z_3,Z_4\}$		P_{11}	$\{Z_1,Z_3\}$
P_4	$\{Z_0,Z_4\}$		P_{12}	$\{Z_0,Z_3\}$
P_5	$\{Z_1,Z_3\}$		P_{13}	$\{Z_1,Z_2\}$
P_6	$\{Z_0,Z_2\}$			
P_7	$\{Z_3,Z_4\}$			

Table 2.1: Control of the butterflies for the hop system

The Z input is the output of the XOR operation as described in the previous section. The butterfly operation can be implemented with multiplexers as depicted in Figure 2.19 on page 88.

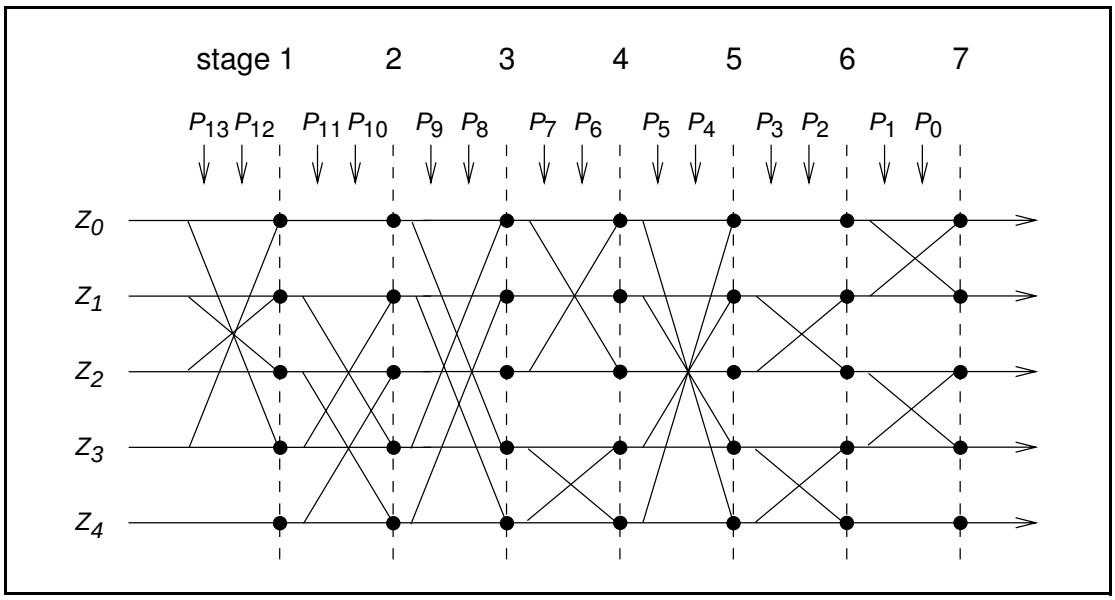


Figure 2.18: Permutation operation for the hop system.

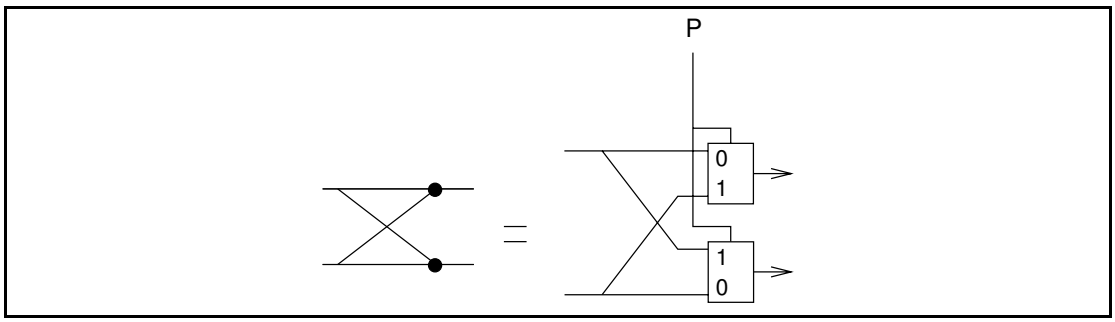


Figure 2.19: Butterfly implementation.

2.6.2.4 Second addition operation

The addition operation only adds a constant to the output of the permutation operation. The addition is applied modulo 79.

2.6.2.5 Register bank

The output of the adder addresses a bank of 79 registers. The registers are loaded with the synthesizer code words corresponding to the hop frequencies 0 to 78. Note that the upper half of the bank contains the even hop frequencies, whereas the lower half of the bank contains the odd hop frequencies.

2.6.3 Adapted hop selection kernel

The adapted hop selection kernel is based on the basic hop selection kernel defined in the preceding sections.

The inputs to the adapted hop selection kernel are the same as for the basic hop system kernel except that the input AFH_channel_map (defined in Link Manager Protocol [Part C] Section 5.2 on page 303) is used. The *AFH_channel_map* indicates which RF channels shall be *used* and which shall be *unused*. When hop sequence adaptation is enabled, the number of *used* RF channels may be reduced from 79 to some smaller value N. All devices shall be capable of operating on an adapted hop sequence (AHS) with $N_{min} \leq N \leq 79$, with any combination of *used* RF channels within the *AFH_channel_map* that meets this constraint. N_{min} is defined in Section 2.3.1 on page 75.

Adaptation of the hopping sequence is achieved through two additions to the basic channel hopping sequence according to Figure 2.16 on page 86:

- *Unused* RF channels are re-mapped uniformly onto *used* RF channels. That is, if the hop selection kernel of the basic system generates an *unused* RF channel, an alternative RF channel out of the set of *used* RF channels is selected pseudo-randomly.
- The *used* RF channel generated for the master-to-slave packet is also used for the immediately following slave-to-master packet (see Section 2.6.1 on page 82).

2.6.3.1 Channel re-mapping function

When the adapted hop selection kernel is selected, the basic hop selection kernel according to Figure 2.16 on page 86 is initially used to determine an RF channel. If this RF channel is *unused* according to the *AFH_channel_map*, the *unused* RF channel is re-mapped by the re-mapping function to one of the *used* RF channels. If the RF channel determined by the basic hop selection kernel is already in the set of *used* RF channels, no adjustment is made. The hop sequence of the (non-adapted) basic hop equals the sequence of the adapted selection kernel on all locations where *used* RF channels are generated by the basic hop. This property facilitates non-AFH slaves remaining synchronized while other slaves in the piconet are using the adapted hopping sequence.

A block diagram of the re-mapping mechanism is shown in Figure 2.20 on page 90. The re-mapping function is a post-processing step to the selection kernel from Figure 2.16 on page 86, denoted as ‘Hop selection of the basic hop’. The output f_k of the basic hop selection kernel is an RF channel number that ranges between 0 and 78. This RF channel will either be in the set of *used* RF channels or in the set of *unused* RF channels.

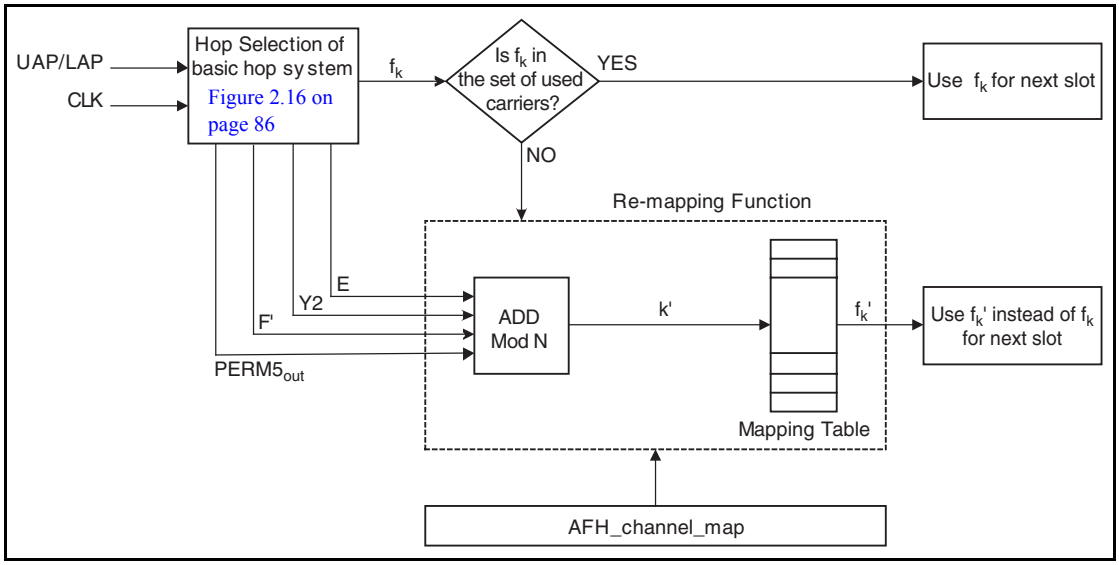


Figure 2.20: Block diagram of adaptive hop selection mechanism

When an unused RF channel is generated by the basic hop selection mechanism, it is re-mapped to the set of *used* RF channels as follows. A new index *k'* ∈ {0, 1,..., *N*-1} is calculated using some of the parameters from the basic hop selection kernel:

$$k' = (PERM5_{out} + E + F' + Y2) \bmod N$$

where *F'* is defined in Table 2.2 on page 91. The index *k'* is then used to select the re-mapped channel from a mapping table that contains all of the even *used* RF channels in ascending order followed by all the odd *used* RF channels in ascending order (i.e., the mapping table of Figure 2.16 on page 86 with all the *unused* RF channels removed).

2.6.4 Control word

In the following section X_{j-i}, $i<j$, will denote bits i, i+1,...,j of the bit vector X. By convention, X_0 is the least significant bit of the vector X.

The control word of the kernel is controlled by the overall control signals X, Y1, Y2, A to F, and F' as illustrated in Figure 2.16 on page 86 and Figure 2.20 on page 90. During paging and inquiry, the inputs A to E use the address values as given in the corresponding columns of Table 2.2 on page 91. In addition, the inputs X, Y1 and Y2 are used. The F and F' inputs are unused. The clock bits CLK_{6-2} (i.e., input X) specifies the phase within the length 32 sequence. CLK_1 (i.e., inputs Y1 and Y2) is used to select between TX and RX. The address inputs determine the sequence order within segments. The final mapping onto the hop frequencies is determined by the register contents.

During the **CONNECTION** state (see Section 8.5 on page 167), the inputs A, C and D shall be derived from the address bits being bit-wise XORed with the

clock bits as shown in the “Connection state” column of Table 2.2 on page 91 (the two most significant bits, MSBs, are XORed together, the two second MSBs are XORed together, etc.).

	Page scan / Interlaced Page Scan / Inquiry scan / Interlaced Inquiry Scan	**Page/Inquiry**	**Master/Slave page response and Inquiry response**	**Connection state**
X	CLKN_{16-12} / $(\mathrm{CLKN}_{16-12}+16) mod 32$ / Xir_{4-0} / $Xir_{4-0}+16) mod 32$	Xp_{4-0}/Xi_{4-0}	$Xprm_{4-0}$ / $Xprs_{4-0}$ / Xir_{4-0}	CLK_{6-2}
Y1	0	$\mathrm{CLKE}_1/\mathrm{CLKN}_1$	$\mathrm{CLKE}_1/\mathrm{CLKN}_1/1$	CLK_1
Y2	0	$32 \times \mathrm{CLKE}_1/$ $32 \times \mathrm{CLKN}_1$	$32 \times \mathrm{CLKE}_1$ / $32 \times \mathrm{CLKN}_1$ / 32×1	$32 \times \mathrm{CLK}_1$
A	A_{27-23}	A_{27-23}	A_{27-23}	$A_{27-23} \oplus \mathrm{CLK}_{25-21}$
B	A_{22-19}	A_{22-19}	A_{22-19}	A_{22-19}
C	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0}$	$A_{8,6,4,2,0} \oplus \mathrm{CLK}_{20-16}$
D	A_{18-10}	A_{18-10}	A_{18-10}	$A_{18-10} \oplus \mathrm{CLK}_{15-7}$
E	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$	$A_{13,11,9,7,5,3,1}$
F	0	0	0	$16 \times \mathrm{CLK}_{27-7} \bmod 79$
F’	n/a	n/a	n/a	$16 \times \mathrm{CLK}_{27-7} \bmod \mathrm{N}$

Table 2.2: Control for hop system.

The five X input bits vary depending on the current state of the device. In the **page scan** and **inquiry scan** substates, the native clock (CLKN) shall be used. In **CONNECTION** state the master clock (CLK) shall be used as input. The situation is somewhat more complicated for the other states.

2.6.4.1 Page scan and inquiry scan hopping sequences

When the sequence selection input is set to page scan, the Bluetooth device address of the scanning device shall be used as address input. When the sequence selection input is set to inquiry scan, the GIAC LAP and the four LSBs of the DCI (as A_{27-24}), shall be used as address input for the hopping sequence. For the transmitted access code and in the receiver correlator, the appropriate GIAC or DIAC shall be used. The application decides which inquiry access code to use depending on the purpose of the inquiry.

2.6.4.2 Page hopping sequence

When the sequence selection input is set to page, the paging device shall start using the **A**-train, i.e., $\{f(k-8), \ldots, f(k), \ldots, f(k+7)\}$, where $f(k)$ is the source's estimate of the current receiver frequency in the paged device. The index k is a function of all the inputs in Figure 2.16. There are 32 possible paging frequencies within each 1.28 second interval. Half of these frequencies belong to the **A**-train, the rest (i.e., $\{f(k+8), \ldots, f(k+15), f(k-16), \ldots, f(k-9)\}$) belong to the **B**-train. In order to achieve the -8 offset of the **A**-train, a constant of 24 shall be added to the clock bits (which is equivalent to -8 due to the modulo 32 operation). The **B**-train is obtained by setting the offset to 8. A cyclic shift of the order within the trains is also necessary in order to avoid a possible repetitive mismatch between the paging and scanning devices. Thus,

$$Xp = [\mathrm{CLKE}_{16-12} + k_{offset} + (\mathrm{CLKE}_{4-2,0} - \mathrm{CLKE}_{16-12}) \bmod 16] \bmod 32, \qquad \text{(EQ 2)}$$

where

$$k_{offset} = \begin{cases} 24 & \text{A-train,} \\ 8 & \text{B-train.} \end{cases} \qquad \text{(EQ 3)}$$

Alternatively, each switch between the **A**- and **B**-trains may be accomplished by adding 16 to the current value of k_{offset} (originally initialized with 24).

2.6.4.3 Slave page response hopping sequence

When the sequence selection input is set to *slave page response,* in order to eliminate the possibility of losing the link due to discrepancies of the native clock CLKN and the master's clock estimate CLKE, the four bits CLKN_{16-12} shall be frozen at their current value. The value shall be frozen at the content it has in the slot where the recipient's access code is detected. The native clock shall *not* be stopped; it is merely the values of the bits used for creating the X-input that are kept fixed for a while. A frozen value is denoted by an asterisk (*) in the discussion below.

For each response slot the paged device shall use an X-input value one larger (modulo 32) than in the preceding response slot. However, the first response shall be made with the X-input kept at the same value as it was when the access code was recognized. Let N be a counter starting at zero. Then, the X-input in the $(N+1)$-th response slot (the first response slot being the one immediately following the page slot now responding to) of the **slave response** substate is:

$$Xprs = [\mathrm{CLKN}^*_{16-12} + N] \bmod 32, \qquad \text{(EQ 4)}$$

The counter N shall be set to zero in the slot where the slave acknowledges the page (see Figure 8.3 on page 160 and Figure 8.4 on page 160). Then, the value of N shall be increased by one each time CLKN_1 is set to zero, which corresponds to the start of a master TX slot. The X-input shall be constructed this way until the first **FHS** packet is received *and* the immediately following response packet has been transmitted. After this the slave shall enter the **CONNECTION** state using the parameters received in the **FHS** packet.

2.6.4.4 Master page response hopping sequence

When the sequence selection input is set to *master page response*, the master shall freeze its estimated slave clock to the value that triggered a response from the paged device. It is equivalent to using the values of the clock estimate when receiving the slave response (since only CLKE_1 will differ from the corresponding page transmission). Thus, the values are frozen when the slave **ID** packet is received. In addition to the clock bits used, the current value of k_{offset} shall also be frozen. The master shall adjust its X-input in the same way the paged device does, i.e., by incrementing this value by one for each time CLKE_1 is set to zero. The first increment shall be done before sending the **FHS** packet to the paged device. Let N be a counter starting at one. The rule for forming the X-input is:

$$Xprm = [\mathrm{CLKE}^*_{16-12} + k_{offset}^* + (\mathrm{CLKE}^*_{4-2,0} - \mathrm{CLKE}^*_{16-12}) \bmod 16 + N] \bmod 32, \quad \text{(EQ 5)}$$

The value of N shall be increased each time CLKE_1 is set to zero, which corresponds to the start of a master TX slot.

2.6.4.5 Inquiry hopping sequence

When the sequence selection input is set to *inquiry*, the X-input is similar to that used in the *page hopping sequence*. Since no particular device is addressed, the native clock CLKN of the inquirer shall be used. Moreover, which of the two train offsets to start with is of no real concern in this state. Consequently,

$$Xi = [\mathrm{CLKN}_{16-12} + k_{offset} + (\mathrm{CLKN}_{4-2,0} - \mathrm{CLKN}_{16-12}) \bmod 16] \bmod 32, \quad \text{(EQ 6)}$$

where k_{offset} is defined by (EQ 3) on page 92. The initial choice of the offset is arbitrary.

The GIAC LAP and the four LSBs of the DCI (as A_{27-24}) shall be used as address input for the hopping sequence generator.

2.6.4.6 Inquiry response hopping sequence

The *inquiry response* hopping sequence is similar to the *slave page response* hopping sequence with respect to the X-input. The clock input shall not be frozen, thus the following equation apply:

$$Xir = [\mathrm{CLKN}_{16-12} + N] \mod 32, \quad \text{(EQ 7)}$$

Furthermore, the counter N is increased not on CLKN_1 basis, but rather after each **FHS** packet has been transmitted in response to the inquiry. There is no restriction on the initial value of N as it is independent of the corresponding value in the inquiring unit.

The GIAC LAP and the four LSBs of the DCI (as A_{27-24}) shall be used as address input for the hopping sequence generator. The other input bits to the generator shall be the same as for page response.

2.6.4.7 Basic and adapted channel hopping sequence

In the *basic* and *adapted channel hopping sequences*, the clock bits to use in the basic or adapted hopping sequence generation shall always be derived from the master clock, CLK. The address bits shall be derived from the Bluetooth device address of the master.

3 PHYSICAL LINKS

A physical link represents a baseband connection between devices. A physical link is always associated with exactly one physical channel. Physical links have common properties that apply to all logical transports on the physical link.

The common properties of physical links are:

- Power control (see Link Manager Protocol Section 4.1.3 on page 235)
- Link supervision (see Section 3.1 on page 95 and Link Manager Protocol Section 4.1.6 on page 242)
- Encryption (see Security [Part H] Section 4 on page 787 and Link Manager Protocol [Part C] Section 4.2.5 on page 257)
- Channel quality-driven data rate change (see Link Manager Protocol Section 4.1.7 on page 243)
- Multi-slot packet control (see Link Manager Protocol Section 4.1.10 on page 247)

3.1 LINK SUPERVISION

A connection can break down due to various reasons such as a device moving out of range, encountering severe interference or a power failure condition. Since this may happen without any prior warning, it is important to monitor the link on both the master and the slave side to avoid possible collisions when the logical transport address (see Section 4.2 on page 97) or parked member address (see Section 4.7.1 on page 105) is reassigned to another slave.

To be able to detect link loss, both the master and the slave shall use a link supervision timer, $T_{supervision}$. Upon reception of a valid packet header with one of the slave's addresses (see Section 4.2 on page 97) on the physical link, the timer shall be reset. If at any time in **CONNECTION** state, the timer reaches the *supervisionTO* value, the connection shall be considered disconnected. The same link supervision timer shall be used for SCO, eSCO, and ACL logical transports.

The timeout period, *supervisionTO*, is negotiated by the Link Manager. Its value shall be chosen so that the supervision timeout will be longer than hold and sniff periods. Link supervision of a parked slave shall be done by unparking and re-parking the slave.

4 LOGICAL TRANSPORTS

4.1 GENERAL

Between master and slave(s), different types of logical transports may be established. Five logical transports have been defined:

- Synchronous Connection-Oriented (SCO) logical transport
- Extended Synchronous Connection-Oriented (eSCO) logical transport
- Asynchronous Connection-Oriented (ACL) logical transport
- Active Slave Broadcast (ASB) logical transport
- Parked Slave Broadcast (PSB) logical transport

The synchronous logical transports are point-to-point logical transports between a master and a single slave in the piconet. The synchronous logical transports typically support time-bounded information like voice or general synchronous data. The master maintains the synchronous logical transports by using reserved slots at regular intervals. In addition to the reserved slots the eSCO logical transport may have a retransmission window after the reserved slots.

The ACL logical transport is also a point-to-point logical transport between the master and a slave. In the slots not reserved for synchronous logical transport(s), the master can establish an ACL logical transport on a per-slot basis to any slave, including the slave(s) already engaged in a synchronous logical transport.

The ASB logical transport is used by a master to communicate with active slaves. The PSB logical transport is used by a master to communicate with parked slaves.

4.2 LOGICAL TRANSPORT ADDRESS (LT_ADDR)

Each slave active in a piconet is assigned a primary 3-bit logical transport address (LT_ADDR). The all-zero LT_ADDR is reserved for broadcast messages. The master does not have an LT_ADDR. A master's timing relative to the slaves distinguishes it from the slaves. A secondary LT_ADDR is assigned to the slave for each eSCO logical transport in use in the piconet. Only eSCO traffic (i.e. NULL, POLL, and one of the EV packet types as negotiated at eSCO logical transport setup) may be sent on these LT_ADDRs. ACL traffic (including LMP) shall always be sent on the primary LT_ADDR. A slave shall only accept packets with matching primary or secondary LT_ADDR and broadcast packets. The LT_ADDR is carried in the packet header (see Section 6.4 on page 116). The LT_ADDR shall only be valid for as long as a slave is in the active mode. As soon as it is disconnected or parked, the slave shall lose all of its LT_ADDRs.

The primary LT_ADDR shall be assigned by the master to the slave when the slave is activated. This is either at connection establishment, at role switch, or when the slave is unparked. At connection establishment and at role switch, the primary LT_ADDR is carried in the **FHS** payload. When unparking, the primary LT_ADDR is carried in the unpark message.

4.3 SYNCHRONOUS LOGICAL TRANSPORTS

The first type of synchronous logical transport, the SCO logical transport is a symmetric, point-to-point link between the master and a specific slave. The SCO logical transport reserves slots and can therefore be considered as a circuit-switched connection between the master and the slave. The master may support up to three SCO links to the same slave or to different slaves. A slave may support up to three SCO links from the same master, or two SCO links if the links originate from different masters. SCO packets are never retransmitted.

The second type of synchronous logical transport, the eSCO logical transport, is a point-to-point logical transport between the master and a specific slave. eSCO logical transports may be symmetric or asymmetric. Similar to SCO, eSCO reserves slots and can therefore be considered a circuit-switched connection between the master and the slave. In addition to the reserved slots, eSCO supports a retransmission window immediately following the reserved slots. Together, the reserved slots and the retransmission window form the complete eSCO window.

4.4 ASYNCHRONOUS LOGICAL TRANSPORT

In the slots not reserved for synchronous logical transports, the master may exchange packets with any slave on a per-slot basis. The ACL logical transport provides a packet-switched connection between the master and all active slaves participating in the piconet. Both asynchronous and isochronous services are supported. Between a master and a slave only a single ACL logical transport shall exist. For most ACL packets, packet retransmission is applied to assure data integrity.

ACL packets not addressed to a specific slave are considered as broadcast packets and should be read by every slave. If there is no data to be sent on the ACL logical transport and no polling is required, no transmission is required.

4.5 TRANSMIT/RECEIVE ROUTINES

This section describes the way to use the packets as defined in Section 6 on page 109 in order to support the traffic on the ACL, SCO and eSCO logical transports. Both single-slave and multi-slave configurations are considered. In addition, the use of buffers for the TX and RX routines are described.

The TX and RX routines described in sections 4.5.1 and 4.5.2 are informative only.

4.5.1 TX Routine

The TX routine is carried out separately for each asynchronous and synchronous link. Figure 4.1 on page 99 shows the asynchronous and synchronous buffers as used in the TX routine. In this figure, only a single TX asynchronous buffer and a single TX synchronous buffer are shown. In the master, there is a separate TX asynchronous buffer for each slave. In addition there may be one or more TX synchronous buffers for each synchronous slave (different SCO or eSCO logical transports may either reuse the same TX synchronous buffer, or each have their own TX synchronous buffer). Each TX buffer consists of two FIFO registers: one **current** register which can be accessed and read by the Link Controller in order to compose the packets, and one **next** register that can be accessed by the Baseband Resource Manager to load new information. The positions of the switches S1 and S2 determine which register is current and which register is next; the switches are controlled by the Link Controller. The switches at the input and the output of the FIFO registers can never be connected to the same register simultaneously.

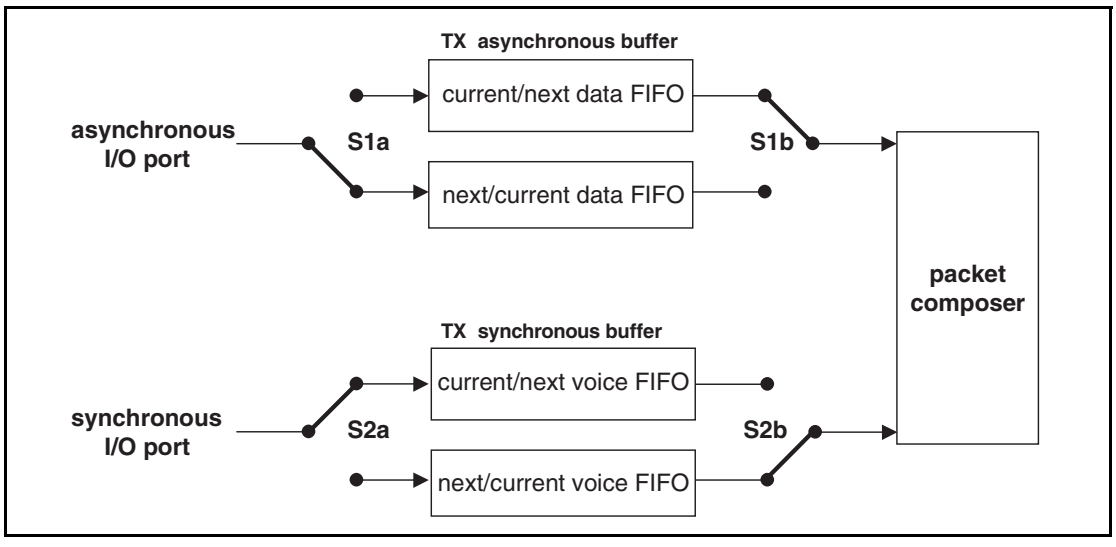


Figure 4.1: Functional diagram of TX buffering.

Of the packets common on the ACL and SCO logical transports (**NULL**, **POLL and DM1**) only the **DM1** packet carries a payload that is exchanged between the Link Controller and the Link Manager; this common packet makes use of the asynchronous buffer. All ACL packets make use of the asynchronous

buffer. All SCO and eSCO packets make use of the synchronous buffer except for the **DV** packet where the synchronous data part is handled by the synchronous buffer and the data part is handled by the asynchronous buffer. In the next sections, the operation for ACL traffic, SCO traffic, eSCO traffic, and combined data-voice traffic on the SCO logical transport are described.

4.5.1.1 ACL traffic

In the case of asynchronous data only the TX ACL buffer in Figure 4.1 on page 99 has to be considered. In this case, only packet types **DM** or **DH** are used, and these can have different lengths. The length is indicated in the payload header. The selection of **DM** or **DH** packets should depend on the quality of the link. See [Part C] Section 4.1.7 on page 243.

The default packet type in pure data traffic is **NULL** (see Section 6.5.1.2 on page 120). This means that, if there is no data to be sent (the data traffic is asynchronous, and therefore pauses occur in which no data is available) or no slaves need to be polled, **NULL** packets are sent instead – in order to send link control information to the other device (e.g. ACK/STOP information for received data). When no link control information is available (either no need to acknowledge and/or no need to stop the RX flow) no packet is sent at all.

The TX routine works as follows. The Baseband Resource Manager loads new data information in the register to which the switch S1a points. Next, it gives a command to the Link Controller, which forces the switch S1 to change (both S1a and S1b switch synchronously). When the payload needs to be sent, the packet composer reads the current register and, depending on the packet type, builds a payload which is appended to the channel access code and the header and is subsequently transmitted. In the response packet (which arrives in the following RX slot if it concerned a master transmission, or may be postponed until some later RX slot if it concerned a slave transmission), the result of the transmission is reported back. In case of an ACK, the switch S1 changes position; if a NAK (explicit or implicit) is received instead, the switch S1 will not change position. In that case, the same payload is retransmitted at the next TX occasion.

As long as the Baseband Resource Manager keeps loading the registers with new information, the Link Controller will automatically transmit the payload; in addition, retransmissions are performed automatically in case of errors. The Link Controller will send **NULL** or nothing when no new data is loaded. If no new data has been loaded in the **next** register, during the last transmission, the packet composer will be pointing to an empty register after the last transmission has been acknowledged and the **next** register becomes the **current** register. If new data is loaded in the **next** register, a **flush** command is required to switch the S1 switch to the proper register. As long as the Baseband Resource Manager keeps loading the data and type registers before each TX slot, the data is automatically processed by the Link Controller since the S1 switch is controlled by the ACK information received in response. However, if the traffic from the Baseband Resource Manager is interrupted once and a default packet is sent instead, a **flush** command is necessary to continue the flow in the Link Controller.

The **flush** command can also be used in case of time-bounded (isochronous) data. In case of a bad link, many retransmissions are necessary. In certain applications, the data is time-bounded: if a payload is retransmitted all the time because of link errors, it may become outdated, and the system might decide to continue with more recent data instead and skip the payload that does not come through. This is accomplished by the **flush** command as well. With the **flush**, the switch S1 is forced to change and the Link Controller is forced to consider the next data payload and overrules the ACK control. Any ACL type of packet can be used to send data or link control information to any other ACL slave.

4.5.1.2 SCO traffic

On the SCO logical transport only **HV** and **DV** packet types are used, See Section 6.5.2 on page 123. The synchronous port may continuously load the **next** register in the synchronous buffer. The S2 switches are changed according to the T_{sco} interval. This T_{sco} interval is negotiated between the master and the slave at the time the SCO logical transport is established.

For each new SCO slot, the packet composer reads the **current** register after which the S2 switch is changed. If the SCO slot has to be used to send control information with high priority concerning a control packet between the master and the SCO slave, or a control packet between the master and any other slave, the packet composer will discard the SCO information and use the control information instead. This control information shall be sent in a DM1 packet. Data or link control information may also be exchanged between the master and the SCO slave by using the **DV** or **DM1** packets.

4.5.1.3 Mixed data/voice traffic

In Section 6.5.2 on page 123, a **DV** packet has been defined that can support both data and voice simultaneously on a single SCO logical transport. When the TYPE is **DV**, the Link Controller reads the data register to fill the data field and the voice register to fill the voice field. Thereafter, the switch S2 is changed. However, the position of S1 depends on the result of the transmission as on the ACL logical transport: only if an ACK has been received will the S1 switch change its position. In each **DV** packet, the voice information is new, but the data information might be retransmitted if the previous transmission failed. If there is no data to be sent, the SCO logical transport will automatically change from **DV** packet type to the current **HV** packet type used before the mixed data/voice transmission. Note that a **flush** command is necessary when the data stream has been interrupted and new data has arrived.

Combined data-voice transmission can also be accomplished by using a separate ACL logical transport in addition to the SCO logical transport(s) if channel capacity permits this.

4.5.1.4 eSCO Traffic

On the eSCO logical transport only **EV**, **POLL** and **NULL** packet types are used, see Section 6.5.3 on page 124. The synchronous port may continuously load the next register in the synchronous buffer. The S2 switches are changed according to the T_{eSCO} interval. This T_{eSCO} interval is negotiated between the master and the slave at the time the eSCO logical transport is established.

For each new eSCO slot, the packet composer reads the current register after which the S2 switch is changed. If the eSCO slot has to be used to send control information with high priority concerning a control packet between the master and the eSCO slave, or an ACL packet between the master and any other slave, the packet composer will discard the eSCO information and use the control information instead. Control information to the eSCO slave is sent in a DM1 packet on the primary LT_ADDR.

4.5.1.5 Default packet types

On the ACL links, the default type is always **NULL** both for the master and the slave. This means that if no user information needs to be sent, either a **NULL** packet is sent if there is **ACK** or **STOP** information, or no packet is sent at all. The **NULL** packet can be used by the master to allocate the next slave-to-master slot to a certain slave (namely the one addressed). However, the slave is not forced to respond to the **NULL** packet from the master. If the master requires a response, it sends a **POLL** packet.

The SCO and eSCO packet types are negotiated at the LM level when the SCO or eSCO logical transport is established. The agreed packet type is also the default packet type for the reserved SCO or eSCO slots.

4.5.2 RX routine

The RX routine is carried out separately for the ACL logical transport and the synchronous logical transports. However, in contrast to the master TX asynchronous buffer, a single RX buffer is shared among all slaves. For the synchronous buffer, how the different synchronous logical transports are distinguished depends on whether extra synchronous buffers are required or not. Figure 4.2 on page 103 shows the asynchronous and synchronous buffers as used in the RX routine. The RX asynchronous buffer consists of two FIFO registers: one register that can be accessed and loaded by the Link Controller with the payload of the latest RX packet, and one register that can be accessed by the Baseband Resource Manager to read the previous payload. The RX synchronous buffer also consists of two FIFO registers: one register which is filled with newly arrived voice information, and one register which can be read by the voice processing unit.

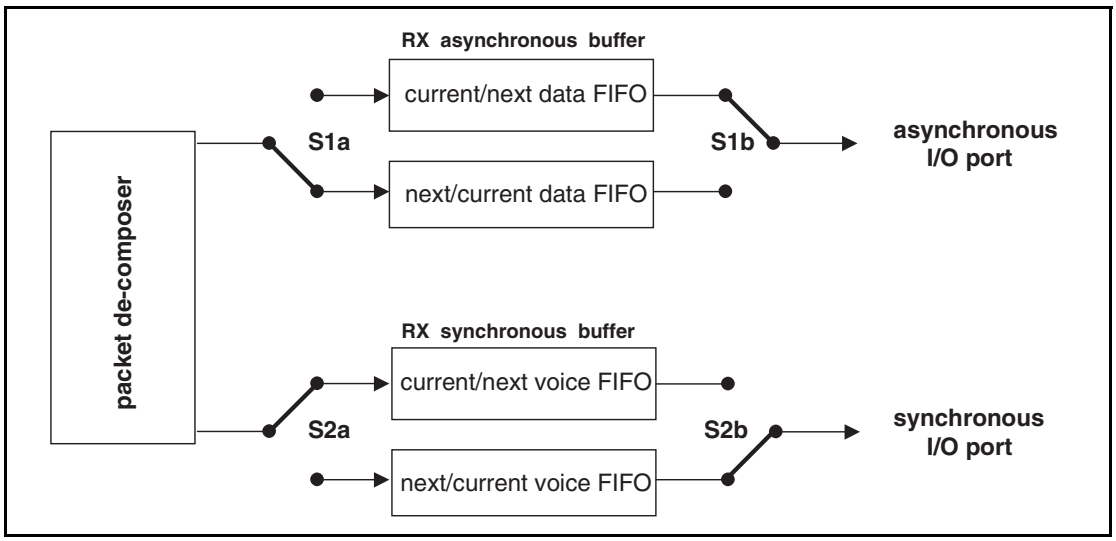


Figure 4.2: Functional diagram of RX buffering

Since the TYPE indication in the header (see Section 6.4.2 on page 116) of the received packet indicates whether the payload contains data and/or voice, the packet de-composer can automatically direct the traffic to the proper buffers. The switch S1 changes every time the Baseband Resource Manager reads the old register. If the next payload arrives before the RX register is emptied, a STOP indication is included in the packet header of the next TX packet that is returned. The STOP indication is removed again as soon as the RX register is emptied. The SEQN field is checked before a new ACL payload is stored into the asynchronous register (flush indication in LLID and broadcast messages influence the interpretation of the SEQN field see Section 7.6 on page 144).

The S2 switch is changed every T_{SCO} or T_{eSCO} for SCO and eSCO respectively. If, due to errors in the header, no new synchronous payload arrives, the switch still changes. The synchronous data processing unit then processes the synchronous data to account for the missing parts.

4.5.3 Flow control

Since the RX ACL buffer can be full while a new payload arrives, flow control is required. The header field FLOW in the return TX packet may use STOP or GO in order to control the transmission of new data.

4.5.3.1 Destination control

As long as data can not be received, a STOP indication shall be transmitted which is automatically inserted by the Link Controller into the header of the return packet. STOP shall be returned as long as the RX ACL buffer is not emptied by the Baseband Resource Manager. When new data can be accepted again, the GO indication shall be returned. GO shall be the default value. All packet types not including data can still be received. Voice communication for example is not affected by the flow control. Although a device can not receive new information, it may still continue to transmit information: the flow control shall be separate for each direction.

4.5.3.2 Source control

On the reception of a STOP signal, the Link Controller shall automatically switch to the default packet type. The ACL packet transmitted just before the reception of the STOP indication shall be kept until a GO signal is received. It may be retransmitted as soon as a GO indication is received. Only default packets shall be sent as long as the STOP indication is received. When no packet is received, GO shall be assumed implicitly. Note that the default packets contain link control information (in the header) for the receive direction (which may still be open) and may contain synchronous data (**HV** or **EV** packets). When a GO indication is received, the Link Controller may resume transmitting the data that is present in the TX ACL buffers.

In a multi-slave configuration, only the transmission to the slave that issued the STOP signal shall be stalled. This means that the master shall only stop transmission from the TX ACL buffer corresponding to the slave that momentarily cannot accept data.

4.6 ACTIVE SLAVE BROADCAST TRANSPORT

The active slave broadcast logical transport is used to transport L2CAP user traffic to all devices in the piconet that are currently connected to the piconet physical channel that is used by the ASB. There is no acknowledgement protocol and the traffic is uni-directional from the piconet master to the slaves. The ASB logical transport may only be used for L2CAP group traffic and shall never be used for L2CAP connection-oriented channels, L2CAP control signalling or LMP control signalling.

The ASB logical transport is unreliable. To improve reliability somewhat each packet is transmitted a number of times. An identical sequence number is used to assist with filtering retransmissions at the slave device.

The ASB logical transport is identified by the reserved, all-zero, LT_ADDR. Packets on the ASB logical transport may be sent by the master at any time.

4.7 PARKED SLAVE BROADCAST TRANSPORT

The parked slave broadcast logical transport is used for communication from the master to the slaves that are parked. The PSB logical transport is more complex than the other logical transports as it consists of a number of phases, each having a different purpose. These phases are the control information phase (used to carry the LMP logical link), the user information phase (used to carry the L2CAP logical link), and the access phase (carrying baseband signalling).

The PSB logical transport is identified by the reserved, all-zero, LT_ADDR.

4.7.1 Parked member address (PM_ADDR)

A slave in the **PARK** state can be identified by its BD_ADDR or by a dedicated parked member address (PM_ADDR). This latter address is an 8-bit member address that separates the parked slaves. The PM_ADDR shall only be valid as long as the slave is parked. When the slave is activated it shall be assigned an LT_ADDR but shall lose the PM_ADDR. The PM_ADDR is assigned to the slave by the master during the parking procedure (see [Part C] Section 4.5.2 on page 272).

The all-zero PM_ADDR shall be reserved for parked slaves that only use their BD_ADDR to be unparked.

4.7.2 Access request address (AR_ADDR)

The access request address (AR_ADDR) is used by the parked slave to determine the slave-to-master half slot in the access window where it is allowed to send access request messages, see also Section 8.9.6 on page 192. The AR_ADDR shall be assigned to the slave when it enters the **PARK** state and shall only be valid as long as the slave is parked. The AR_ADDR is not necessarily unique; i.e. different parked slaves may have the same AR_ADDR.

5 LOGICAL LINKS

Five logical links are defined:

- Link Control (LC)
- ACL Control (ACL-C)
- User Asynchronous/Isochronous (ACL-U)
- User Synchronous (SCO-S)
- User Extended Synchronous (eSCO-S)

The control logical links LC and ACL-C are used at the link control level and link manager level, respectively. The ACL-U logical link is used to carry either asynchronous or isochronous user information. The SCO-S, and eSCO-S logical links are used to carry synchronous user information. The LC logical link is carried in the packet header, all other logical links are carried in the packet payload. The ACL-C and ACL-U logical links are indicated in the logical link ID, LLID, field in the payload header. The SCO-S and eSCO-S logical links are carried by the synchronous logical transports only; the ACL-U link is normally carried by the ACL logical transport; however, they may also be carried by the data in the DV packet on the SCO logical transport. The ACL-C link may be carried either by the SCO or the ACL logical transport.

5.1 LINK CONTROL LOGICAL LINK (LC)

The LC control logical link shall be mapped onto the packet header. This logical link carries low level link control information like ARQ, flow control, and payload characterization. The LC logical link is carried in every packet except in the **ID** packet which does not have packet header.

5.2 ACL CONTROL LOGICAL LINK (ACL-C)

The ACL-C logical link shall carry control information exchanged between the link managers of the master and the slave(s). The ACL-C logical link shall use DM1 packets. The ACL-C logical link is indicated by the LLID code 11 in the payload header.

5.3 USER ASYNCHRONOUS/ISOCHRONOUS LOGICAL LINK (ACL-U)

The ACL-U logical link shall carry L2CAP asynchronous and isochronous user data. These messages may be transmitted in one or more baseband packets. For fragmented messages, the start packet shall use an LLID code of 10 in the payload header. Remaining continuation packets shall use LLID code 01. If there is no fragmentation, all packets shall use the LLID start code 10.

5.3.1 Pausing the ACL-U logical link

When paused by LM, the Link Controller transmits the current packet with ACL-U information, if any, until an ACK is received or, optionally, until an explicit NACK is received. While the ACL-U logical link is paused, the Link Controller shall not transmit any packets with ACL-U logical link information.

If the ACL-U was paused after an ACK, the next sequence number shall be used on the next packet. If the ACL-U was paused after a NAK, the same sequence number shall be used on the next packet and the un-acknowledged packet shall be transmitted once the ACL-U logical link is un-paused.

When the ACL-U logical link is un-paused by LM, the Link Controller may resume transmitting packets with ACL-U information.

5.4 USER SYNCHRONOUS DATA LOGICAL LINK (SCO-S)

The SCO-S logical link carries transparent synchronous user data. This logical link is carried over the synchronous logical transport SCO.

5.5 USER EXTENDED SYNCHRONOUS DATA LOGICAL LINK (eSCO-S)

The eSCO-S logical link also carries transparent synchronous user data. This logical link is carried over the extended synchronous logical transport eSCO.

5.6 LOGICAL LINK PRIORITIES

The ACL-C logical link shall have a higher priority than the ACL-U logical link when scheduling traffic on the shared ACL logical transport, except in the case when retransmissions of unacknowledged ACL packets shall be given priority over traffic on the ACL-C logical link. The ACL-C logical link should also have priority over traffic on the SCO-S and eSCO-S logical links but opportunities for interleaving the logical links should be taken.

6 PACKETS

Bluetooth devices shall use the packets as defined in the following sections.

6.1 GENERAL FORMAT

6.1.1 Basic Rate

The general packet format of Basic Rate packets is shown in Figure 6.1 on page 109. Each packet consists of 3 entities: the access code, the header, and the payload. In the figure, the number of bits per entity is indicated.

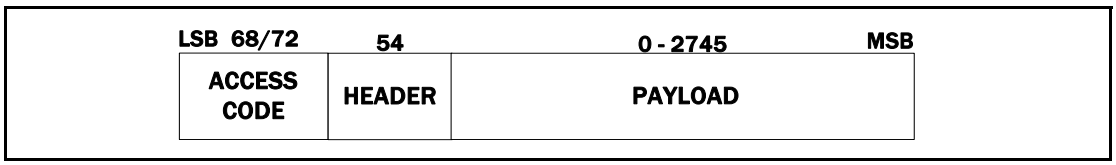


Figure 6.1: General Basic Rate packet format.

The access code is 72 or 68 bits and the header is 54 bits. The payload ranges from zero to a maximum of 2745 bits. Different packet types have been defined. Packet may consist of:

- the shortened access code only (see ID packet on page 116)
- the access code and the packet header
- the access code, the packet header and the payload.

6.1.2 Enhanced Data Rate

The general format of Enhanced Data Rate packets is shown in General enhanced data rate packet format. The access code and packet header are identical in format and modulation to Basic Rate packets. Enhanced Data Rate packets have a guard time and synchronization sequence following the packet header. Following the payload are two trailer symbols. The guard time, synchronization sequence and trailer are defined in section 6.6.

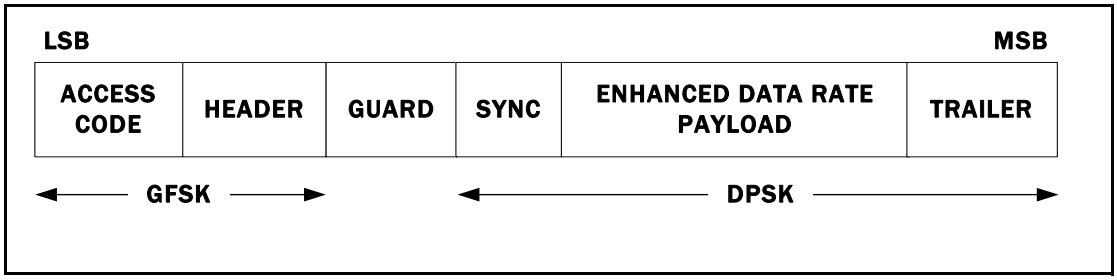


Figure 6.2: General enhanced data rate packet format

6.2 BIT ORDERING

The bit ordering when defining packets and messages in the *Baseband Specification*, follows the *Little Endian* format. The following rules apply:

- The *least significant bit* (LSB) corresponds to b_0;
- The LSB is the first bit sent over the air;
- In illustrations, the LSB is shown on the left side;

Furthermore, data fields generated internally at baseband level, such as the packet header fields and payload header length, shall be transmitted with the LSB first. For instance, a 3-bit parameter X=3 is sent as:

$$b_0 b_1 b_2 = 110$$

over the air where 1 is sent first and 0 is sent last.

6.3 ACCESS CODE

Every packet starts with an access code. If a packet header follows, the access code is 72 bits long, otherwise the access code is 68 bits long and is known as a shortened access code. The shortened access code does not contain a trailer. This access code is used for synchronization, DC offset compensation and identification. The access code identifies all packets exchanged on a physical channel: all packets sent in the same physical channel are preceded by the same access code. In the receiver of the device, a sliding correlator correlates against the access code and triggers when a threshold is exceeded. This trigger signal is used to determine the receive timing.

The shortened access code is used in paging, inquiry, and park. In this case, the access code itself is used as a signalling message and neither a header nor a payload is present.

The access code consists of a preamble, a sync word, and possibly a trailer, see Figure 6.3 on page 111. For details see Section 6.3.1 on page 111.

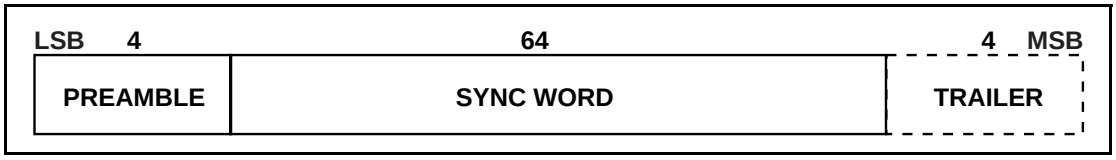


Figure 6.3: Access code format

6.3.1 Access code types

The different access code types use different Lower Address Parts (LAPs) to construct the sync word. The LAP field of the BD_ADDR is explained in Section 1.2 on page 66. A summary of the different access code types is in Table 6.1 on page 111.

Code type	LAP	Code length	Comments
CAC	Master	72	See also Section 1.3 on page 67
DAC	Paged device	68/72[1]	
GIAC	Reserved	68/72*	
DIAC	Dedicated	68/72*	

Table 6.1: Summary of access code types.

1. length 72 is only used in combination with FHS packets

The CAC consists of a **preamble**, **sync word**, and **trailer** and its total length is 72 bits. When used as self-contained messages without a header, the DAC and IAC do not include the trailer bits and are of length 68 bits.

6.3.2 Preamble

The preamble is a fixed zero-one pattern of 4 symbols used to facilitate DC compensation. The sequence is either 1010 or 0101, depending on whether the LSB of the following sync word is 1 or 0, respectively. The preamble is shown in Figure 6.4 on page 112.

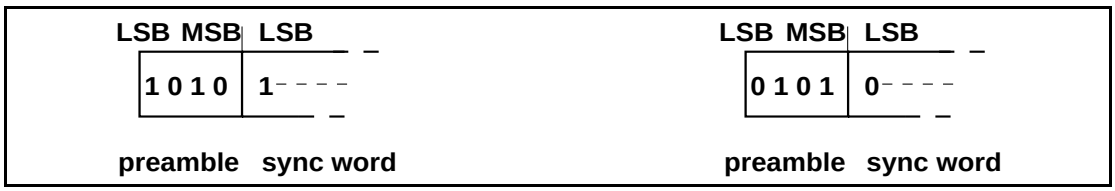


Figure 6.4: Preamble

6.3.3 Sync word

The sync word is a 64-bit code word derived from a 24 bit address (LAP); for the CAC the master's LAP is used; for the GIAC and the DIAC, reserved, dedicated LAPs are used; for the DAC, the slave LAP is used. The construction guarantees large Hamming distance between sync words based on different LAPs. In addition, the good auto correlation properties of the sync word improve timing acquisition.

6.3.3.1 Synchronization word definition

The sync words are based on a (64,30) expurgated block code with an overlay (bit-wise XOR) of a 64 bit full length pseudo-random noise (PN) sequence. The expurgated code guarantees large Hamming distance ($d_{min} = 14$) between sync words based on different addresses. The PN sequence improves the auto correlation properties of the access code. The following steps describe how the sync word shall be generated:

1. Generate information sequence;
2. XOR this with the "information covering" part of the PN overlay sequence;
3. Generate the codeword;
4. XOR the codeword with all 64 bits of the PN overlay sequence;

The information sequence is generated by appending 6 bits to the 24 bit LAP (step 1). The appended bits are 001101 if the MSB of the LAP equals 0. If the MSB of the LAP is 1 the appended bits are 110010. The LAP MSB together with the appended bits constitute a length-seven Barker sequence. The purpose of including a Barker sequence is to further improve the auto correlation properties. In step 2 the information is pre-scrambled by XORing it with the bits $p_{34} \ldots p_{63}$ of the PN sequence (defined in section 6.3.3.2 on page 115). After generating the codeword (step 3), the complete PN sequence is XORed to the

codeword (step 4). This step de-scrambles the information part of the codeword. At the same time the parity bits of the codeword are scrambled. Consequently, the original LAP and Barker sequence are ensured a role as a part of the access code sync word, and the cyclic properties of the underlying code is removed. The principle is depicted in Figure 6.5 on page 113

In the following discussion, binary sequences will be denoted by their corresponding D-transform (in which D^i represents a delay of *i* time units). Let $p'(D) = p'_0 + p'_1 D + \ldots + p'_{62} D^{62}$ be the 63 bit PN sequence, where p'_0 is the first bit (LSB) leaving the PRNG (see Figure 6.6 on page 115), and, p'_{62} is the last bit (MSB). To obtain 64 bits, an extra zero is appended at the *end* of this sequence (thus, $p'(D)$ is unchanged). For notational convenience, the reciprocal of this extended polynomial, $p(D) = D^{63} p'(1/D)$, will be used in the following discussion. This is the sequence $p'(D)$ in reverse order. We denote the 24 bit lower address part (LAP) of the Bluetooth device address by $a(D) = a_0 + a_1 D + \ldots + a_{23} D^{23}$ (a_0 is the LSB of the Bluetooth device address).

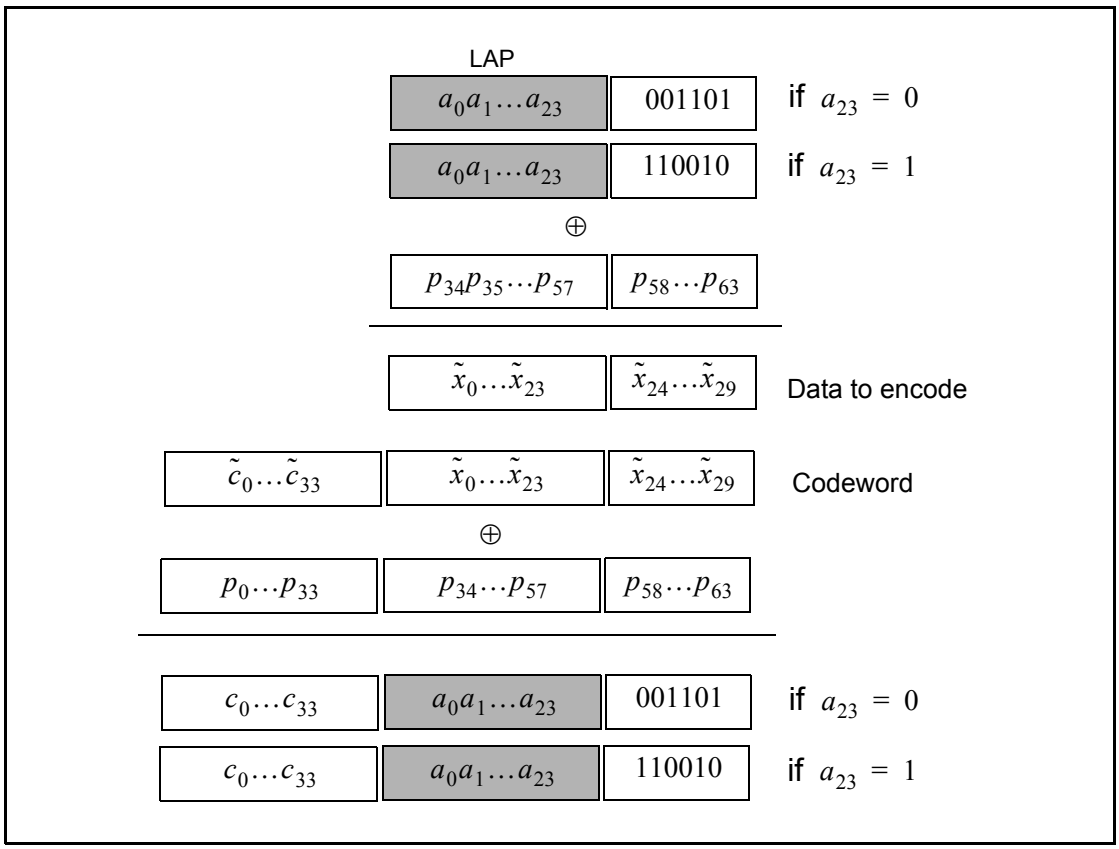


Figure 6.5: Construction of the sync word.

The (64,30) block code generator polynomial is denoted $g(D) = (1 + D)g'(D)$, where $g'(D)$ is the generator polynomial `157464165547` (octal notation) of a primitive binary (63,30) BCH code. Thus, in octal notation *g(D)* is

$$g(D) = 260534236651, \quad \text{(EQ 8)}$$

the left-most bit corresponds to the high-order (g_{34}) coefficient.The DC-free four bit sequences `0101` and `1010` can be written

$$\begin{cases} F_0(D) = D + D^3, \\ F_1(D) = 1 + D^2, \end{cases} \quad \text{(EQ 9)}$$

respectively. Furthermore,

$$\begin{cases} B_0(D) = D^2 + D^3 + D^5, \\ B_1(D) = 1 + D + D^4, \end{cases} \quad \text{(EQ 10)}$$

which are used to create the length seven Barker sequences. Then, the access code shall be generated by the following procedure:

1. Format the 30 information bits to encode:

$$x(D) = a(D) + D^{24}B_{a_{23}}(D).$$

2. Add the information covering part of the PN overlay sequence:

$$\tilde{x}(D) = x(D) + p_{34} + p_{35}D + \ldots + p_{63}D^{29}.$$

3. Generate parity bits of the (64,30) expurgated block code:[1]

$$\tilde{c}(D) = D^{34}\tilde{x}(D) \mod g(D).$$

4. Create the codeword:

$$\tilde{s}(D) = D^{34}\tilde{x}(D) + \tilde{c}(D).$$

5. Add the PN sequence:

$$s(D) = \tilde{s}(D) + p(D).$$

6. Append the (DC-free) preamble and trailer:

$$y(D) = F_{c_0}(D) + D^4 s(D) + D^{68}F_{a_{23}}(D).$$

1. $x(D)$ mod $y(D)$ denotes the remainder when $x(D)$ is divided by $y(D)$.

6.3.3.2 Pseudo-random noise sequence generation

To generate the PN sequence the primitive polynomial $h(D) = 1 + D + D^3 + D^4 + D^6$ shall be used. The LFSR and its starting state are shown in Figure 6.6 on page 115. The PN sequence generated (including the extra terminating zero) becomes (hexadecimal notation) 83848D96BBCC54FC. The LFSR output starts with the left-most bit of this PN sequence. This corresponds to $p'(D)$ of the previous section. Thus, using the reciprocal $p(D)$ as overlay gives the 64 bit sequence:

$$\boldsymbol{p} = 3F2A33DD69B121C1, \quad \text{(EQ 11)}$$

where the left-most bit is $p_0 = 0$ (there are two initial zeros in the binary representation of the hexadecimal digit 3), and $p_{63} = 1$ is the right-most bit.

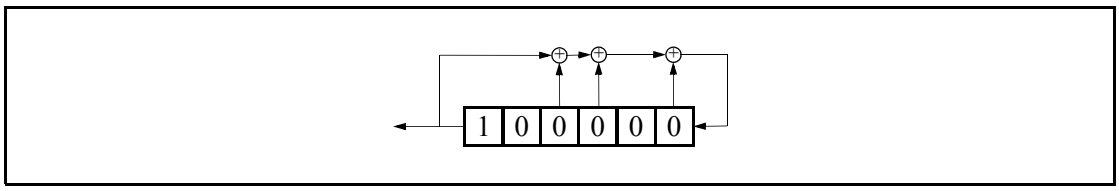


Figure 6.6: LFSR and the starting state to generate $p'(D)$.

6.3.4 Trailer

The trailer is appended to the sync word as soon as the packet header follows the access code. This is typically the case with the CAC, but the trailer is also used in the DAC and IAC when these codes are used in FHS packets exchanged during page response and inquiry response.

The trailer is a fixed zero-one pattern of four symbols. The trailer together with the three MSBs of the syncword form a 7-bit pattern of alternating ones and zeroes which may be used for extended DC compensation. The trailer sequence is either 1010 or 0101 depending on whether the MSB of the sync word is 0 or 1, respectively. The choice of trailer is illustrated in Figure 6.7 on page 115.

Figure 6.7: Trailer in CAC when MSB of sync word is 0 (a), and when MSB of sync word is 1 (b).

6.4 PACKET HEADER

The header contains link control (LC) information and consists of 6 fields:

- LT_ADDR: 3- bit logical transport address
- TYPE: 4-bit type code
- FLOW: 1-bit flow control
- ARQN: 1-bit acknowledge indication
- SEQN: 1-bit sequence number
- HEC: 8-bit header error check

The total header, including the HEC, consists of 18 bits, see Figure 6.8 on page 116, and is encoded with a rate 1/3 FEC (not shown but described in Section 7.4 on page 142) resulting in a 54-bit header. The LT_ADDR and TYPE fields shall be sent LSB first.

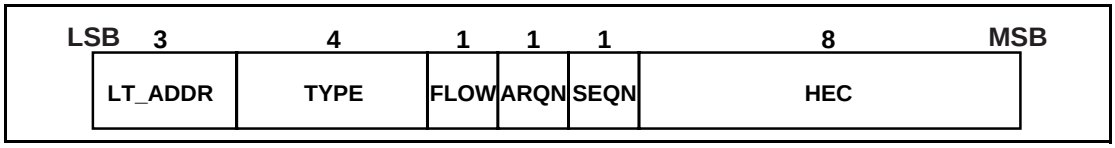


Figure 6.8: Header format.

6.4.1 LT_ADDR

The 3-bit LT_ADDR field contains the logical transport address for the packet (see Section 4.2 on page 97). This field indicates the destination slave for a packet in a master-to-slave transmission slot and indicates the source slave for a slave-to-master transmission slot.

6.4.2 TYPE

Sixteen different types of packets can be distinguished. The 4-bit TYPE code specifies which packet type is used. The interpretation of the TYPE code depends on the logical transport address in the packet. First, it shall be determined whether the packet is sent on an SCO logical transport, an eSCO logical transport, or an ACL logical transport. Second, it shall be determined whether Enhanced Data Rate has been enabled for the logical transport (ACL or eSCO) indicated by LT_ADDR. It can then be determined which type of SCO packet, eSCO packet, or ACL packet has been received. The TYPE code determines how many slots the current packet will occupy (see the slot occupancy column in Table 6.2 on page 119). This allows the non-addressed receivers to refrain from listening to the channel for the duration of the remaining slots. In Section 6.5 on page 118, each packet type is described in more detail.

6.4.3 FLOW

The FLOW bit is used for flow control of packets over the ACL logical transport. When the RX buffer for the ACL logical transport in the recipient is full, a STOP indication (FLOW=0) shall be returned to stop the other device from transmitting data temporarily. The STOP signal only affects ACL packets. Packets including only link control information (ID, POLL, and NULL packets), SCO packets or eSCO packets can still be received. When the RX buffer can accept data, a GO indication (FLOW=1) shall be returned. When no packet is received, or the received header is in error, a GO shall be assumed implicitly. In this case, the slave can receive a new packet with CRC although its RX buffer is still not emptied. The slave shall then return a NAK in response to this packet even if the packet passed the CRC check.

The FLOW bit is not used on the eSCO logical transport or the ACL-C logical link and shall be set to one on transmission and ignored upon receipt.

6.4.4 ARQN

The 1-bit acknowledgment indication ARQN is used to inform the source of a successful transfer of payload data with CRC, and can be positive acknowledge ACK or negative acknowledge NAK. See Section 7.6 on page 144 for initialization and usage of this bit.

6.4.5 SEQN

The SEQN bit provides a sequential numbering scheme to order the data packet stream. See section 7.6.2 on page 147 for initialization and usage of the SEQN bit. For broadcast packets, a modified sequencing method is used, see Section 7.6.5 on page 150.

6.4.6 HEC

Each header has a header-error-check to check the header integrity. The HEC is an 8-bit word (generation of the HEC is specified in Section 7.1.1 on page 138). Before generating the HEC, the HEC generator is initialized with an 8-bit value. For FHS packets sent in **master response** substate, the slave upper address part (UAP) shall be used. For FHS packets sent in **inquiry response**, the default check initialization (DCI, see Section 1.2.1 on page 66) shall be used. In all other cases, the UAP of the master device shall be used.

After the initialization, a HEC shall be calculated for the 10 header bits. Before checking the HEC, the receiver shall initialize the HEC check circuitry with the proper 8-bit UAP (or DCI). If the HEC does not check, the entire packet shall be discarded. More information can be found in Section 7.1 on page 138.

6.5 PACKET TYPES

The packets used on the piconet are related to the logical transports they are used in. Three logical transports with distinct packet types are defined (see Section 4 on page 97): the SCO logical transport, the eSCO logical transport, and the ACL logical transport. For each of these logical transports, 15 different packet types can be defined.

To indicate the different packets on a logical transport, the 4-bit TYPE code is used. The packet types are divided into four segments. The first segment is reserved for control packets. All control packets occupy a single time slot. The second segment is reserved for packets occupying a single time slot. The third segment is reserved for packets occupying three time slots. The fourth segment is reserved for packets occupying five time slots. The slot occupancy is reflected in the segmentation and can directly be derived from the type code. Table 6.2 on page 119 summarizes the packets defined for the SCO, eSCO, and ACL logical transport types.

All packet types with a payload shall use GFSK modulation unless specified otherwise in the following sections.

ACL logical transports Enhanced Data Rate packet types are explicitly selected via LMP using the *packet_type_table* (ptt) parameter. eSCO Enhanced Data Rate packet types are selected when the eSCO logical transport is established.

.

Segment	TYPE code $b_3b_2b_1b_0$	Slot occupancy	SCO logical transport (1 Mbps)	eSCO logical transport (1 Mbps)	eSCO logical transport (2-3 Mbps)	ACL logical transport (1 Mbps) ptt=0	ACL logical transport (2-3 Mbps) ptt=1
1	0000	1	NULL	NULL	NULL	NULL	NULL
	0001	1	POLL	POLL	POLL	POLL	POLL
	0010	1	FHS	reserved	reserved	FHS	FHS
	0011	1	DM1	reserved	reserved	DM1	DM1
2	0100	1	undefined	undefined	undefined	DH1	2-DH1
	0101	1	HV1	undefined	undefined	undefined	undefined
	0110	1	HV2	undefined	2-EV3	undefined	undefined
	0111	1	HV3	EV3	3-EV3	undefined	undefined
	1000	1	DV	undefined	undefined	undefined	3-DH1
	1001	1	undefined	undefined	undefined	AUX1	AUX1
3	1010	3	undefined	undefined	undefined	DM3	2-DH3
	1011	3	undefined	undefined	undefined	DH3	3-DH3
	1100	3	undefined	EV4	2-EV5	undefined	undefined
	1101	3	undefined	EV5	3-EV5	undefined	undefined
4	1110	5	undefined	undefined	undefined	DM5	2-DH5
	1111	5	undefined	undefined	undefined	DH5	3-DH5

Table 6.2: Packets defined for synchronous and asynchronous logical transport types.

6.5.1 Common packet types

There are five common kinds of packets. In addition to the types listed in segment 1 of the previous table, the ID packet is also a common packet type but is not listed in segment 1 because it does not have a packet header.

6.5.1.1 ID packet

The identity or ID packet consists of the device access code (DAC) or inquiry access code (IAC). It has a fixed length of 68 bits. It is a very robust packet since the receiver uses a bit correlator to match the received packet to the known bit sequence of the ID packet.

6.5.1.2 NULL packet

The NULL packet has no payload and consists of the channel access code and packet header only. Its total (fixed) length is 126 bits. The NULL packet may be used to return link information to the source regarding the success of the previous transmission (ARQN), or the status of the RX buffer (FLOW). The NULL packet may not have to be acknowledged.

6.5.1.3 POLL packet

The POLL packet is very similar to the NULL packet. It does not have a payload. In contrast to the NULL packet, it requires a confirmation from the recipient. It is not a part of the ARQ scheme. The POLL packet does not affect the ARQN and SEQN fields. Upon reception of a POLL packet the slave shall respond with a packet even when the slave does not have any information to send unless the slave has scatternet commitments in that timeslot. This return packet is an implicit acknowledgement of the POLL packet. This packet can be used by the master in a piconet to poll the slaves. Slaves shall not transmit the POLL packet.

6.5.1.4 FHS packet

The FHS packet is a special control packet containing, among other things, the Bluetooth device address and the clock of the sender. The payload contains 144 information bits plus a 16-bit CRC code. The payload is coded with a rate 2/3 FEC with a gross payload length of 240 bits.

Figure 6.9 on page 120 illustrates the format and contents of the FHS payload. The payload consists of eleven fields. The FHS packet is used in page master response, inquiry response and in role switch.

The FHS packet contains real-time clock information. This clock information shall be updated before each retransmission. The retransmission of the FHS payload is different than retransmissions of ordinary data payloads where the same payload is used for each retransmission. The FHS packet is used for frequency hop synchronization before the piconet channel has been established, or when an existing piconet changes to a new piconet.

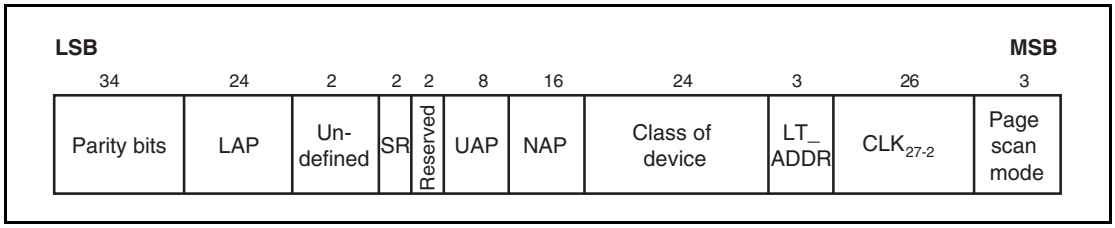


Figure 6.9: Format of the FHS payload.

Each field is described in more detail below:

Parity bits	This 34-bit field contains the parity bits that form the first part of the sync word of the access code of the device that sends the FHS packet. These bits are derived from the LAP as described in Section 1.2 on page 66.
LAP	This 24-bit field shall contain the lower address part of the device that sends the FHS packet.
Undefined	This 2-bit field is reserved for future use and shall be set to zero.
SR	This 2-bit field is the scan repetition field and indicates the interval between two consecutive page scan windows, see also Table 6.4 and Table 8.1 on page 155
Reserved	This 2-bit field shall be set to 10.
UAP	This 8-bit field shall contain the upper address part of the device that sends the FHS packet.
NAP	This 16-bit field shall contain the non-significant address part of the device that sends the FHS packet (see also Section 1.2 on page 66 for LAP, UAP, and NAP).
Class of device	This 24-bit field shall contain the class of device of the device that sends the FHS packet. The field is defined in Bluetooth Assigned Numbers (https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers).
LT_ADDR	This 3-bit field shall contain the logical transport address the recipient shall use if the FHS packet is used at connection setup or role switch. A slave responding to a master or a device responding to an inquiry request message shall include an all-zero LT_ADDR field if it sends the FHS packet.
CLK_{27-2}	This 26-bit field shall contain the value of the native clock of the device that sends the FHS packet, sampled at the beginning of the transmission of the access code of this FHS packet. This clock value has a resolution of 1.25ms (two-slot interval). For each new transmission, this field is updated so that it accurately reflects the real-time clock value.
Page scan mode	This 3-bit field shall indicate which scan mode is used by default by the sender of the FHS packet. The interpretation of the page scan mode is illustrated in Table 6.5.

Table 6.3: Description of the FHS payload

The device sending the FHS shall set the SR bits according to Table 6.4.

SR bit format b_1b_0	SR mode
00	R0
01	R1
10	R2
11	reserved

Table 6.4: Contents of SR field

The device sending the FHS shall set the page scan mode bits according to Table 6.5.

Bit format $b_2b_1b_0$	Page scan mode
000	Mandatory scan mode
001	Reserved for future use
010	Reserved for future use
011	Reserved for future use
100	Reserved for future use
101	Reserved for future use
110	Reserved for future use
111	Reserved for future use

Table 6.5: Contents of page scan mode field

The LAP, UAP, and NAP together form the 48-bit Bluetooth Device Address of the device that sends the FHS packet. Using the parity bits and the LAP, the recipient can directly construct the channel access code of the sender of the FHS packet.

When initializing the HEC and CRC for the FHS packet of inquiry response, the UAP shall be the DCI.

6.5.1.5 DM1 packet

DM1 is part of segment 1 in order to support control messages in any logical transport that allows the DM1 packet (see Table 6.2 on page 119). However, it may also carry regular user data. Since the DM1 packet can be regarded as an ACL packet, it will be discussed in Section 6.5.4 on page 126.

6.5.2 SCO packets

HV and DV packets are used on the synchronous SCO logical transport. The HV packets do not include a CRC and shall not be retransmitted. DV packets include a CRC on the data section, but not on the synchronous data section. The data section of DV packets shall be retransmitted. SCO packets may be routed to the synchronous I/O port. Four packets are allowed on the SCO logical transport: HV1, HV2, HV3 and DV. These packets are typically used for 64kb/s speech transmission but may be used for transparent synchronous data.

6.5.2.1 HV1 packet

The **HV1** packet has 10 information bytes. The bytes are protected with a rate 1/3 FEC. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

6.5.2.2 HV2 packet

The **HV2** packet has 20 information bytes. The bytes are protected with a rate 2/3 FEC. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

6.5.2.3 HV3 packet

The **HV3** packet has 30 information bytes. The bytes are not protected by FEC. No CRC is present. The payload length is fixed at 240 bits. There is no payload header present.

6.5.2.4 DV packet

The **DV** packet is a combined data - voice packet.The DV packet shall only be used in place of an HV1 packet. The payload is divided into a voice field of 80 bits and a data field containing up to 150 bits, see Figure 6.10. The voice field is not protected by FEC. The data field has between 1 and 10 information bytes (including the 1-byte payload header) and includes a 16-bit CRC. The data field is encoded with a rate 2/3 FEC. Since the **DV** packet has to be sent at regular intervals due to its synchronous contents, it is listed under the SCO packet types. The voice and data fields shall be treated separately. The voice field shall be handled in the same way as normal SCO data and shall never be retransmitted; that is, the voice field is always new. The data field is checked for errors and shall be retransmitted if necessary. When the asynchronous data field in the DV packet has not be acknowledged before the SCO logical transport is terminated, the asynchronous data field shall be retransmitted in a DM1 packet.

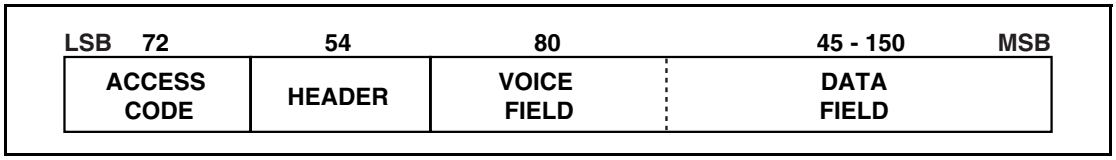


Figure 6.10: DV packet format

6.5.3 eSCO packets

EV packets are used on the synchronous eSCO logical transport. The packets include a CRC and retransmission may be applied if no acknowledgement of proper reception is received within the retransmission window. eSCO packets may be routed to the synchronous I/O port. Three eSCO packet types (EV3, EV4, EV5) are defined for Basic Rate operation and four additional eSCO packet types (2-EV3, 3-EV3, 2-EV5, 3-EV5) for Enhanced Data Rate operation. The eSCO packets may be used for 64kb/s speech transmission as well as transparent data at 64kb/s and other rates.

6.5.3.1 EV3 packet

The **EV3** packet has between 1 and 30 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The EV3 packet may cover up to a single time slot. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

6.5.3.2 EV4 packet

The **EV4** packet has between 1 and 120 information bytes plus a 16-bit CRC code. The EV4 packet may cover to up three time slots. The information plus CRC bits are coded with a rate 2/3 FEC. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

6.5.3.3 EV5 packet

The **EV5** packet has between 1 and 180 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The EV5 packet may cover up to three time slots. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

6.5.3.4 2-EV3 packet

The **2-EV3** packet is similar to the EV3 packet except that the payload is modulated using $\pi/4$-DQPSK. It has between 1 and 60 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 2-EV3 packet covers a single time slot. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

6.5.3.5 2-EV5 packet

The **2-EV5** packet is similar to the EV5 packet except that the payload is modulated using π/4-DQPSK. It has between 1 and 360 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 2-EV5 packet may cover up to three time slots. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

6.5.3.6 3-EV3 packet

The **3-EV3** packet is similar to the EV3 packet except that the payload is modulated using 8DPSK. It has between 1 and 90 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 3-EV3 packet covers a single time slot. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

6.5.3.7 3-EV5 packet

The **3-EV5** packet is similar to the EV5 packet except that the payload is modulated using 8DPSK. It has between 1 and 540 information bytes plus a 16-bit CRC code. The bytes are not protected by FEC. The 3-EV5 packet may cover up to three time slots. There is no payload header present. The payload length is set during the LMP eSCO setup and remains fixed until the link is removed or re-negotiated.

6.5.4 ACL packets

ACL packets are used on the asynchronous logical transport. The information carried may be user data or control data.

Seven packet types are defined for Basic Rate operation: DM1, DH1, DM3, DH3, DM5, DH5 and AUX1. Six additional packets are defined for Enhanced Data Rate operation: 2-DH1, 3-DH1, 2-DH3, 3-DH3, 2-DH5 and 3-DH5.

6.5.4.1 DM1 packet

The DM1 packet carries data information only. The payload has between 1 and 18 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. The DM1 packet occupies a single time slot. The information plus CRC bits are coded with a rate 2/3 FEC. The payload header in the DM1 packet is 1 byte long, see Figure 6.12 on page 130. The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

6.5.4.2 DH1 packet

This packet is similar to the DM1 packet, except that the information in the payload is not FEC encoded. As a result, the DH1 packet has between 1 and 28 information bytes (including the 1-byte payload header) plus a 16-bit CRC code. The DH1 packet occupies a single time slot.

6.5.4.3 DM3 packet

The DM3 packet may occupy up to three time slots. The payload has between 2 and 123 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The information plus CRC bits are coded with a rate 2/3 FEC. The payload header in the DM3 packet is 2 bytes long, see Figure 6.13 on page 131. The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

6.5.4.4 DH3 packet

This packet is similar to the DM3 packet, except that the information in the payload is not FEC encoded. As a result, the DH3 packet has between 2 and 185 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The DH3 packet may occupy up to three time slots.

6.5.4.5 DM5 packet

The DM5 packet may occupy up to five time slots. The payload has between 2 and 226 information bytes (including the 2-byte payload header) plus a 16-bit

CRC code. The payload header in the DM5 packet is 2 bytes long. The information plus CRC bits are coded with a rate 2/3 FEC. The length indicator in the payload header specifies the number of user bytes (excluding payload header and the CRC code).

6.5.4.6 DH5 packet

This packet is similar to the DM5 packet, except that the information in the payload is not FEC encoded. As a result, the DH5 packet has between 2 and 341 information bytes (including the2-byte payload header) plus a 16-bit CRC code. The DH5 packet may occupy up to five time slots.

6.5.4.7 AUX1 packet

This packet resembles a DH1 packet but has no CRC code. The AUX1 packet has between 1 and 30 information bytes (including the 1-byte payload header). The AUX1 packet occupies a single time slot. The AUX1 packet shall not be used for the ACL-U or ACL-C logical links. An AUX1 packet may be discarded.

6.5.4.8 2-DH1 packet

This packet is similar to the DH1 packet except that the payload is modulated using π/4-DQPSK. The 2-DH1 packet has between 2 and 56 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 2-DH1 packet occupies a single time slot.

6.5.4.9 2-DH3 packet

This packet is similar to the DH3 packet except that the payload is modulated using π/4-DQPSK. The 2-DH3 packet has between 2 and 369 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 2-DH3 packet may occupy up to three time slots.

6.5.4.10 2-DH5 packet

This packet is similar to the DH5 packet except that the payload is modulated using π/4-DQPSK. The 2-DH5 packet has between 2 and 681 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 2-DH5 packet may occupy up to five time slots.

6.5.4.11 3-DH1 packet

This packet is similar to the DH1 packet except that the payload is modulated using 8DPSK. The 3-DH1 packet has between 2 and 85 information bytes

(including the 2-byte payload header) plus a 16-bit CRC code. The 3-DH1 packet occupies a single time slot.

6.5.4.12 3-DH3 packet

This packet is similar to the DH3 packet except that the payload is modulated using 8DPSK. The 3-DH3 packet has between 2 and 554 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 3-DH3 packet may occupy up to three time slots.

6.5.4.13 3-DH5 packet

This packet is similar to the DH5 packet except that the payload is modulated using 8DPSK. The 3-DH5 packet has between 2 and 1023 information bytes (including the 2-byte payload header) plus a 16-bit CRC code. The 3-DH5 packet may occupy up to five time slots.

6.6 PAYLOAD FORMAT

In the payload, two fields are distinguished: the synchronous data field and the asynchronous data field. The ACL packets only have the asynchronous data field and the SCO and eSCOpackets only have the synchronous data field – with the exception of the DV packets which have both.

6.6.1 Synchronous data field

In SCO, which is only supported in Basic Rate mode, the synchronous data field has a fixed length and consists only of the synchronous data body portion. No payload header is present.

In Basic Rate eSCO, the synchronous data field consists of two segments: a synchronous data body and a CRC code. No payload header is present.

In Enhanced Data Rate eSCO, the synchronous data field consists of five segments: a guard time, a synchronization sequence, a synchronous data body, a CRC code and a trailer. No payload header is present.

1. Enhanced Data Rate Guard Time

For Enhanced Data Rate packets the guard time is defined as the period starting at the end of the last GFSK symbol of the header and ending at the start of the reference symbol of the synchronization sequence. The length of the guard time shall be between 4.75 µsec and 5.25 µsec.

2. Enhanced Data Rate Synchronization Sequence

For Enhanced Data Rate packets the symbol timing at the start of the synchronization sequence shall be within ±¼ µsec of the symbol timing of the last GFSK symbol of the packet header. The length of the synchronization sequence is 11 µsec (11 DPSK symbols) and consists of a reference symbol (with arbitrary phase) followed by ten DPSK symbols.

The phase changes between the DPSK symbols (shown in Synchronization sequence) shall be

$$\{\varphi_1, \varphi_2, \varphi_3, \varphi_4, \varphi_5, \varphi_6, \varphi_7, \varphi_8, \varphi_9, \varphi_{10}\} = \{3\pi/4, -3\pi/4, 3\pi/4, -3\pi/4, 3\pi/4, -3\pi/4, -3\pi/4, 3\pi/4, 3\pi/4, 3\pi/4\} \quad \text{(EQ 12)}$$

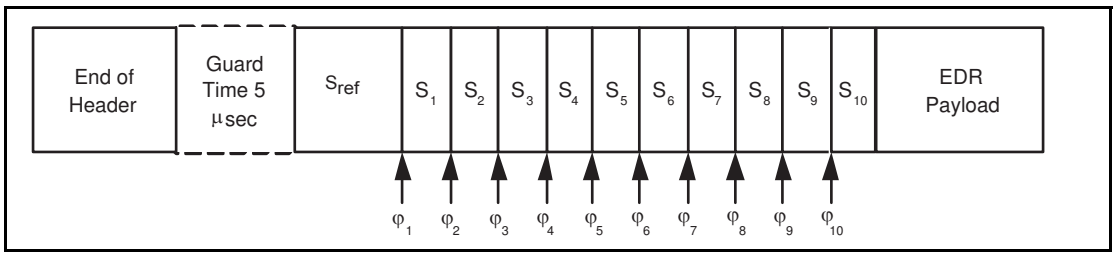


Figure 6.11: Synchronization sequence

Sref is the reference symbol. φ_1 is the phase change between the reference symbol and the first DPSK symbol S_1. φ_k is the phase change between the k-1th symbol S_{k-1} and the kth symbol S_k

Note: the synchronization sequence may be generated using the modulator by pre-pending the data with bits that generate the synchronization sequence.

For π/4-DQPSK, the bit sequence used to generate the synchronization sequence is 0,1,1,1,0,1,1,1,0,1,1,1,1,1,0,1,0,1,0,1.

For 8DPSK, the bit sequence used to generate the synchronization sequence is 0,1,0,1,1,1,0,1,0,1,1,1,0,1,0,1,1,1,1,1,1,1,0,1,0,0,1,0,0,1,0.

3. Synchronous data body

For HV and DV packets, the synchronous data body length is fixed. For EV packets, the synchronous data body length is negotiated during the LMP eSCO setup. Once negotiated, the synchronous data body length remains constant unless re-negotiated. The synchronous data body length may be different for each direction of the eSCO logical transport.

4. CRC code

The 16-bit CRC in the payload is generated as specified in Section 7.1 on page 138. The 8-bit UAP of the master is used to initialize the CRC generator.

Only the Synchronous data body segment is used to generate the CRC code.

5. Enhanced Data Rate Trailer

For Enhanced Data Rate packets, two trailer symbols shall be added to the end of the payload. The trailer bits shall be all zero, i.e. {00, 00} for the π/4-DQPSK and {000, 000} for the 8DPSK.

6.6.2 Asynchronous data field

Basic rate ACL packets have an asynchronous data field consisting of two or three segments: a payload header, a payload body, and possibly a CRC code (the AUX1 packet does not carry a CRC code).

Enhanced Data Rate ACL packets have an asynchronous data field consisting of six segments: a guard time, a synchronization sequence, a payload header, a payload body, a CRC and a trailer.

1. Enhanced Data Rate Guard time

This is the same as defined for the Synchronous data field in section 6.6.1.

2. Enhanced Data Rate Synchronization sequence

This is the same as defined for the Synchronous data field in section 6.6.1.

3. Payload header

The payload header is one or two bytes long. Basic rate packets in segments one and two have a 1-byte payload header; Basic Rate packets in segments three and four and all Enhanced Data Rate packets have a 2-byte payload header. The payload header specifies the logical link (2-bit LLID indication), controls the flow on the logical channels (1-bit FLOW indication), and has a payload length indicator (5 bits and 10 bits for 1-byte and 2-byte payload headers, respectively). In the case of a 2-byte payload header, the length indicator is extended by five bits into the next byte. The remaining three bits of the second byte are reserved for future use and shall be set to zero. The formats of the 1-byte and 2-byte payload headers are shown in Figure 6.12 on page 130 and Figure 6.13 on page 131.

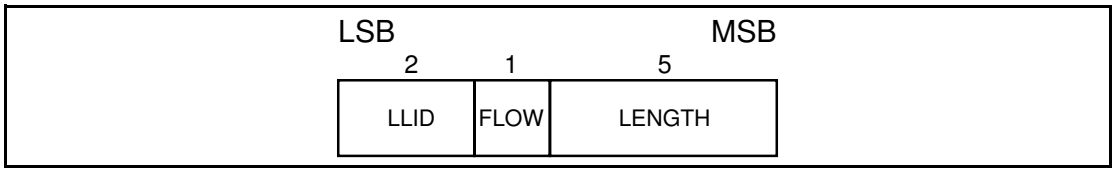


Figure 6.12: Payload header format for Basic Rate single-slot ACL packets.

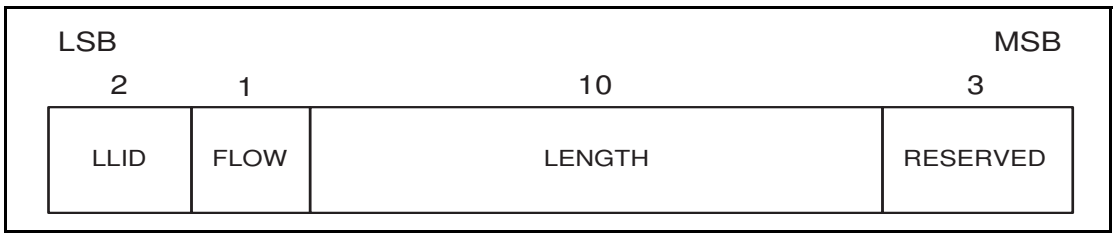


Figure 6.13: Payload header format for multi-slot ACL packets and all EDR ACL packets.

The LLID field shall be transmitted first, the length field last. In Table 6.6 on page 132, more details about the contents of the LLID field are listed.

LLID code b_1b_0	Logical Link	Information
00	NA	undefined
01	ACL-U	Continuation fragment of an L2CAP message
10	ACL-U	Start of an L2CAP message or no fragmentation
11	ACL-C	LMP message

Table 6.6: Logical link LLID field contents

An L2CAP message may be fragmented into several packets. Code 10 shall be used for an ACL-U packet carrying the first fragment of such a message; code 01 shall be used for continuing fragments. If there is no fragmentation, code 10 shall be used for every packet. Code 11 shall be used for LMP messages. Code 00 is reserved for future use.

The flow indicator in the payload is used to control the flow at the L2CAP level. It is used to control the flow per logical link. FLOW=1 means flow-on (GO) and FLOW=0 means flow-off (STOP). After a new connection has been established the flow indicator shall be set to GO. When a device receives a payload header with the flow bit set to STOP, it shall stop the transmission of ACL packets before an additional amount of payload data is sent. This amount is defined as the flow control lag, expressed as a number of bytes. The shorter the flow control lag, the less buffering the other device must dedicate to this function. The flow control lag shall not exceed 1792 bytes (7×256 bytes). In order to allow devices to optimize the selection of packet length and buffer space, the flow control lag of a given implementation shall be provided in the LMP_features_res message.

If a packet containing the payload flow bit of STOP is received, with a valid packet header but bad payload, the payload flow control bit shall be ignored. The baseband ACK contained in the packet header will be received and a further ACL packet may be transmitted. Each occurrence of this situation allows a further ACL packet to be sent in spite of the flow control request being sent via the payload header flow control bit. It is recommended that devices that use the payload header flow bit should ensure that no further ACL packets are sent until the payload flow bit has been correctly received. This can be accomplished by simultaneously turning on the flow bit in the packet header and keeping it on until an ACK is received back (ARQN=1). This will typically be only one round trip time. Since they lack a payload CRC, AUX1 packets should not be used with a payload flow bit of STOP.

The Baseband Resource Manager is responsible for setting and processing the flow bit in the payload header. Real-time flow control shall be carried out at the packet level by the link controller via the flow bit in the packet header (see Section 6.4.3 on page 117). With the payload flow bit, traffic from the remote end can be controlled. It is allowed to generate and send an ACL packet with payload length zero irrespective of flow status. L2CAP start-fragment and continue-fragment indications (LLID=10 and LLID=01) also retain their meaning when the payload length is equal to zero (i.e. an empty start-fragment shall not be sent in the middle of an on-going ACL-U packet transmission). It is always safe to send an ACL packet with length=0 and LLID=01. The payload flow bit has its own meaning for each logical link (ACL-U or ACL-C), Table 6.7 on page 133. On the ACL-C logical link, no flow control is applied and the payload FLOW bit shall always be set to one.

LLID code b_1b_0	Usage and semantics of the ACL payload header FLOW bit
00	Not defined, reserved for future use.
01 or 10	Flow control of the ACL-U channel (L2CAP messages)
11	Always set FLOW=1 on transmission and ignore the bit on reception

Table 6.7: Use of payload header flow bit on the logical links.

The length indicator shall be set to the number of bytes (i.e. 8-bit words) in the payload excluding the payload header and the CRC code; i.e. the payload body only. With reference to Figure 6.12 and Figure 6.13, the MSB of the length field in a 1-byte header is the last (right-most) bit in the payload header; the MSB of the length field in a 2-byte header is the fourth bit (from left) of the second byte in the payload header.

4. Payload body

The payload body includes the user information and determines the effective user throughput. The length of the payload body is indicated in the length field of the payload header.

5. CRC code generation

The 16-bit cyclic redundancy check code in the payload is generated as specified in Section 7.1 on page 138. Before determining the CRC code, an 8-bit value is used to initialize the CRC generator. For the CRC code in the FHS packets sent in **master response** substate, the UAP of the slave is used. For the FHS packet sent in **inquiry response** substate, the DCI (see Section 1.2.1 on page 66) is used. For all other packets, the UAP of the master is used.

Only the Payload header and Payload body segments are used to generate the CRC code.

6. Enhanced Data Rate Trailer

This is the same as defined for the Synchronous data field in section 6.6.1.

6.7 PACKET SUMMARY

A summary of the packets and their characteristics is shown in Table 6.8, Table 6.9 and Table 6.10. The payload represents the packet payload excluding FEC, CRC, and payload header.

Type	Payload (bytes)	FEC	CRC	Symmetric Max. Rate	Asymmetric Max. Rate
ID	na	na	na	na	na
NULL	na	na	na	na	na
POLL	na	na	na	na	na
FHS	18	2/3	yes	na	na

Table 6.8: Link control packets

Type	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kb/s)	Asymmetric Max. Rate (kb/s)	
						Forward	Reverse
DM1	1	0-17	2/3	yes	108.8	108.8	108.8
DH1	1	0-27	no	yes	172.8	172.8	172.8
DM3	2	0-121	2/3	yes	258.1	387.2	54.4
DH3	2	0-183	no	yes	390.4	585.6	86.4
DM5	2	0-224	2/3	yes	286.7	477.8	36.3
DH5	2	0-339	no	yes	433.9	723.2	57.6
AUX1	1	0-29	no	no	185.6	185.6	185.6
2-DH1	2	0-54	no	yes	345.6	345.6	345.6
2-DH3	2	0-367	no	yes	782.9	1174.4	172.8
2-DH5	2	0-679	no	yes	869.7	1448.5	115.2
3-DH1	2	0-83	no	yes	531.2	531.2	531.2
3-DH3	2	0-552	no	yes	1177.6	1766.4	235.6
3-DH5	2	0-1021	no	yes	1306.9	2178.1	177.1

Table 6.9: ACL packets

Type	Payload Header (bytes)	User Payload (bytes)	FEC	CRC	Symmetric Max. Rate (kb/s)
HV1	na	10	1/3	no	64.0
HV2	na	20	2/3	no	64.0
HV3	na	30	no	no	64.0
DV[1]	1 D	10+(0-9) D	2/3 D	yes D	64.0+57.6 D
EV3	na	1-30	No	Yes	96
EV4	na	1-120	2/3	Yes	192
EV5	na	1-180	No	Yes	288
2-EV3	na	1-60	No	Yes	192
2-EV5	na	1-360	No	Yes	576
3-EV3	na	1-90	No	Yes	288
3-EV5	na	1-540	No	Yes	864

Table 6.10: Synchronous packets

1. Items followed by ‘D’ relate to data field only.

7 BITSTREAM PROCESSING

Bluetooth devices shall use the bitstream processing schemes as defined in the following sections.

Before the payload is sent over the air interface, several bit manipulations are performed in the transmitter to increase reliability and security. An HEC is added to the packet header, the header bits are scrambled with a whitening word, and FEC coding is applied. In the receiver, the inverse processes are carried out. Figure 7.1 on page 137 shows the processes carried out for the packet header both at the transmit and the receive side. All header bit processes are mandatory.

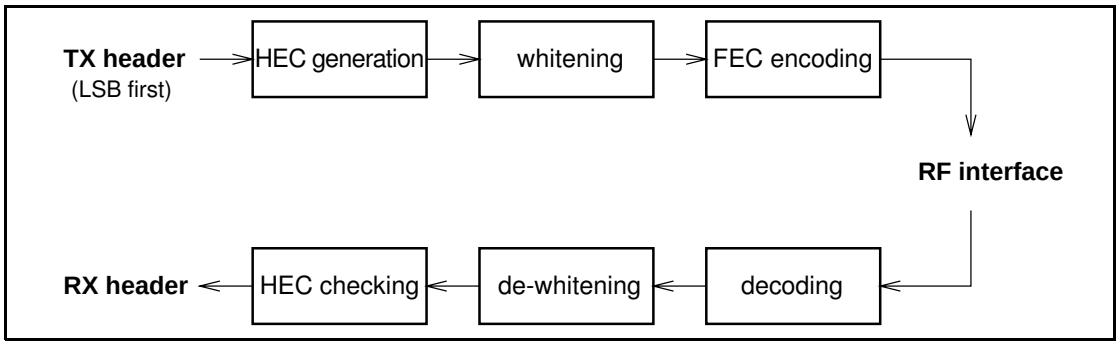


Figure 7.1: Header bit processes.

Figure 7.2 on page 137 shows the processes that may be carried out on the payload. In addition to the processes defined for the packet header, encryption can be applied on the payload. Only whitening and de-whitening, as explained in Section 7.2 on page 141, are mandatory for every payload; all other processes are optional and depend on the packet type (see Section 6.6 on page 128) and whether encryption is enabled. In Figure 7.2 on page 137, optional processes are indicated by dashed blocks.

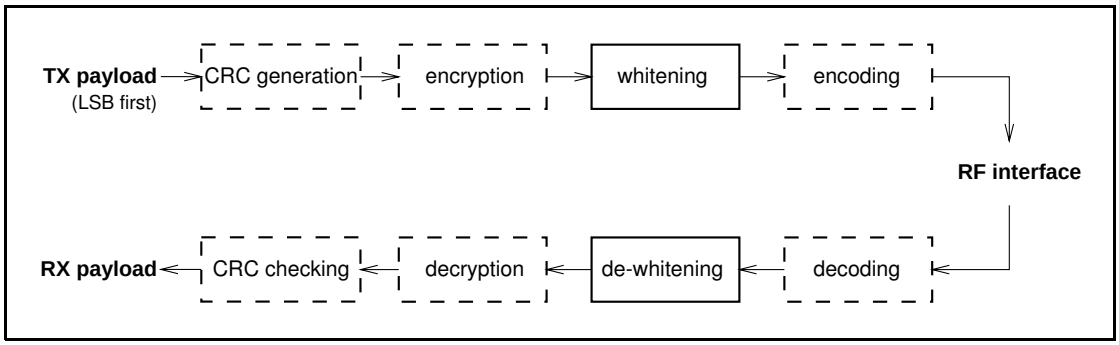


Figure 7.2: Payload bit processes.

7.1 ERROR CHECKING

The packet can be checked for errors or wrong delivery using the channel access code, the HEC in the header, and the CRC in the payload. At packet reception, the access code is checked first. Since the 64-bit sync word in the channel access code is derived from the 24-bit master LAP, this checks if the LAP is correct, and prevents the receiver from accepting a packet of another piconet (provided the LAP field of the master's BD_ADDR is different).

The HEC and CRC computations are normally initialized with the UAP of the master. Even though the access code may be the same for two piconets the different UAP values will typically cause the HEC and CRC to fail. However, there is an exception where no common UAP is available in the transmitter and receiver. This is the case when the HEC and CRC are generated for the FHS packet in **inquiry response** substate. In this case the DCI value shall be used.

The generation and check of the HEC and CRC are summarized in Figure 7.5 on page 139 and Figure 7.8 on page 140. Before calculating the HEC or CRC, the shift registers in the HEC/CRC generators shall be initialized with the 8-bit UAP (or DCI) value. Then the header and payload information shall be shifted into the HEC and CRC generators, respectively (with the LSB first).

7.1.1 HEC generation

The HEC generating LFSR is depicted in Figure 7.3 on page 138. The generator polynomial is $g(D) = (D+1)(D^7 + D^4 + D^3 + D^2 + 1) = D^8 + D^7 + D^5 + D^2 + D + 1$. Initially this circuit shall be pre-loaded with the 8-bit UAP such that the LSB of the UAP (denoted UAP_0) goes to the left-most shift register element, and, UAP_7 goes to the right-most element. The initial state of the HEC LFSR is depicted in Figure 7.4 on page 139. Then the data shall be shifted in with the switch S set in position 1. When the last data bit has been clocked into the LFSR, the switch S shall be set in position 2, and, the HEC can be read out from the register. The LFSR bits shall be read out from right to left (i.e., the bit in position 7 is the first to be transmitted, followed by the bit in position 6, etc.).

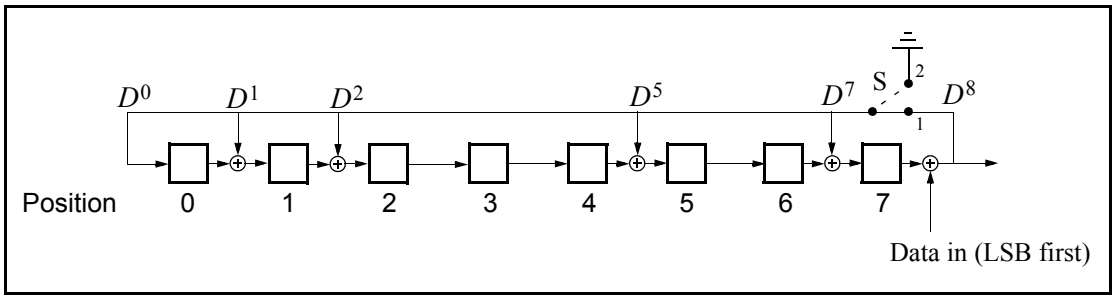


Figure 7.3: The LFSR circuit generating the HEC.

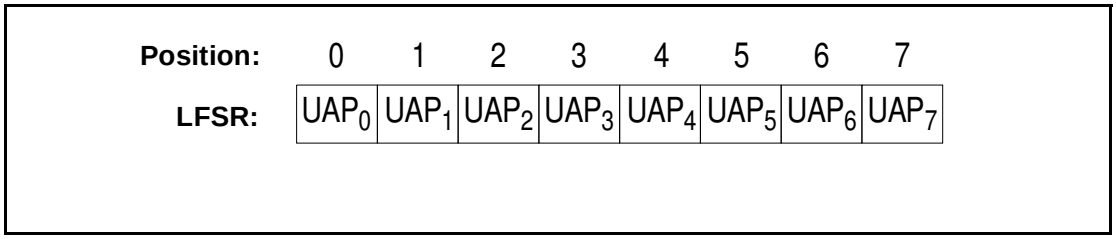


Figure 7.4: Initial state of the HEC generating circuit.

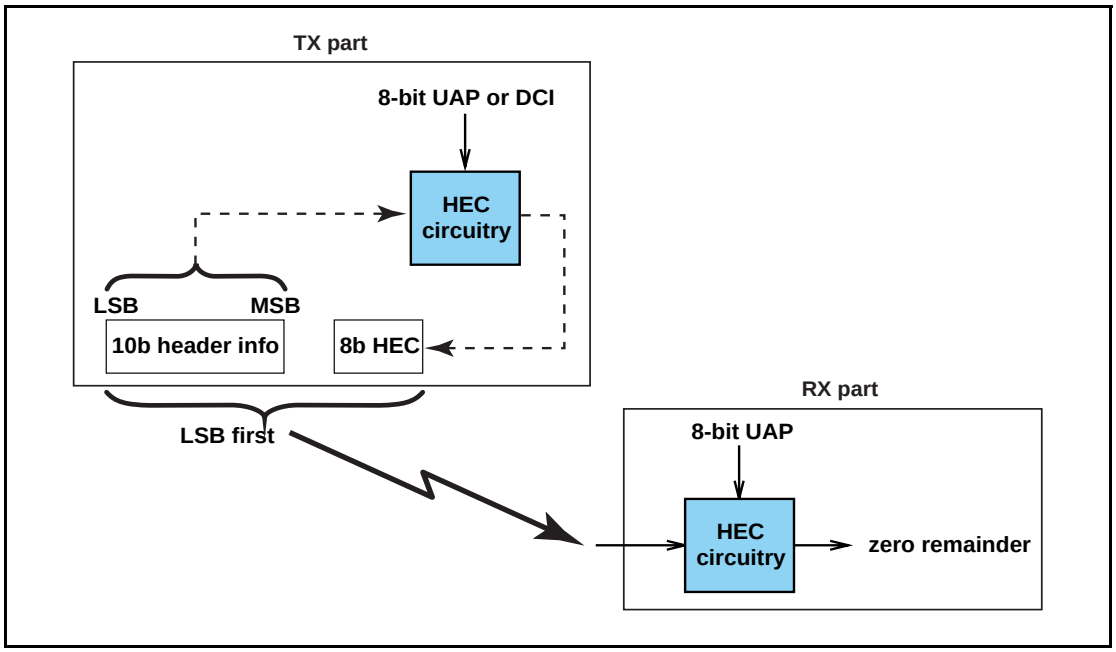


Figure 7.5: HEC generation and checking.

7.1.2 CRC generation

The 16 bit LFSR for the CRC is constructed similarly to the HEC using the CRC-CCITT generator polynomial $g(D) = D^{16} + D^{12} + D^5 + 1$ (i.e. `210041` in octal representation) (see Figure 7.6 on page 140). For this case, the 8 left-most bits shall be initially loaded with the 8-bit UAP (UAP_0 to the left and UAP_7 to the right) while the 8 right-most bits shall be reset to zero. The initial state of the 16 bit LFSR is specified in Figure 7.7 on page 140. The switch S shall be set in position 1 while the data is shifted in. After the last bit has entered the LFSR, the switch shall be set in position 2, and, the register's contents shall be transmitted, from right to left (i.e., starting with position 15, then position 14, etc.).

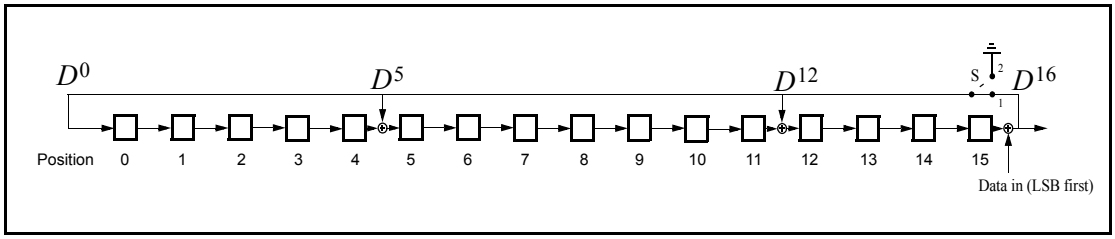


Figure 7.6: The LFSR circuit generating the CRC.

Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LFSR:	UAP_0	UAP_1	UAP_2	UAP_3	UAP_4	UAP_5	UAP_6	UAP_7	0	0	0	0	0	0	0	0

Figure 7.7: Initial state of the CRC generating circuit.

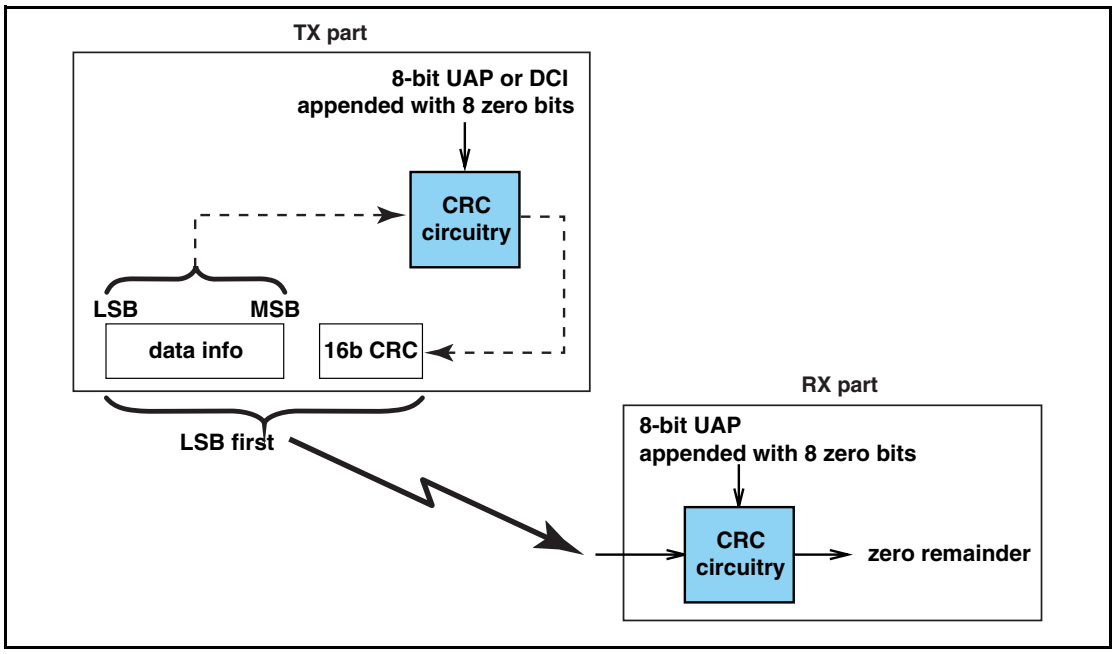


Figure 7.8: CRC generation and checking.

7.2 DATA WHITENING

Before transmission, both the header and the payload shall be scrambled with a data whitening word in order to randomize the data from highly redundant patterns and to minimize DC bias in the packet. The scrambling shall be performed prior to the FEC encoding.

At the receiver, the received data shall be descrambled using the same whitening word generated in the recipient. The descrambling shall be performed after FEC decoding.

The whitening word is generated with the polynomial $g(D) = D^7 + D^4 + 1$ (i.e., `221` in octal representation) and shall be subsequently XORed with the header and the payload. The whitening word is generated with the linear feedback shift register shown in Figure 7.9 on page 141. Before each transmission, the shift register shall be initialized with a portion of the master Bluetooth clock, CLK_{6-1}, extended with an MSB of value one. This initialization shall be carried out with CLK_1 written to position 0, CLK_2 written to position 1, etc. An exception is the FHS packet sent during page response or inquiry, where initialization of the whitening register shall be carried out differently. Instead of the master clock, the X-input used in the **inquiry** or **page response** (depending on current state) routine shall be used, see Table 2.2. The 5-bit value shall be extended with two MSBs of value 1. During register initialization, the LSB of X (i.e., X_0) shall be written to position 0, X_1 shall be written to position 1, etc.

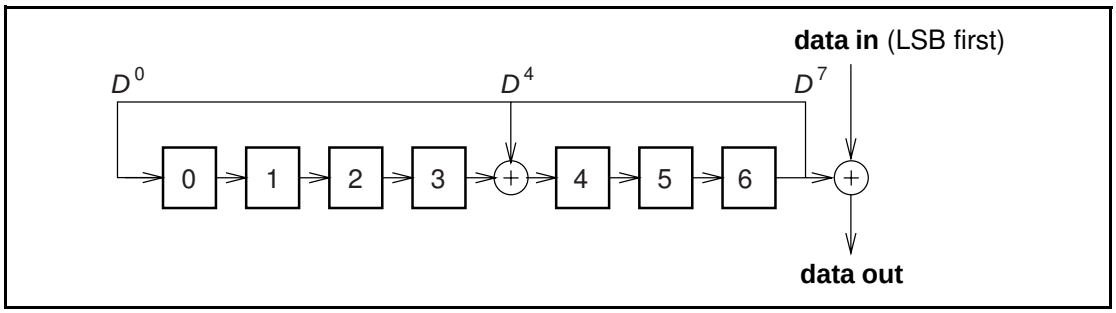


Figure 7.9: Data whitening LFSR.

After initialization, the packet header and the payload (including the CRC) are whitened. The payload whitening shall continue from the state the whitening LFSR had at the end of HEC. There shall be no re-initialization of the shift register between packet header and payload. The first bit of the “data in” sequence shall be the LSB of the packet header.

For Enhanced Data Rate packets, whitening shall not be applied to the guard, synchronization and trailer portions of the Enhanced Data Rate packets. During the periods where whitening is not applied the LFSR shall be paused.

7.3 ERROR CORRECTION

There are three error correction schemes defined for Bluetooth:

- 1/3 rate FEC
- 2/3 rate FEC
- ARQ scheme for the data

The purpose of the FEC scheme on the data payload is to reduce the number of retransmissions. However, in a reasonable error-free environment, FEC gives unnecessary overhead that reduces the throughput. Therefore, the packet definitions given in Section 6 on page 109 have been kept flexible to use FEC in the payload or not, resulting in the **DM** and **DH** packets for the ACL logical transport, **HV** packets for the SCO logical transport, and **EV** packets for the eSCO logical transport. The packet header is always protected by a 1/3 rate FEC since it contains valuable link information and is designed to withstand more bit errors.

Correction measures to mask errors in the voice decoder are not included in this section. This matter is discussed in Section 9.3 on page 198.

7.4 FEC CODE: RATE 1/3

A simple 3-times repetition FEC code is used for the header. The repetition code is implemented by repeating each bit three times, see the illustration in Figure 7.10 on page 142. The 3-times repetition code is used for the entire header, as well as for the synchronous data field in the **HV1** packet.

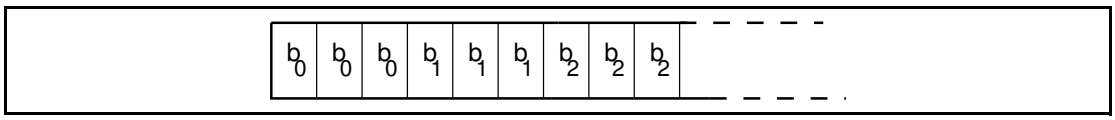


Figure 7.10: Bit-repetition encoding scheme.

7.5 FEC CODE: RATE 2/3

The other FEC scheme is a (15,10) shortened Hamming code. The generator polynomial is $g(D) = (D+1)(D^4+D+1)$. This corresponds to 65 in octal notation. The LFSR generating this code is depicted in Figure 7.11 on page 143. Initially all register elements are set to zero. The 10 information bits are sequentially fed into the LFSR with the switches S1 and S2 set in position 1. Then, after the final input bit, the switches S1 and S2 are set in position 2, and the five parity bits are shifted out. The parity bits are appended to the information bits. Subsequently, each block of 10 information bits is encoded into a 15 bit codeword. This code can correct all single errors and detect all double errors in each codeword. This 2/3 rate FEC is used in the **DM** packets, in the data field of the **DV** packet, in the **FHS** packet, in the **HV2** packet, and in the **EV4** packet. Since the encoder operates with information segments of length 10, tail bits with value zero shall be appended after the CRC bits to bring the total number of bits equal to a multiple of 10. The number of tail bits to append shall be the least possible that achieves this (i.e., in the interval 0...9). These tail bits are not included in the payload length indicator for ACL packets or in the payload length field of the eSCO setup LMP command.

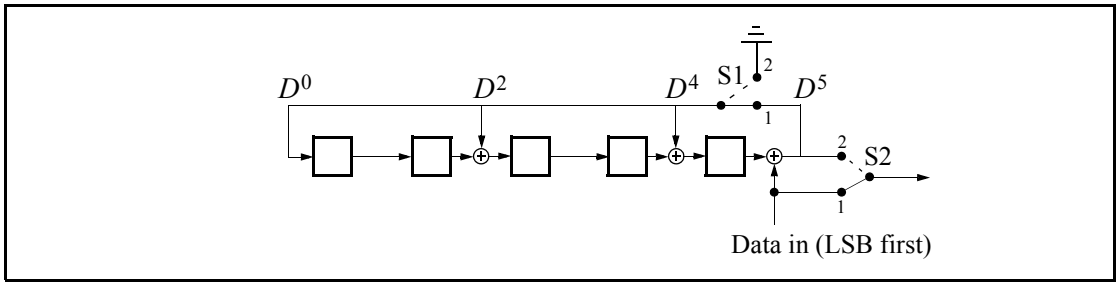


Figure 7.11: LFSR generating the (15,10) shortened Hamming code.

7.6 ARQ SCHEME

With an automatic repeat request scheme, **DM, DH** the data field of **DV** packets, and **EV** packets shall be transmitted until acknowledgement of a successful reception is returned by the destination (or timeout is exceeded). The acknowledgement information shall be included in the header of the return packet. The ARQ scheme is only used on the payload in the packet and only on packets that have a CRC. The packet header and the synchronous data payload of HV and DV packets are not protected by the ARQ scheme.

7.6.1 Unnumbered ARQ

Bluetooth uses a fast, unnumbered acknowledgment scheme. An ACK (ARQN=1) or a NAK (ARQN=0) is returned in response to the receipt of previously received packet. The slave shall respond in the slave-to-master slot directly following the master-to-slave slot unless the slave has scatternet commitments in that timeslot; the master shall respond at the next event addressing the same slave (the master may have addressed other slaves between the last received packet from the considered slave and the master response to this packet). For a packet reception to be successful, at least the HEC must pass. In addition, the CRC must pass if present.

In the first POLL packet at the start of a new connection (as a result of a page, page scan, role switch or unpark) the master shall initialize the ARQN bit to NAK. The response packet sent by the slave shall also have the ARQN bit set to NAK. The subsequent packets shall use the following rules. The initial value of the master's eSCO ARQN at link set-up shall be NAK.

The ARQ bit shall only be affected by data packets containing CRC and empty slots. As shown in Figure 7.12 on page 145, upon successful reception of a CRC packet, the ARQN bit shall be set to ACK. If, in any receive slot in the slave, or, in a receive slot in the master following transmission of a packet, one of these events applies:

1. no access code is detected,
2. the HEC fails,
3. the CRC fails,

then the ARQN bit shall be set to NAK. In eSCO the ARQN bit may be set to ACK even when the CRC on an EV packet has failed thus enabling delivery of erroneous packets.

Packets that have correct HEC but that are addressed to other slaves, or packets other than DH, DM, DV or EV packets, shall not affect the ARQN bit, except as noted in Section 7.6.2.2 on page 148. In these cases the ARQN bit shall be left as it was prior to reception of the packet. For ACL packets, if a CRC packet with a correct header has the same SEQN as the previously received CRC packet, the ARQN bit shall be set to ACK and the payload shall be ignored without checking the CRC. For eSCO packets, the SEQN shall not be used

when determining the ARQN. If an eSCO packet has been received successfully within the eSCO window subsequent receptions within the eSCO window shall be ignored. At the end of the eSCO window, the master's ARQN shall be retained for the first master-to-slave transmission in the next eSCO window.

The ARQ bit in the FHS packet is not meaningful. Contents of the ARQN bit in the FHS packet shall not be checked.

Broadcast packets shall be checked on errors using the CRC, but no ARQ scheme shall be applied. Broadcast packets shall never be acknowledged.

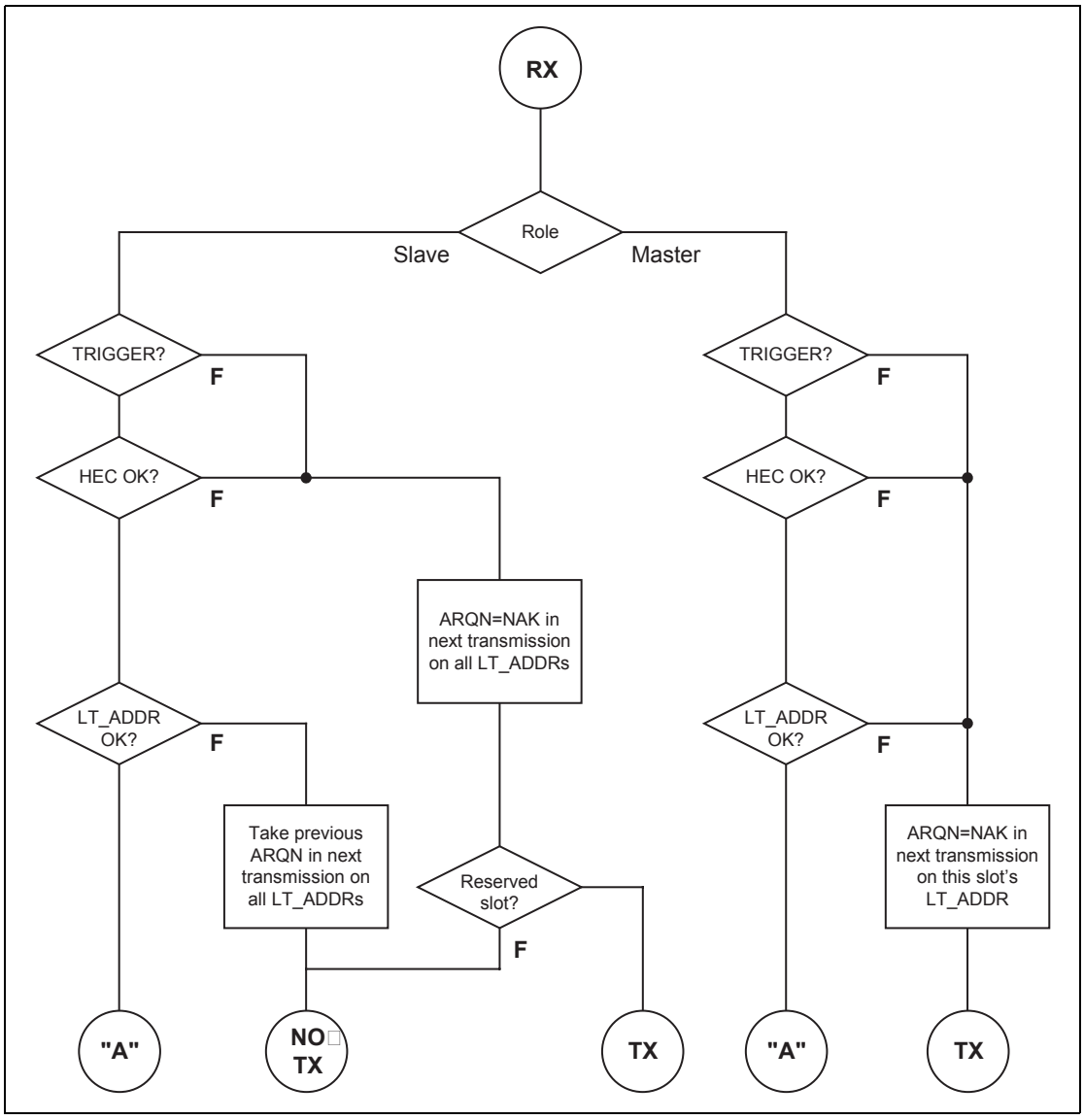


Figure 7.12: Stage 1 of the receive protocol for determining the ARQN bit.

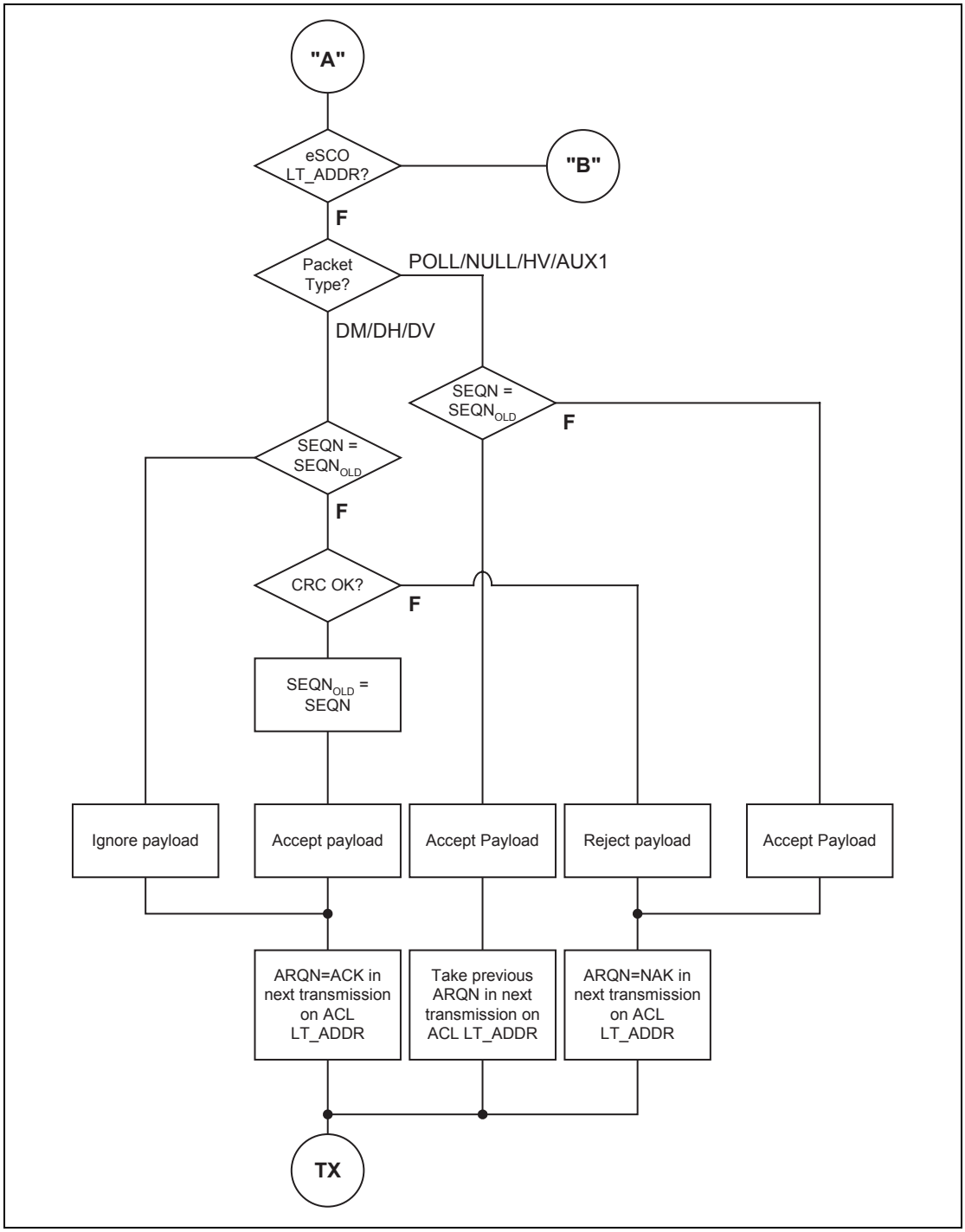


Figure 7.13: Stage 2 (ACL) of the receive protocol for determining the ARQN bit.

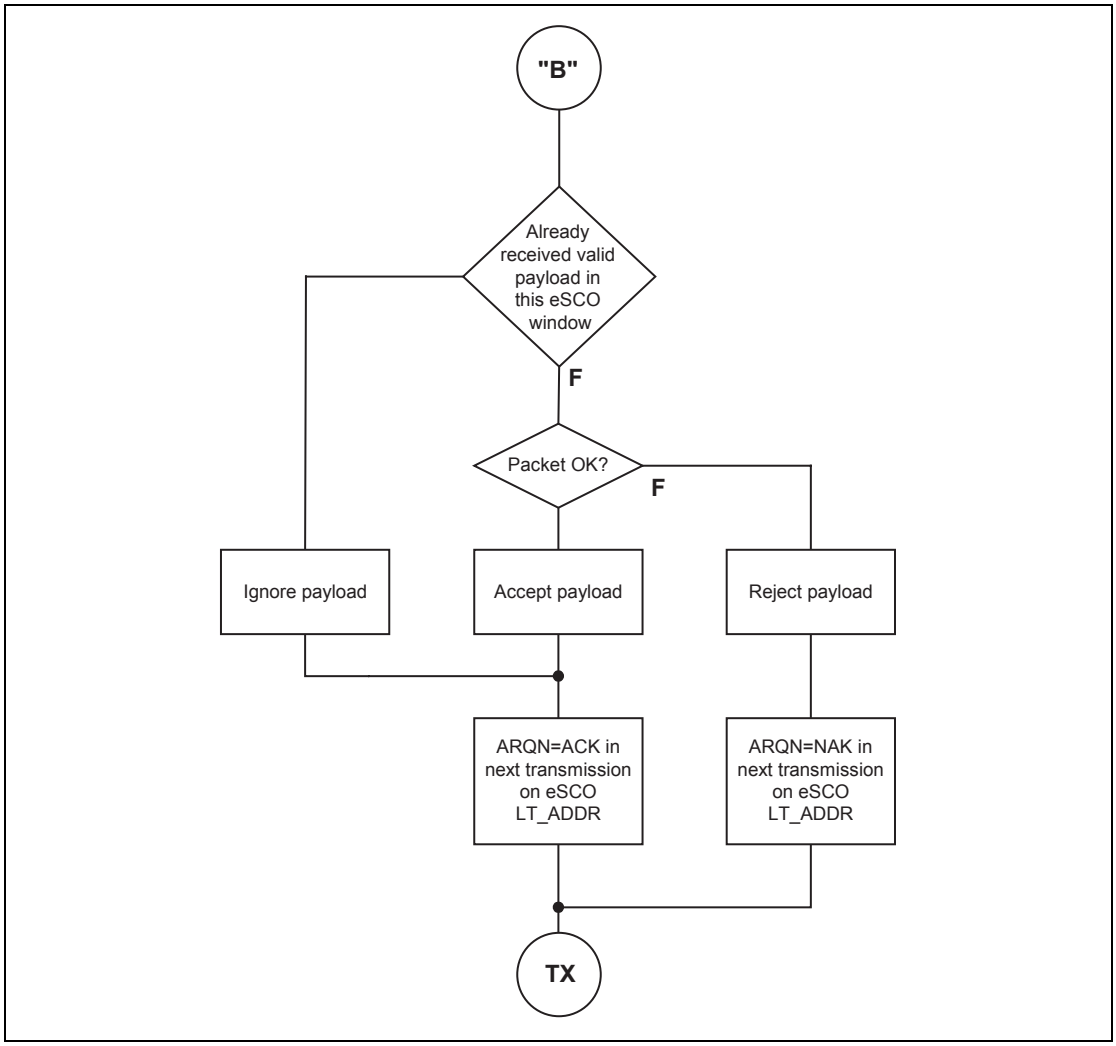


Figure 7.14: Stage 2 (eSCO) of the receive protocol for determining the ARQN bit.

7.6.2 Retransmit filtering

The data payload shall be transmitted until a positive acknowledgment is received or a timeout is exceeded. A retransmission shall be carried out either because the packet transmission itself failed, or because the acknowledgment transmitted in the return packet failed (note that the latter has a lower failure probability since the header is more heavily coded). In the latter case, the destination keeps receiving the same payload over and over again. In order to filter out the retransmissions in the destination, the SEQN bit is present in the header. Normally, this bit is alternated for every new CRC data payload transmission. In case of a retransmission, this bit shall not be changed so the destination can compare the SEQN bit with the previous SEQN value. If different, a new data payload has arrived; otherwise it is the same data payload and may be ignored. Only new data payloads shall be transferred to the Baseband Resource Manager. Note that CRC data payloads can be carried only by **DM**, **DH**, **DV** or **EV** packets.

7.6.2.1 Initialization of SEQN at start of new connection

The SEQN bit of the first CRC data packet at the start of a connection (as a result of page, page scan, role switch or unpark) on both the master and the slave sides shall be set to 1. The subsequent packets shall use the rules in the following sections.

7.6.2.2 ACL and SCO retransmit filtering

The SEQN bit shall only be affected by the CRC data packets as shown in Figure 7.15. It shall be inverted every time a new CRC data packet is sent. The CRC data packet shall be retransmitted with the same SEQN number until an ACK is received or the packet is flushed. When an ACK is received, a new payload may be sent and on that transmission the SEQN bit shall be inverted. If a device decides to flush (see Section 7.6.3 on page 150), and it has not received an acknowledgement for the current packet, it shall replace the current packet with an ACL-U continuation packet with the same sequence number as the current packet and length zero. If it replaces the current packet in this way it shall not move on to transmit the next packet until it has received an ack.

If the slave receives a packet other than DH, DM, DV or EV with the SEQN bit inverted from that in the last header succesfully received on the same LT_ADDR, it shall set the ARQN bit to NAK until a DH, DM, DV or EV packet is successfully received.

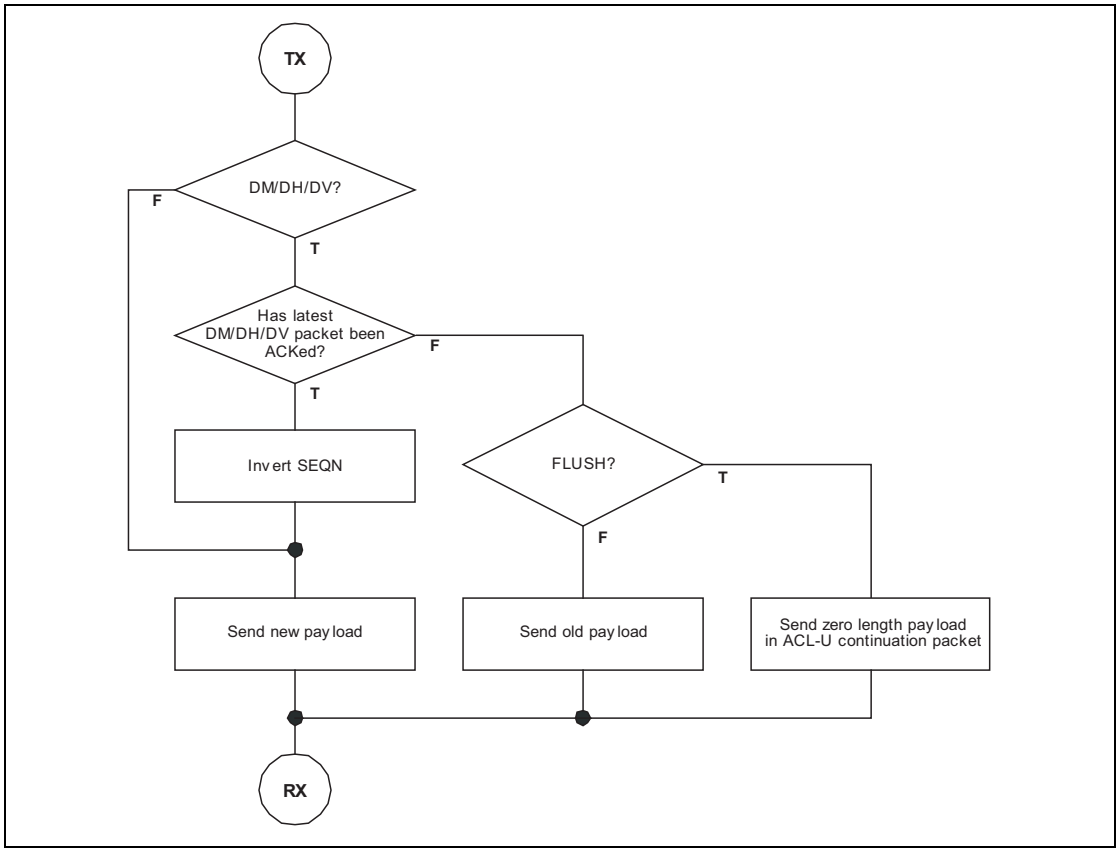


Figure 7.15: Transmit filtering for packets with CRC.

7.6.2.3 eSCO retransmit filtering

In eSCO, the SEQN bit shall be toggled every eSCO window. The value shall be constant for the duration of the eSCO window. The initial value of SEQN shall be zero.

For a given eSCO window the SEQN value shall be constant.

7.6.2.4 FHS retransmit filtering

The SEQN bit in the FHS packet is not meaningful. This bit may be set to any value. Contents of the SEQN bit in the FHS packet shall not be checked.

7.6.2.5 Packets without CRC retransmit filtering

During transmission of packets without a CRC the SEQN bit shall remain the same as it was in the previous packet.

7.6.3 Flushing payloads

In ACL, the ARQ scheme can cause variable delay in the traffic flow since retransmissions are inserted to assure error-free data transfer. For certain communication links, only a limited amount of delay is allowed: retransmissions are allowed up to a certain limit at which the current payload shall be ignored. This data transfer is indicated as **isochronous traffic**. This means that the retransmit process must be overruled in order to continue with the next data payload. Aborting the retransmit scheme is accomplished by *flushing* the old data and forcing the link controller to take the next data instead.

Flushing results in loss of remaining portions of an L2CAP message. Therefore, the packet following the flush shall have a start packet indication of LLID = 10 in the payload header for the next L2CAP message. This informs the destination of the flush. (see Section 6.6 on page 128). Flushing will not necessarily result in a change in the SEQN bit value, see the previous section.

The Flush Timeout defines a maximum period after which all segments of the ACL-U packet are flushed from the Controller buffer. The Flush Timeout shall start when the First segment of the ACL-U packet is stored in the Controller buffer. After the Flush timeout has expired the Link Controller may continue transmissions according to the procedure described in Section 7.6.2.2 on page 148, however the Baseband Resource Manager shall not continue the transmission of the ACL-U packet to the Link Controller. If the Baseband Resource Manager has further segments of the packet queued for transmission to the Link Controller it shall delete the remaining segments of the ACL-U packet from the queue. In case the complete ACL-U packet was not stored in the Controller buffer yet, any Continuation segments, received for the ACL logical transport, shall be flushed, until a First segment is received. When the complete ACL-U packet has been flushed, the Link Manager shall continue transmission of the next ACL-U packet for the ACL logical transport. The default Flush Timeout shall be infinite, i.e. re-transmissions are carried out until physical link loss occurs. This is also referred to as a 'reliable channel'. All devices shall support the default Flush Timeout.

In eSCO, packets shall be automatically flushed at the end of the eSCO window.

7.6.4 Multi-slave considerations

In a piconet with multiple logical transports, the master shall carry out the ARQ protocol independently on each logical transport.

7.6.5 Broadcast packets

Broadcast packets are packets transmitted by the master to all the slaves simultaneously. (see paragraph 8.6.4) If multiple hop sequences are being used each transmission may only be received by some of the slaves. In this case the master shall repeat the transmission on each hop sequence. A broad-

cast packet shall be indicated by the all-zero LT_ADDR (note; the FHS packet is the only packet which may have an all-zero LT_ADDR but is not a broadcast packet). Broadcast packets shall not be acknowledged (at least not at the LC level).

Since broadcast messages are not acknowledged, each broadcast packet is transmitted at least a fixed number of times. A broadcast packet should be transmitted N_{BC} times before the next broadcast packet of the same broadcast message is transmitted, see Figure 7.16 on page 152. Optionally, a broadcast packet may be transmitted N_{BC} + 1 times. Note: N_{BC}=1 means that each broadcast packet should be sent only once, but optionally may be sent twice. However, time-critical broadcast information may abort the ongoing broadcast train. For instance, unpark messages sent at beacon instances may do this, see Section 8.9.5 on page 192.

If multiple hop sequences are being used then the master may transmit on the different hop sequences in any order, providing that transmission of a new broadcast packet shall not be started until all transmissions of any previous broadcast packet have completed on all hop sequences. The transmission of a single broadcast packet may be interleaved among the hop sequences to minimize the total time to broadcast a packet. The master has the option of transmitting only N_{BC} times on channels common to all hop sequences.

Broadcast packets with a CRC shall have their own sequence number. The SEQN of the first broadcast packet with a CRC shall be set to SEQN = 1 by the master and shall be inverted for each new broadcast packet with CRC thereafter. Broadcast packets without a CRC have no influence on the sequence number. The slave shall accept the SEQN of the first broadcast packet it receives in a connection and shall check for change in SEQN for subsequent broadcast packets. Since there is no acknowledgement of broadcast messages and there is no end packet indication, it is important to receive the start packets correctly. To ensure this, repetitions of the broadcast packets that are L2CAP start packets and LMP packets shall not be filtered out. These packets shall be indicated by LLID=1X in the payload header as explained in section 6.6 on page 128. Only repetitions of the L2CAP continuation packets shall be filtered out.

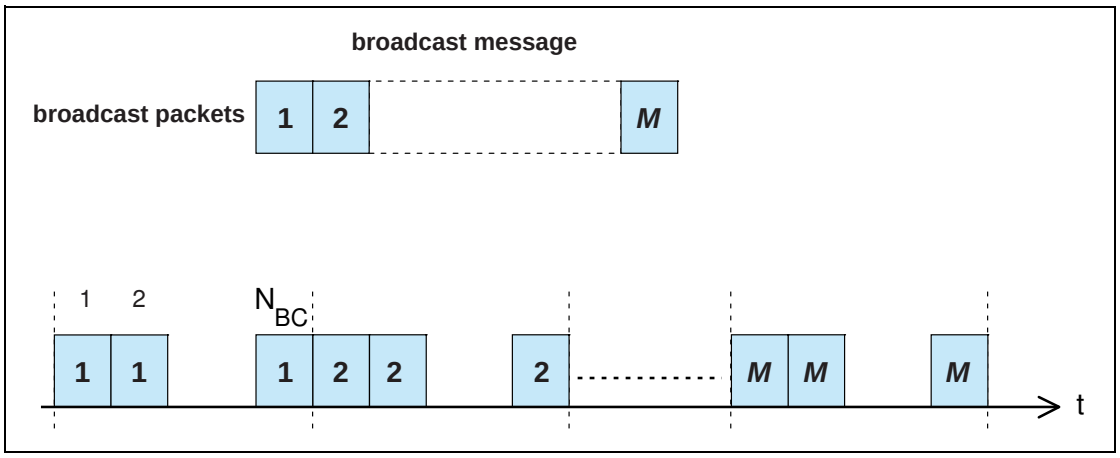


Figure 7.16: Broadcast repetition scheme

8 LINK CONTROLLER OPERATION

This section describes how a piconet is established and how devices can be added to and released from the piconet. Several states of operation of the devices are defined to support these functions. In addition, the operation of several piconets with one or more common members, the scatternet, is discussed.

8.1 OVERVIEW OF STATES

Figure 8.1 on page 153 shows a state diagram illustrating the different states used in the link controller. There are three major states: **STANDBY**, **CONNECTION**, and **PARK**; in addition, there are seven substates, **page**, **page scan**, **inquiry**, **inquiry scan**, **master response**, **slave response**, and **inquiry response**. The substates are interim states that are used to establish connections and enable device discovery. To move from one state or substate to another, either commands from the link manager are used, or internal signals in the link controller are used (such as the trigger signal from the correlator and the timeout signals).

.

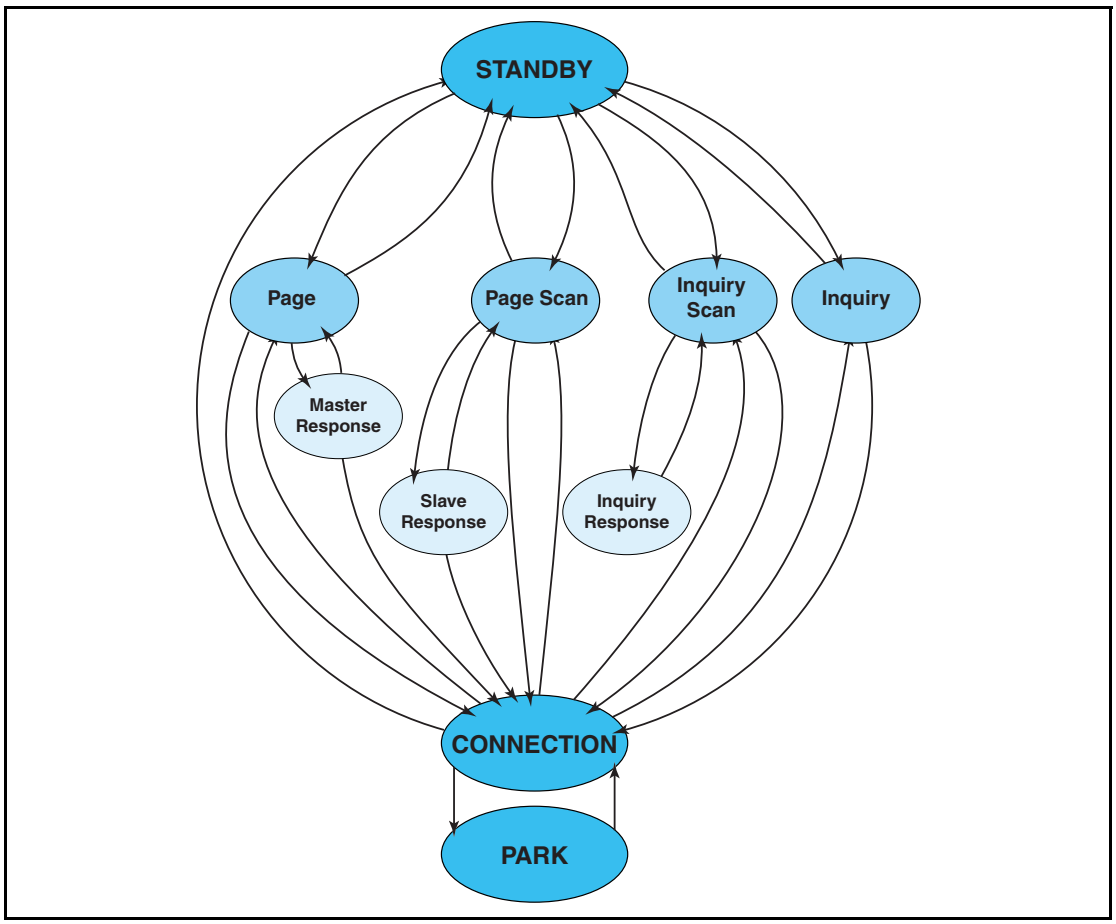


Figure 8.1: State diagram of link controller.

8.2 STANDBY STATE

The **STANDBY** state is the default state in the device. In this state, the device may be in a low-power mode. Only the native clock is running at the accuracy of the LPO (or better).

The controller may leave the **STANDBY** state to scan for page or inquiry messages, or to page or inquiry itself.

8.3 CONNECTION ESTABLISHMENT SUBSTATES

In order to establish new connections the paging procedure is used. Only the Bluetooth device address is required to set up a connection. Knowledge about the clock, obtained from the inquiry procedure (see Section 8.4 on page 163) or from a previous connection with this device, and the page scanning mode of the other device will accelerate the setup procedure. A device that establishes a connection carries out a page procedure and will automatically become the master of the connection.

8.3.1 Page scan substate

In the **page scan** substate, a device may be configured to use either the standard or interlaced scanning procedure. During a standard scan, a device listens for the duration of the scan window $T_{w\_page\_scan}$ (11.25ms default, see HCI [Part E] Section 7.3.20 on page 494), while the interlaced scan is performed as two back to back scans of $T_{w\_page\_scan}$. If the scan interval is not at least twice the scan window, then interlaced scan shall not be used. During each scan window, the device shall listen at a single hop frequency, its correlator matched to its device access code (DAC). The scan window shall be long enough to completely scan 16 page frequencies.

When a device enters the **page scan** substate, it shall select the scan frequency according to the page hopping sequence determined by the device's Bluetooth device address, see Section 2.6.4.1 on page 91. The phase in the sequence shall be determined by $CLKN_{16-12}$ of the device's native clock; that is, every 1.28s a different frequency is selected.

In the case of a standard scan, if the correlator exceeds the trigger threshold during the **page scan**, the device shall enter the **slave response** substate described in Section 8.3.3.1 on page 160. The scanning device may also use interlaced scan. In this case, if the correlator does not exceed the trigger threshold during the first scan it shall scan a second time using the phase in the sequence determined by $[CLKN_{16-12} + 16] \bmod 32$. If on this second scan the correlator exceeds the trigger threshold the device shall enter the **slave response** substate using $[CLKN_{16-12} + 16] \bmod 32$ as the frozen CLKN* in the calculation for $Xprs^{(79)}$, see Section 2.6.4.3 on page 92 for details. If the correlator does not exceed the trigger threshold during a scan in normal mode or

during the second scan in interlaced scan mode it shall return to either the **STANDBY** or **CONNECTION** state.

The **page scan** substate can be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and the device can use all the capacity to carry out the **page scan**. Before entering the **page scan** substate from the **CONNECTION** state, the device should reserve as much capacity as possible for scanning. If desired, the device may place ACL connections in Hold, Park or Sniff, see Section 8.8 on page 185 and Section 8.9 on page 185. Synchronous connections should not be interrupted by the page scan, although eSCO retransmissions should be paused during the scan. The page scan may be interrupted by the reserved synchronous slots which should have higher priority than the **page scan**. SCO packets should be used requiring the least amount of capacity (**HV3** packets). The scan window shall be increased to minimize the setup delay. If one SCO logical transport is present using **HV3** packets and T_{SCO}=6 slots or one eSCO logical transport is present using **EV3** packets and T_{ESCO}=6 slots, a total scan window $T_{w\ page\ scan}$ of at least 36 slots (22.5ms) is recommended; if two SCO links are present using **HV3** packets and T_{SCO}=6 slots or two eSCO links are present using **EV3** packets and T_{ESCO}=6 slots, a total scan window of at least 54 slots (33.75ms) is recommended.

The scan interval $T_{page\ scan}$ is defined as the interval between the beginnings of two consecutive page scans. A distinction is made between the case where the scan interval is equal to the scan window $T_{w\ page\ scan}$ (continuous scan), the scan interval is maximal 1.28s, or the scan interval is maximal 2.56s. These three cases shall determine the behavior of the paging device; that is, whether the paging device shall use R0, R1 or R2, see also Section 8.3.2 on page 156. Table 8.1 illustrates the relationship between $T_{page\ scan}$ and modes R0, R1 and R2. Although scanning in the R0 mode is continuous, the scanning may be interrupted for example by reserved synchronous slots. The scan interval information is included in the SR field in the FHS packet.

SR mode	$T_{page\ scan}$
R0	≤ 1.28s and = $T_{w\ page\ scan}$
R1	≤ 1.28s
R2	≤ 2.56s
Reserved	-

Table 8.1: Relationship between scan interval, and paging modes R0, R1 and R2.

8.3.2 Page substate

The **page** substate is used by the master (source) to activate and connect to a slave (destination) in the **page scan** substate. The master tries to coincide with the slave's scan activity by repeatedly transmitting the paging message consisting of the slave's device access code (DAC) in different hop channels. Since the Bluetooth clocks of the master and the slave are not synchronized, the master does not know exactly when the slave wakes up and on which hop frequency. Therefore, it transmits a train of identical page scan messages at different hop frequencies and listens in between the transmit intervals until it receives a response from the slave.

The page procedure in the master consists of a number of steps. First, the Host communicates the BD_ADDR of the slave to the Controller. This BD_ADDR shall be used by the master to determine the page hopping sequence, see Section 2.6.4.2 on page 92. The slave's BD_ADDR shall be used to determine the page hopping sequence, see Section 2.6.4.2 on page 92. For the phase in the sequence, the master shall use an estimate of the slave's clock. For example, this estimate can be derived from timing information that was exchanged during the last encounter with this particular device (which could have acted as a master at that time), or from an inquiry procedure. With this estimate CLKE of the slave's Bluetooth clock, the master can predict on which hop channel the slave starts page scanning.

The estimate of the Bluetooth clock in the slave can be completely wrong. Although the master and the slave use the same hopping sequence, they use different phases in the sequence and might never select the same frequency. To compensate for the clock drifts, the master shall send its page message during a short time interval on a number of wake-up frequencies. It shall transmit also on hop frequencies just before and after the current, predicted hop frequency. During each TX slot, the master shall sequentially transmit on two different hop frequencies. In the following RX slot, the receiver shall listen sequentially to two corresponding RX hops for ID packet. The RX hops shall be selected according to the page response hopping sequence. The page response hopping sequence is strictly related to the page hopping sequence: for each page hop there is a corresponding page response hop. The RX/TX timing in the **page** substate is described in Section 2.2.5 on page 72, see also Figure 2.7 on page 77. In the next TX slot, it shall transmit on two hop frequencies different from the former ones. Note: The hop rate is increased to 3200 hops/s.

With the increased hopping rate as described above, the transmitter can cover 16 different hop frequencies in 16 slots or 10 ms. The page hopping sequence is divided over two paging trains **A** and **B** of 16 frequencies. Train **A** includes the 16 hop frequencies surrounding the current, predicted hop frequency *f(k)*, where k is determined by the clock estimate $CLKE_{16\text{-}12}$. The first train consists of hops

f(k-8), f(k-7),...,f(k),...,f(k+7)

When the difference between the Bluetooth clocks of the master and the slave is between -8x1.28 s and +7x1.28 s, one of the frequencies used by the master will be the hop frequency the slave will listen to. Since the master does not know when the slave will enter the **page scan** substate, the master has to repeat this train **A** N_{page} times or until a response is obtained, whichever is shorter. If the slave scan interval corresponds to R1, the repetition number is at least 128; if the slave scan interval corresponds to R2 or if the master has not previously read the slave's SR mode, the repetition number is at least 256. If the master has not previously read the slave's SR mode it shall use Npage >= 256. Note that $CLKE_{16-12}$ changes every 1.28 s; therefore, every 1.28 s, the trains will include different frequencies of the page hopping set.

When the difference between the Bluetooth clocks of the master and the slave is less than -8x1.28 s or larger than +7x1.28 s, the remaining 16 hops are used to form the new 10 ms train **B.** The second train consists of hops

f(k-16), f(k-15),...,f(k-9),f(k+8)...,f(k+15)

Train **B** shall be repeated for N_{page} times. If no response is obtained, train **A** shall be tried again N_{page} times. Alternate use of train A and train B shall be continued until a response is received or the timeout *pageTO* is exceeded. If a response is returned by the slave, the master device enters the **master response** substate.

The **page** substate may be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and the device can use all the capacity to carry out the page. Before entering the **page** substate from the **CONNECTION** state, the device should free as much capacity as possible for scanning. To ensure this, it is recommended that the ACL connections are put on hold or park. However, the synchronous connections shall not be disturbed by the page. This means that the page will be interrupted by the reserved SCO and eSCO slots which have higher priority than the page. In order to obtain as much capacity for paging, it is recommended to use the SCO packets which use the least amount of capacity (**HV3** packets). If SCO or eSCO links are present, the repetition number N_{page} of a single train shall be increased, see Table 8.2. Here it has been assumed that the **HV3** packet are used with an interval T_{SCO}=6 slots or **EV3** packets are used with an interval of T_{ESCO}=6 slots, which would correspond to a 64 kb/s synchronous link.

.

SR mode	no synchronous link	one synchronous link (HV3)	two synchronous links (HV3)
R0	$N_{page} \geq 1$	$N_{page} \geq 2$	$N_{page} \geq 3$
R1	$N_{page} \geq 128$	$N_{page} \geq 256$	$N_{page} \geq 384$
R2	$N_{page} \geq 256$	$N_{page} \geq 512$	$N_{page} \geq 768$

Table 8.2: Relationship between train repetition, and paging modes R0, R1 and R2 when synchronous links are present.

The construction of the page train shall be independent of the presence of synchronous links; that is, synchronous packets are sent on the reserved slots but shall not affect the hop frequencies used in the unreserved slots, see Figure 8.2 on page 158.

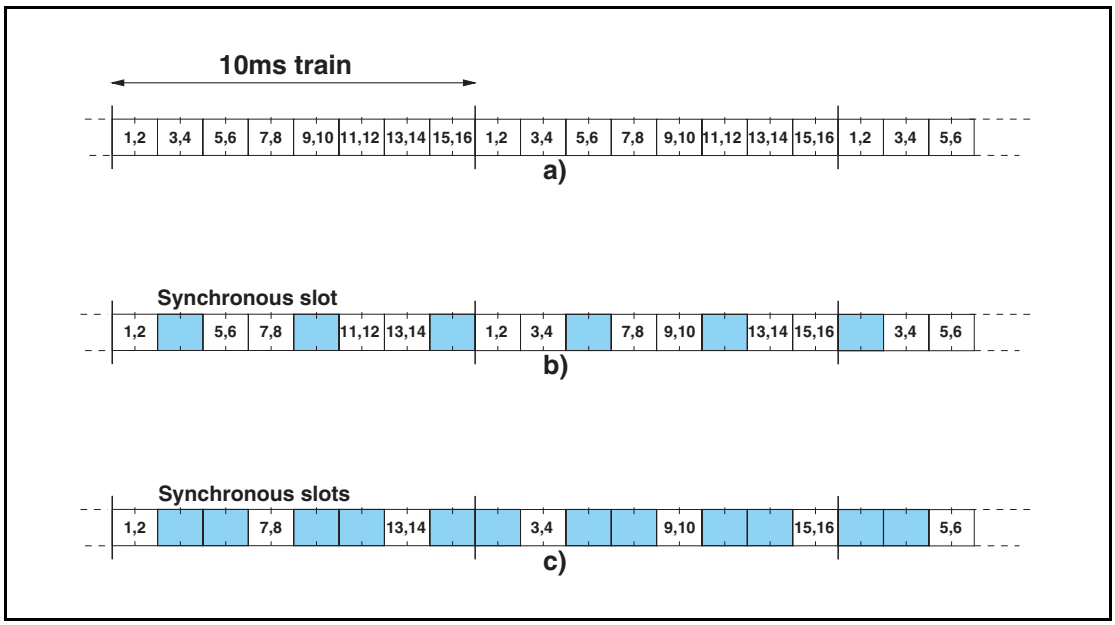


Figure 8.2: Conventional page (a), page while one synchronous link present (b), page while two synchronous links present (c).

8.3.3 Page response substates

When a page message is successfully received by the slave, there is a coarse FH synchronization between the master and the slave. Both the master and the slave enter a response substate to exchange vital information to continue the connection setup. It is important for the piconet connection that both devices shall use the same channel access code, use the same channel hopping sequence, and their clocks are synchronized. These parameters shall be derived from the master device. The device that initializes the connection (starts paging) is defined as the master device (which is thus only valid during the time the piconet exists). The channel access code and channel hopping sequence shall be derived from the Bluetooth device address (BD_ADDR) of the master. The timing shall be determined by the master clock. An offset shall be added to the slave's native clock to temporarily synchronize the slave clock to the master clock. At start-up, the master parameters are transmitted from the master to the slave. The messaging between the master and the slave at start-up is specified in this section.

The initial messaging between master and slave is shown in Table 8.3 on page 159 and in Figure 8.3 on page 160 and Figure 8.4 on page 160. In those two figures frequencies f (k), f(k+1), etc. are the frequencies of the page hopping sequence determined by the slave's BD_ADDR. The frequencies f'(k), f'(k+1), etc. are the corresponding page_response frequencies (slave-to-master). The frequencies g(m) belong to the basic channel hopping sequence.

Step	Message	Packet Type	Direction	Hopping Sequence	Access Code and Clock
1	Page	ID	Master to slave	Page	Slave
2	First slave page response	ID	Slave to master	Page response	Slave
3	Master page response	FHS	Master to slave	Page	Slave
4	Second slave page response	ID	Slave to master	Page response	Slave
5	1st packet master	POLL	Master to slave	Channel	Master
6	1st packet slave	Any type	Slave to master	Channel	Master

Table 8.3: Initial messaging during start-up.

In step 1 (see Table 8.3 on page 159), the master device is in **page** substate and the slave device in the **page scan** substate. Assume in this step that the page message sent by the master reaches the slave. On receiving the page message, the slave enters the **slave response** in step 2. The master waits for a reply from the slave and when this arrives in step 2, it will enter the **master response** in step 3. Note that during the initial message exchange, all parame-

ters are derived from the slave's device address, and that only the page hopping and page response hopping sequences are used (are also derived from the slave's device address). Note that when the master and slave enter the response states, their clock input to the page and page response hop selection is frozen as is described in Section 2.6.4.3 on page 92.

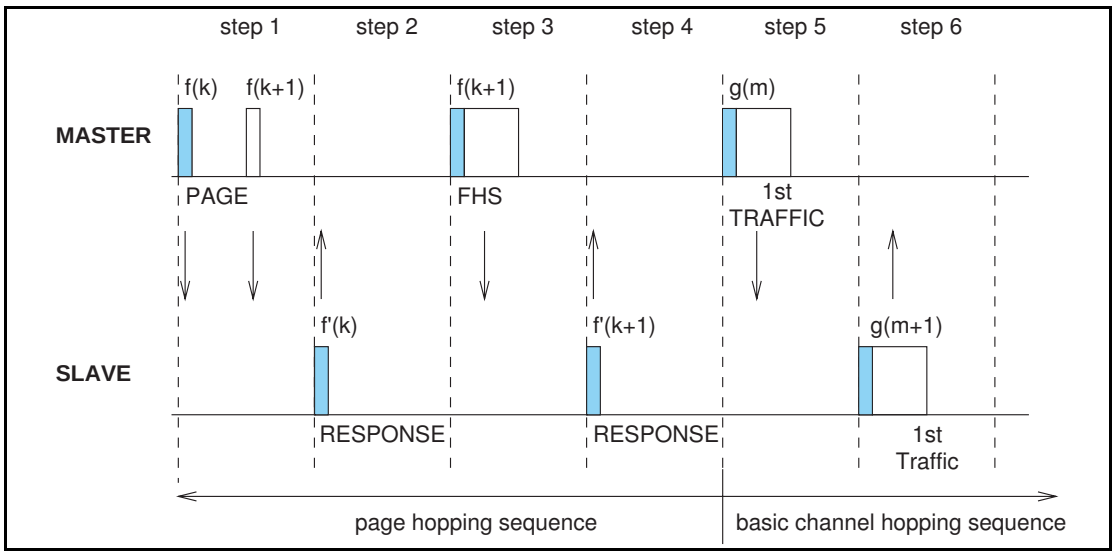


Figure 8.3: Messaging at initial connection when slave responds to first page message.

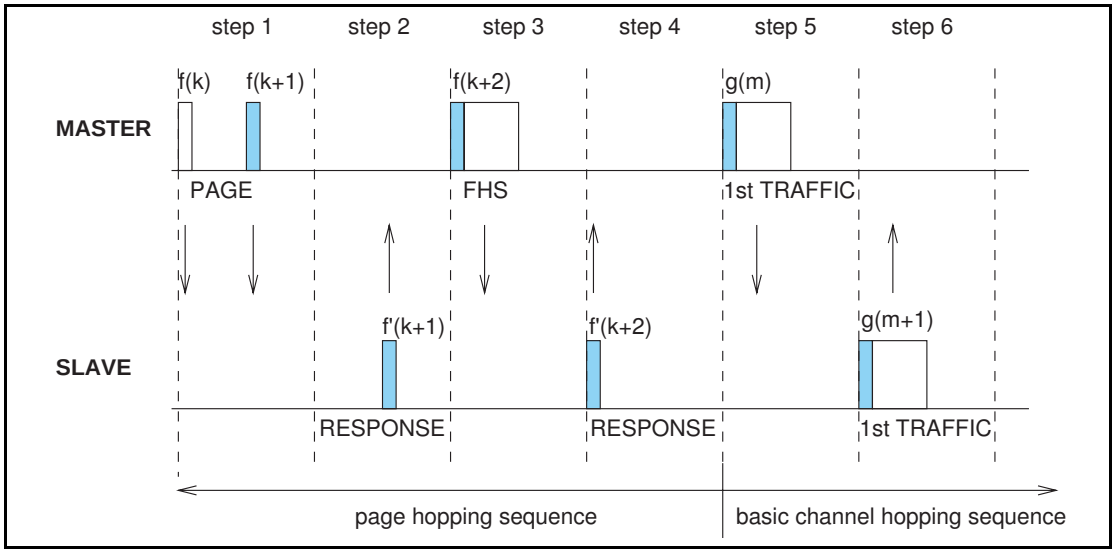


Figure 8.4: Messaging at initial connection when slave responds to second page message.

8.3.3.1 Slave response substate

After having received the page message in step 1, the slave device shall transmit a slave page response message (the slave's device access code) in step 2. This response message shall be the slave's device access code. The slave shall transmit this response 625 μs after the beginning of the received page message and at the response hop frequency that corresponds to the hop fre-

quency in which the page message was received. The slave transmission is therefore time aligned to the master transmission. During initial messaging, the slave shall still use the page response hopping sequence to return information to the master. The clock input $CLKN_{16-12}$ shall be frozen at the value it had at the time the page message was received.

After having sent the response message, the slave's receiver shall be activated 312.5 μs after the start of the response message and shall await the arrival of an **FHS** packet. Note that an **FHS** packet can arrive 312.5 μs after the arrival of the page message as shown in Figure 8.4 on page 160, and not after 625 μs as is usually the case in the piconet physical channel RX/TX timing. More details about the timing can be found in Section 2.4.4 on page 78.

If the setup fails before the **CONNECTION** state has been reached, the following procedure shall be carried out. The slave shall listen as long as no **FHS** packet is received until *pageRespTO* is exceeded. Every 1.25 ms, however, it shall select the next master-to-slave hop frequency according to the page hop sequence. If nothing is received after *pageRespTO*, the slave shall return back to the **page scan** substate for one scan period. Length of the scan period depends on the synchronous reserved slots present. If no page message is received during this additional scan period, the slave shall resume scanning at its regular scan interval and return to the state it was in prior to the first page scan state.

If an **FHS** packet is received by the slave in the **slave response** substate, the slave shall return a slave page response message in step 4 to acknowledge reception of the **FHS** packet. This response shall use the page response hopping sequence. The transmission of the slave page response packet is based on the reception of the **FHS** packet. Then the slave shall change to the master's channel access code and clock as received from the **FHS** packet. Only the 26 MSBs of the master clock are transferred: the timing shall be such that CLK_1 and CLK_0 are both zero at the time the **FHS** packet was received as the master transmits in even slots only. The offset between the master's clock and the slave's clock shall be determined from the master's clock in the **FHS** packet and reported to the slave's Baseband Resource Manager.

Finally, the slave enters the **CONNECTION** state in step 5. From then on, the slave shall use the master's clock and the master's BD_ADDR to determine the basic channel hopping sequence and the channel access code. The slave shall use the LT_ADDR in the **FHS** payload as the primary LT_ADDR in the **CONNECTION** state. The connection mode shall start with a POLL packet transmitted by the master. The slave may respond with any type of packet. If the POLL packet is not received by the slave, or the response packet is not received by the master, within *newconnectionTO* number of slots after FHS packet acknowledgement, the master and the slave shall return to **page** and **page scan** substates, respectively. See Section 8.5 on page 167

8.3.3.2 Master response substate

When the master has received a slave page response message in step 2, it shall enter the **master response** routine. It shall freeze the current clock input to the page hop selection scheme. The master shall then transmit an **FHS** packet in step 3 containing the master's real-time Bluetooth clock, the master's BD_ADDR, the BCH parity bits, and the class of device. The **FHS** packet contains all information to construct the channel access code without requiring a mathematical derivation from the master's Bluetooth device address. The LT_ADDR field in the packet header of FHS packets in the master response substate shall be set to all-zeros. The **FHS** packet shall be transmitted at the beginning of the master-to-slave slot following the slot in which the slave responded. The FHS packet shall carry the all-zero LT_ADDR. The TX timing of the **FHS** is not based on the reception of the response packet from the slave. The **FHS** packet may therefore be sent 312.5 μs after the reception of the response packet like shown in Figure 8.4 on page 160 and not 625 μs after the received packet as is usual in the piconet physical channel RX/TX timing, see also Section 2.4.4 on page 78.

After the master has sent its **FHS** packet, it shall wait for a second slave page response message in step 4 acknowledging the reception of the **FHS** packet. This response shall be the slave's device access code. If no response is received, the master shall retransmit the **FHS** packet with an updated clock and still using the slave's parameters. It shall retransmit the FHS packet with the clock updated each time until a second slave page response message is received, or the timeout of *pagerespTO* is exceeded. In the latter case, the master shall return to the **page** substate and send an error message to the Baseband Resource Manager. During the retransmissions of the **FHS** packet, the master shall use the page hopping sequence.

If the slave's response is received, the master shall change to using the master parameters, so it shall use the channel access code and the master clock. The lower clock bits CLK_0 and CLK_1 shall be reset to zero at the start of the **FHS** packet transmission and are not included in the **FHS** packet. Finally, the master enters the **CONNECTION** state in step 5. The master BD_ADDR shall be used to change to a new hopping sequence, the *basic channel hopping sequence*. The basic channel hopping sequence uses all 79 hop channels in a pseudo-random fashion, see also Section 2.6.4.7 on page 94. The master shall now send its first traffic packet in a hop determined with the new (master) parameters. This first packet shall be a POLL packet. See Section 8.5 on page 167. This packet shall be sent within *newconnectionTO* number of slots after reception of the FHS packet acknowledgement. The slave may respond with any type of packet. If the POLL packet is not received by the slave or the POLL packet response is not received by the master within *newconnectionTO* number of slots, the master and the slave shall return to **page** and **page scan** substates, respectively.

8.4 DEVICE DISCOVERY SUBSTATES

In order to discover other devices a device shall enter **inquiry** substate. In this substate, it shall repeatedly transmit the inquiry message (ID packet, see Section 6.5.1.1 on page 119) at different hop frequencies. The **inquiry** hop sequence is derived from the LAP of the GIAC. Thus, even when DIACs are used, the applied hopping sequence is generated from the GIAC LAP. A device that allows itself to be discovered, shall regularly enter the **inquiry scan** substate to respond to inquiry messages. The following sections describe the message exchange and contention resolution during inquiry response. The inquiry response is optional: a device is not forced to respond to an inquiry message.

During the **inquiry** substate, the discovering device collects the Bluetooth device addresses and clocks of all devices that respond to the inquiry message. It can then, if desired, make a connection to any one of them by means of the previously described page procedure.

The inquiry message broadcast by the source does not contain any information about the source. However, it may indicate which class of devices should respond. There is one general inquiry access code (GIAC) to inquire for any device, and a number of dedicated inquiry access codes (DIAC) that only inquire for a certain type of device. The inquiry access codes are derived from reserved Bluetooth device addresses and are further described in Section 6.3.1 on page 111.

8.4.1 Inquiry scan substate

The **inquiry scan** substate is very similar to the **page scan** substate. However, instead of scanning for the device's device access code, the receiver shall scan for the inquiry access code long enough to completely scan for 16 inquiry frequencies. Two types of scans are defined: standard and interlaced. In the case of a standard scan the length of this scan period is denoted $T_{w\_inquiry\_scan}$ (11.25ms default, see HCI [Part E] Section 7.3.22 on page 496). The standard scan is performed at a single hop frequency as defined by Xir_{4-0} (see Section 2.6.4.6 on page 94). The interlaced scan is performed as two back to back scans of $T_{w\_inquiry\_scan}$ where the first scan is on the normal hop frequency and the second scan is defined by $[Xir_{4-0} + 16] \bmod 32$. If the scan interval is not at least twice the scan window then interlaced scan shall not be used. The inquiry procedure uses 32 dedicated inquiry hop frequencies according to the inquiry hopping sequence. These frequencies are determined by the general inquiry address. The phase is determined by the native clock of the device carrying out the **inquiry scan**; the phase changes every 1.28s.

Instead of, or in addition to, the general inquiry access code, the device may scan for one or more dedicated inquiry access codes. However, the scanning shall follow the inquiry scan hopping sequence determined by the general inquiry address. If an inquiry message is received during an inquiry wake-up period, the device shall enter the **inquiry response** substate.

The **inquiry scan** substate can be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and the device can use all the capacity to carry out the **inquiry scan**. Before entering the **inquiry scan** substate from the **CONNECTION** state, the device should reserve as much capacity as possible for scanning. If desired, the device may place ACL logical transports in Sniff, Hold, or Park. Synchronous logical transports are preferably not interrupted by the **inquiry scan**, although eSCO retransmissions should be paused during the scan. In this case, the **inquiry scan** may be interrupted by the reserved synchronous slots. SCO packets should be used requiring the least amount of capacity (**HV3** packets). The scan window, $T_{w\ inquiry\ scan}$, shall be increased to increase the probability to respond to an inquiry message. If one SCO logical transport is present using HV3 packets and T_{SCO}=6 slots or one eSCO logical transport is present using EV3 packets and T_{ESCO}=6 slots, a total scan window of at least 36 slots (22.5ms) is recommended; if two SCO links are present using HV3 packets and T_{SCO}=6 slots or two eSCO links are present using EV3 packets and T_{ESCO}=6 slots, a total scan window of at least 54 slots (33.75ms) is recommended.

The scan interval $T_{inquiry\ scan}$ is defined as the interval between two consecutive inquiry scans. The **inquiry scan** interval shall be less than or equal to 2.56 s.

8.4.2 Inquiry substate

The **inquiry** substate is used to discover new devices. This substate is very similar to the **page** substate; the TX/RX timing shall be the same as in paging, see Section 2.4.4 on page 78 and Figure 2.7 on page 77. The TX and RX frequencies shall follow the inquiry hopping sequence and the inquiry response hopping sequence, and are determined by the general inquiry access code and the native clock of the discovering device. In between inquiry transmissions, the receiver shall scan for inquiry response messages. When a response is received, the entire packet (an **FHS** packet) is read, after which the device shall continue with inquiry transmissions. The device in an **inquiry** substate shall not acknowledge the inquiry response messages. If enabled by the Host (see HCI [Part E] Section 7.3.54 on page 533), the RSSI value of the inquiry response message shall be measured. It shall keep probing at different hop channels and in between listening for response packets. As in the **page** substate, two 10 ms trains **A** and **B** are defined, splitting the 32 frequencies of the inquiry hopping sequence into two 16-hop parts. A single train shall be repeated for at least $N_{inquiry}$=256 times before a new train is used. In order to collect all responses in an error-free environment, at least three train switches must have taken place. As a result, the **inquiry** substate may have to last for 10.24 s unless the inquirer collects enough responses and aborts the **inquiry** substate earlier. If desired, the inquirer may also prolong the **inquiry** substate to increase the probability of receiving all responses in an error-prone environment. If an inquiry procedure is automatically initiated periodically (say a 10 s period every minute), then the interval between two inquiry instances shall be determined randomly. This is done to avoid two devices synchronizing their inquiry procedures.

The **inquiry** substate is continued until stopped by the Baseband Resource Manager (when it decides that it has sufficient number of responses), when a timeout has been reached (*inquiryTO*), or by a command from the host to cancel the inquiry procedure.

The **inquiry** substate can be entered from the **STANDBY** state or the **CONNECTION** state. In the **STANDBY** state, no connection has been established and the device can use all the capacity to carry out the inquiry. Before entering the **inquiry** substate from the **CONNECTION** state, the device should free as much capacity as possible for scanning. To ensure this, it is recommended that the ACL logical transports are placed in Sniff, Hold, or Park. However, the reserved slots of synchronous logical transports shall not be disturbed by the inquiry. This means that the inquiry will be interrupted by the reserved SCO and eSCO slots which have higher priority than the inquiry. In order to obtain as much capacity as possible for inquiry, it is recommended to use the SCO packets which use the least amount of capacity (**HV3** packets). If SCO or eSCO links are present, the repetition number $N_{inquiry}$ shall be increased, see Table 8.4 on page 166.
Here it has been assumed that **HV3** packets are used with an interval T_{SCO}=6

slots or **EV3** packets are used with an interval of T_{ESCO}=6 slots, which would correspond to a 64 kb/s synchronous link.

	No synchronous links	One synchronous link (HV3)	Two synchronous links (HV3)
$N_{inquiry}$	≥ 256	≥ 512	≥ 768

Table 8.4: Increase of train repetition when synchronous links are present.

8.4.3 Inquiry response substate

The slave response substate for inquiries differs completely from the slave response substate applied for pages. When the inquiry message is received in the **inquiry scan** substate, the recipient shall return an inquiry response (FHS) packet containing the recipient's device address (BD_ADDR) and other parameters.

The following protocol in the slave's **inquiry response** shall be used. On the first inquiry message received in this substate the slave shall enter the **inquiry response** substate and shall return an **FHS** response packet to the master 625us after the inquiry message was received. A contention problem may arise when several devices are in close proximity to the inquiring device and all respond to an inquiry message at the same time. However, because every device has a free running clock it is highly unlikely that they all use the same phase of the inquiry hopping sequence. In order to avoid repeated collisions between devices that wake up in the same inquiry hop channel simultaneously, a device shall back-off for a random period of time. Thus, if the device receives an inquiry message and returns an FHS packet, it shall generate a random number, RAND, between 0 and MAX_RAND. For scanning intervals ≥ 1.28s MAX_RAND shall be 1023, however, for scanning intervals < 1.28s MAX_RAND may be as small as 127. A profile that uses a special DIAC may choose to use a smaller MAX_RAND than 1023 even when the scanning interval is ≥ 1.28s. The slave shall return to the **CONNECTION** or **STANDBY** state for the duration of at least RAND time slots. Before returning to the **CONNECTION** and **STANDBY** state, the device may go through the **page scan** substate. After at least RAND slots, the device shall add an offset of 1 to the phase in the inquiry hop sequence (the phase has a 1.28 s resolution) and return to the **inquiry scan** substate again. If the slave is triggered again, it shall repeat the procedure using a new RAND. The offset to the clock accumulates each time an **FHS** packet is returned. During a probing window, a slave may respond multiple times, but on different frequencies and at different times. Reserved synchronous slots should have priority over response packets; that is, if a response packet overlaps with a reserved synchronous slot, it shall not be sent but the next inquiry message is awaited.

The messaging during the inquiry routines is summarized in Table 8.5 on page 167. In step 1, the master transmits an inquiry message using the inquiry access code and its own clock. The slave responds with the **FHS** packet containing the slave's Bluetooth device address, native clock and other slave information. This **FHS** packet is returned at times that tend to be random. The **FHS**

packet is not acknowledged in the inquiry routine, but it is retransmitted at other times and frequencies as long as the master is probing with inquiry messages.

Step	Message	Packet Type	Direction	Hopping Sequence	Access Code and Clock
1	Inquiry	ID	master to slave	inquiry	inquiry
2	Inquiry response	FHS	slave to master	inquiry response	inquiry

Table 8.5: Messaging during inquiry routines.

8.5 CONNECTION STATE

In the **CONNECTION** state, the connection has been established and packets can be sent back and forth. In both devices, the channel (master) access code, the master's Bluetooth clock, and the AFH_channel_map are used. **CONNECTION** state uses the *basic* or *adapted channel hopping sequence*.

The **CONNECTION** state starts with a POLL packet sent by the master to verify the switch to the master's timing and channel frequency hopping. The slave may respond with any type of packet. If the slave does not receive the POLL packet or the master does not receive the response packet for *newconnectionTO* number of slots, both devices shall return to **page**/**page scan** substates.

The first information packets in the **CONNECTION** state contain control messages that characterize the link and give more details regarding the devices. These messages are exchanged between the link managers of the devices. For example, they may define the SCO logical transport and the sniff parameters. Then the transfer of user information can start by alternately transmitting and receiving packets.

The **CONNECTION** state is left through a **detach** or **reset** command. The **detach** command is used if the link has been disconnected in the normal way; all configuration data in the link controller shall remain valid. The **reset** command is a soft reset of the link controller. The functionality of the soft reset is described in [Part E] Section 7.3.2 on page 469.

In the **CONNECTION** state, if a device is not going to be nominally present on the channel at all times it may describe its unavailability by using sniff mode or hold mode (see Figure 8.5 on page 168).

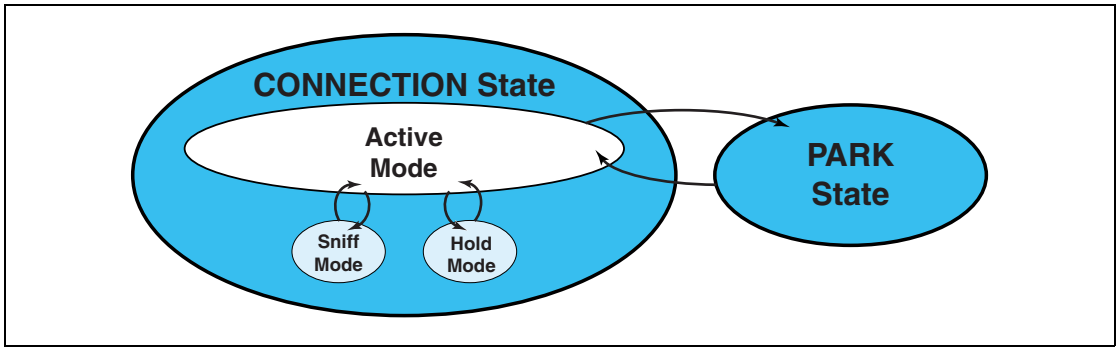


Figure 8.5: Connection state.

8.6 ACTIVE MODE

In the active mode, both master and slave actively participate on the channel. Up to seven slaves may be in the active mode at any given time. The master schedules the transmission based on traffic demands to and from the different slaves. In addition, it supports regular transmissions to keep slaves synchronized to the channel. Slaves in the active mode listen in the master-to-slave slots for packets. These devices are known as *active slaves*. If an active slave is not addressed, it may sleep until the next new master transmission. Slaves can derive the number of slots the master has reserved for transmission from TYPE field in the packet header; during this time, the non-addressed slaves do not have to listen on the master-to-slave slots. When a device is participating in multiple piconets it should listen in the master-to-slave slot for the current piconet. It is recommended that a device not be away from each piconet in which it is participating for more than T_{poll} slots. A periodic master transmission is required to keep the slaves synchronized to the channel. Since the slaves only need the channel access code to synchronize, any packet type can be used for this purpose.

Only the slave that is addressed by one of its LT_ADDRs (primary or secondary) may return a packet in the next slave-to-master slot. If no valid packet header is received, the slave may only respond in its reserved SCO or eSCO slave-to-master slot. In the case of a broadcast message, no slave shall return a packet (an exception is the access window for access requests in the **PARK** state, see Section 8.9 on page 185).

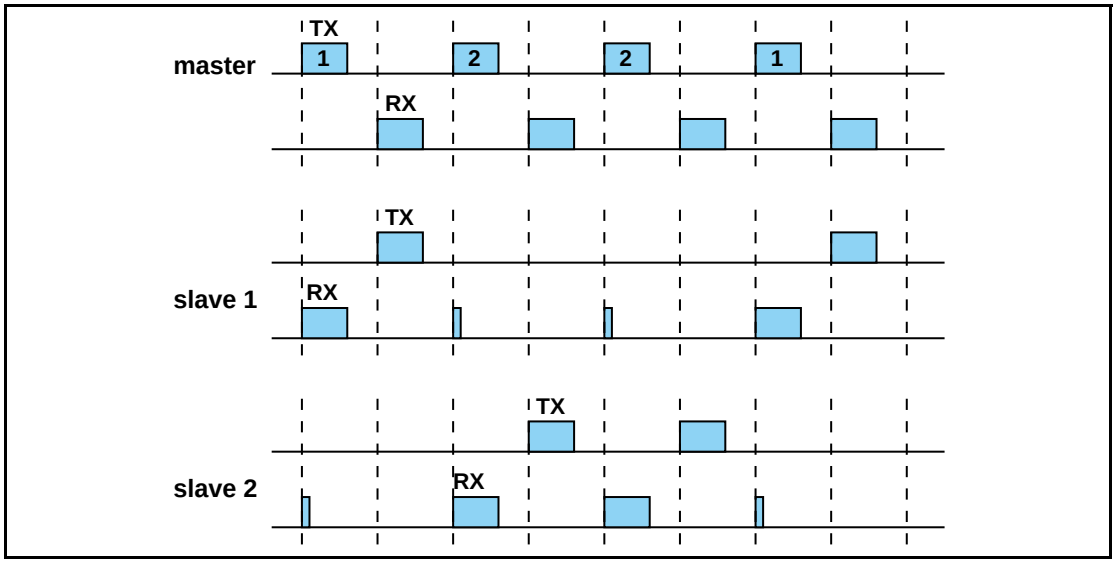


Figure 8.6: RX/TX timing in multi-slave configuration

For ACL logical transports the mode selection may be left to real time packet type selections. The packet type table (ptt) in section 6.5 allows the selection of Basic Rate or Enhanced Data Rate for each of the packet type codes, however; the DM1 packet is available in all packet type tables. ACL traffic over this given physical or logical link shall utilize the packet types in the given column of Packets defined for synchronous and asynchronous logical transport types.

8.6.1 Polling in the active mode

The master always has full control over the piconet. Due to the TDD scheme, slaves can only communicate with the master and not other slaves. In order to avoid collisions on the ACL logical transport, a slave is only allowed to transmit in the slave-to-master slot when addressed by the LT_ADDR in the packet header in the preceding master-to-slave slot. If the LT_ADDR in the preceding slot does not match, or a valid packet header was not received, the slave shall not transmit.

The master normally attempts to poll a slave's ACL logical transport no less often than once every T_{poll} slots. T_{poll} is set by the Link Manager (see [Part C] Section 4.1.8 on page 244).

The slave's ACL logical transport may be polled with any packet type except for FHS and ID. For example, polling during SCO may use HV packets, since the slave may respond to an HV packet with a DM1 packet (see Section 8.6.2 on page 169).

8.6.2 SCO

The SCO logical transport shall be established by the master sending an SCO setup message via the LM protocol. This message contains timing parameters

including the SCO interval T_{SCO} and the offset D_{SCO} to specify the reserved slots.

In order to prevent clock wrap-around problems, an initialization flag in the LMP setup message indicates whether initialization procedure 1 or 2 is being used. The slave shall apply the initialization method as indicated by the initialization flag. The master shall use initialization 1 when the MSB of the current master clock (CLK_{27}) is 0; it shall use initialization 2 when the MSB of the current master clock (CLK_{27}) is 1. The master-to-slave SCO slots reserved by the master and the slave shall be initialized on the slots for which the clock satisfies the applicable equation:

$$CLK_{27-1} \bmod T_{SCO} = D_{SCO} \qquad \text{for initialization 1}$$

$$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_{SCO} = D_{SCO} \qquad \text{for initialization 2}$$

The slave-to-master SCO slots shall directly follow the reserved master-to-slave SCO slots. After initialization, the clock value CLK(k+1) for the next master-to-slave SCO slot shall be derived by adding the fixed interval T_{SCO} to the clock value of the current master-to-slave SCO slot:

$$CLK(k+1) = CLK(k) + T_{SCO}$$

The master will send SCO packets to the slave at regular intervals (the SCO interval T_{SCO} counted in slots) in the reserved master-to-slave slots. An HV1 packet can carry 1.25ms of speech at a 64 kb/s rate. An HV1 packet shall therefore be sent every two time slots (T_{SCO}=2). An HV2 packet can carry 2.5ms of speech at a 64 kb/s rate. An HV2 packet shall therefore be sent every four time slots (T_{SCO}=4). An HV3 packet can carries 3.75ms of speech at a 64 kb/s rate. An HV3 packet shall therefore be sent every six time slots (T_{SCO}=6).

The slave is allowed to transmit in the slot reserved for its SCO logical transport unless the (valid) LT_ADDR in the preceding slot indicates a different slave. If no valid LT_ADDR can be derived in the preceding slot, the slave may still transmit in the reserved SCO slot.

Since the DM1 packet is recognized on the SCO logical transport, it may be sent during the SCO reserved slots if a valid packet header with the primary LT_ADDR is received in the preceding slot. DM1 packets sent during SCO reserved slots shall only be used to send ACL-C data.

The slave shall not transmit anything other than an HV packet in a reserved SCO slot unless it decodes its own slave address in the packet header of the packet in the preceding master-to-slave transmission slot.

8.6.3 eSCO

The eSCO logical transport is established by the master sending an eSCO setup message via the LM protocol. This message contains timing parameters including the eSCO interval T_{ESCO} and the offset D_{ESCO} to specify the reserved slots.

To enter eSCO, the master or slave shall send an eSCO setup command via the LM protocol. This message shall contain the eSCO interval T_{ESCO} and an offset D_{ESCO}. In order to prevent clock wrap-around problems, an initialization flag in the LMP setup message indicates whether initialization procedure 1 or 2 shall be used. The initiating device shall use initialization 1 when the MSB of the current master clock (CLK_{27}) is 0; it shall use initialization 2 when the MSB of the current master clock (CLK_{27}) is 1. The responding device shall apply the initialization method as indicated by the initialization flag. The master-to-slave eSCO slots reserved by the master and the slave shall be initialized on the slots for which the clock satisfies the applicable equation:

$$CLK_{27-1} \bmod T_{ESCO} = D_{ESCO} \qquad \text{for initialization 1}$$

$$(\overline{CLK_{27}}, CLK_{26-1}) \bmod T_{ESCO} = D_{ESCO} \qquad \text{for initialization 2}$$

The slave-to-master eSCO slots shall directly follow the reserved master-to-slave eSCO slots. After initialization, the clock value CLK(k+1) for the next master-to-slave eSCO slot shall be found by adding the fixed interval T_{ESCO} to the clock value of the current master-to-slave eSCO slot:

$$CLK(k+1) = CLK(k) + T_{ESCO}$$

When an eSCO logical transport is established, the master shall assign an additional LT_ADDR to the slave. This provides the eSCO logical transport with an ARQ scheme that is separate from that of the ACL logical transport. All traffic on a particular eSCO logical transport, and only that eSCO traffic, is carried on the eSCO LT_ADDR. The eSCO ARQ scheme uses the ARQN bit in the packet header, and operates similarly to the ARQ scheme on ACL links.

There are two different polling rules in eSCO. In the eSCO reserved slots, the polling rule is the same as to the SCO reserved slots. The master may send a packet in the master slot. The slave may transmit on the eSCO LT_ADDR in the following slot either if it received a packet on the eSCO LT_ADDR in the previous slot, or if it did not receive a valid packet header in the previous slot. When the master-to-slave packet type is a three-slot packet, the slave's transmit slot is the fourth slot of the eSCO reserved slots. A master shall transmit in an eSCO retransmission window on a given eSCO LT_ADDR only if it addressed that eSCO LT_ADDR in the immediately preceding eSCO reserved slots. A slave may transmit on an eSCO LT_ADDR in the eSCO reserved slots only if the slave did not received a valid packet header with a different LT_ADDR in the eSCO reserved slots. Inside retransmission windows, the

same polling rule as for ACL traffic is used: the slave shall transmit on the eSCO channel only if it received a valid packet header and correct LT_ADDR on the eSCO channel in the previous master-to-slave transmission slot. The master may transmit on any non-eSCO LT_ADDR in any master-to-slave transmission slot inside the eSCO retransmission window. The master shall only transmit on an eSCO LT_ADDR in the retransmission window if there are enough slots left for both the master and slave packets to complete in the retransmission window. The master may refrain from transmitting in any slot during the eSCO retransmission window. When there is no data to transmit from master to slave, either due to the traffic being unidirectional or due to the master-to-slave packet having been ACK'ed, the master shall use the POLL packet. When the master-to-slave packet has been ACK'ed, and the slave-to-master packet has been correctly received, the master shall not address the slave on the eSCO LT_ADDR until the next eSCO reserved slot, with the exception that the master may transmit a NULL packet with ARQN=ACK on the eSCO LT_ADDR. When there is no data to transmit from slave to master, either due to the traffic being unidirectional or due to the slave-to-master packet having been ACK'ed, the slave shall use NULL packets. eSCO traffic should be given priority over ACL traffic in the retransmission window.

Figure 8.7 on page 172 shows the eSCO window when single slot packets are used.

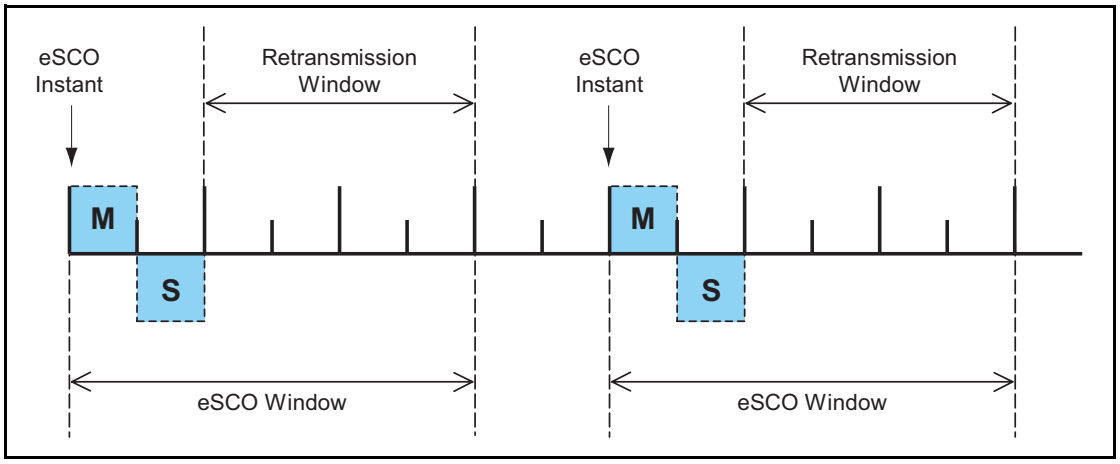


Figure 8.7: eSCO Window Details for Single-Slot Packets

When multi-slot packets are used in either direction of the eSCO logical transport, the first transmission continues into the following slots. The retransmission window in this case starts the slot after the end of the slave-to-master packet, i.e. two, four or six slots immediately following the eSCO instant are reserved and should not be used for other traffic. Figure 8.8 on page 173 shows the eSCO window when multi-slot packets are used in one direction and single-slot packets are used in the other direction.

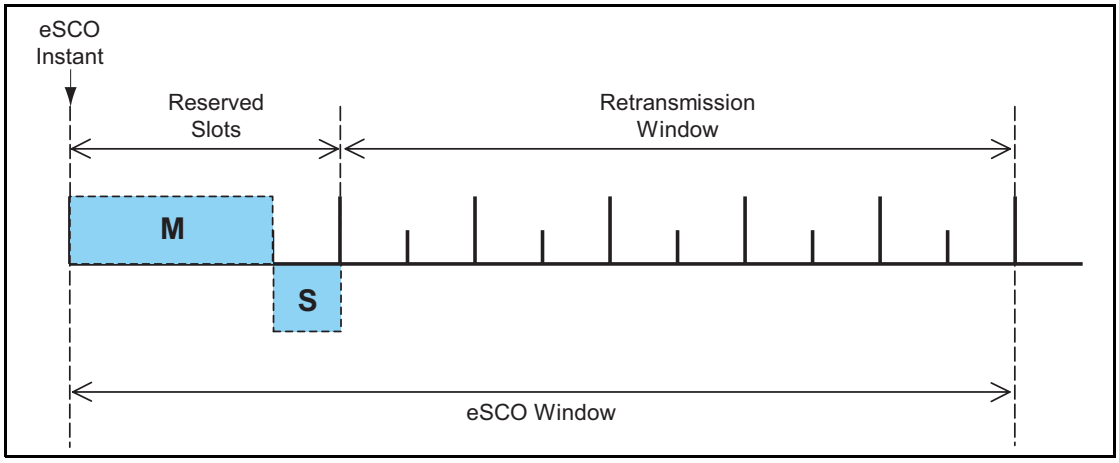


Figure 8.8: eSCO Window Details for Asymmetric Traffic

eSCO windows may overlap on the master, but shall not overlap on an individual slave.

In the reserved slots both master and slave shall only listen and transmit at their allocated slots at the first transmission time of each eSCO window. Intermittent lapses due to, for instance, time-critical signaling during connection establishment are allowed. Both master and slave may refrain from sending data and may use instead POLL and NULL packets respectively. When the master transmits a POLL packet instead of an EV4 or EV5 packet the slave shall transmit starting in the same slot as if the master transmitted an EV4 or EV5 packet. If the slave does not receive anything in the reserved master-to-slave transmission slot it shall transmit in the same slot as if the master had transmitted the negotiated packet type. For example, if the master had negotiated an EV5 packet the slave would transmit three slots later. [If the master does not receive a slave transmission in response to an eSCO packet it causes an implicit NAK of the packet in question. The listening requirements for the slave during the retransmission window are the same as for an active ACL logical transport.

8.6.4 Broadcast scheme

The master of the piconet can broadcast messages to all slaves on the ASB-U, PSB-C, and PSB-U logical transports. A broadcast packet shall have an LT_ADDR set to all zero. Each new broadcast message (which may be carried by a number of packets) shall start with the start of L2CAP message indication (LLID=10).

The Broadcast LT_ADDR shall use a ptt=0.

A broadcast packet shall never be acknowledged. In an error-prone environment, the master may carry out a number of retransmissions to increase the probability for error-free delivery, see also Section 7.6.5 on page 150.

In order to support the **PARK** state (as described in Section 8.9 on page 185), a master transmission shall take place at fixed intervals. This master transmission will act as a beacon to which slaves can synchronize. If no traffic takes place at the beacon event, broadcast packets shall be sent. More information is given in Section 8.9 on page 185.

8.6.5 Role switch

There are several occasions when a role switch is used.

- a role switch is necessary when joining an existing piconet by paging, since by definition, the paging device is initially master of a "small" piconet only involving the pager (master) and the paged (slave) device.
- a role switch is necessary in order for a slave in an existing piconet to set up a new piconet with itself as master and the original piconet master as slave. If the original piconet had more than one slave, then this implies a double role for the original piconet master; it becomes a slave in the new piconet while still maintaining the original piconet as master.

Prior to the role switch, encryption if present, shall be stopped in the old piconet. A role switch shall not be performed if the physical link is in Sniff or Hold mode, in the **PARK** state, or has any synchronous logical transports.

For the master and slave involved in the role switch, the switch results in a reversal of their TX and RX timing: a TDD switch. Additionally, since the piconet parameters are derived from the Bluetooth device address and clock of the master, a role switch inherently involves a redefinition of the piconet as well: a piconet switch. The new piconet's parameters shall be derived from the former slave's device address and clock.

Assume device A is to become master; device B was the former master. Then there are two alternatives: either the slave initiates the role switch or the master initiates the role switch. These alternatives are described in Link Manager Protocol, [Part C] Section 4.4.2 on page 268. The baseband procedure is the same regardless of which alternative is used.

In step 1, the slave A and master B shall perform a TDD switch using the former hopping scheme (still using the Bluetooth device address and clock of device B), so there is no piconet switch yet. The slot offset information sent by slave A shall not be used yet but shall be used in step 3. Device A now becomes the master, device B the slave. The LT_ADDR formerly used by device A in its slave role, shall now be used by slave B.

At the moment of the TDD switch, both devices A and B shall start a timer with a time out of *newconnectionTO*. The timer shall be stopped in slave B as soon as it receives an FHS packet from master A on the TDD-switched channel. The timer shall be stopped in master A as soon as it receives an ID packet from slave B. If the *newconnectionTO* expires, the master and slave shall return to the old piconet timing and AFH state, taking their old roles of master and slave. The FHS packet shall be sent by master A using the "old" piconet parameters. The LT_ADDR in the FHS packet header shall be the former LT_ADDR used by device A. The LT_ADDR carried in the FHS payload shall be the new LT_ADDR intended for device B when operating on the new piconet. After the FHS acknowledgment, which is the ID packet and shall be sent by the slave on the old hopping sequence, both master A and slave B shall use the new channel parameters of the new piconet as indicated by the FHS with the sequence selection set to *basic channel hopping sequence*. If the new master has physi-

cal links that are *AFH enabled*, following the piconet switch the new master is responsible for controlling the AFH operational mode of its new slave.

Since the old and new masters' clocks are synchronized, the clock information sent in the FHS payload shall indicate the new master's clock at the beginning of the FHS packet transmission. Furthermore, the 1.25 ms resolution of the clock information given in the FHS packet is not sufficient for aligning the slot boundaries of the two piconets. The slot-offset information in the LMP message previously sent by device A shall be used to provide more accurate timing information. The slot offset indicates the delay between the start of the master-to-slave slots of the old and new piconet channels. This timing information ranges from 0 to 1249 μs with a resolution of 1μs. It shall be used together with the clock information in the FHS packet to accurately position the correlation window when switching to the new master's timing after acknowledgment of the FHS packet.

After reception of the FHS packet acknowledgment, the new master A shall switch to its own timing with the sequence selection set to the *basic channel hopping sequence* and shall send a POLL packet to verify the switch. Both the master and the slave shall start a new timer with a time out of *newconnectionTO* on FHS packet acknowledgment. The start of this timer shall be aligned with the beginning of the first master TX slot boundary of the new piconet, following the FHS packet acknowledgment. The slave shall stop the timer when the POLL packet is received; the master shall stop the timer when the POLL packet is acknowledged. The slave shall respond with any type of packet to acknowledge the POLL. Any pending AFH_Instant shall be cancelled once the POLL packet has been received by the slave. If no response is received, the master shall re-send the POLL packet until *newconnectionTO* is reached. If this timer expires, both the slave and the master shall return to the old piconet timing with the old master and slave roles. Expiry of the timer shall also restore the state associated with AFH (including any pending AFH_Instant), Channel Quality Driven Data Rate (CQDDR, Link Manager Protocol [Part C] Section 4.1.7 on page 243) and power control (Link Manager Protocol [Part C] Section 4.1.3 on page 235). The procedure may then start again beginning at step 1. Aligning the timer with TX boundaries of the new piconet ensures that no device returns to the old piconet timing in the middle of a master RX slot.

After the role switch the ACL logical transport is reinitialized as if it were a new connection. For example, the SEQN of the first data packet containing a CRC on the new piconet channel shall be set according to the rules in section 7.6.2 on page 147.

A parked slave must be unparked before it can participate in a role switch.

8.6.6 Scatternet

Multiple piconets may cover the same area. Since each piconet has a different master, the piconets hop independently, each with their own hopping sequence and phase as determined by the respective master. In addition, the packets carried on the channels are preceded by different channel access codes as determined by the master device addresses. As more piconets are added, the probability of collisions increases; a graceful degradation of performance results as is common in frequency-hopping spread spectrum systems.

If multiple piconets cover the same area, a device can participate in two or more overlaying piconets by applying time multiplexing. To participate on the proper channel, it shall use the associated master device address and proper clock offset to obtain the correct phase. A device can act as a slave in several piconets, but only as a master in a single piconet: since two piconets with the same master are synchronized and use the same hopping sequence, they are one and the same piconet. A group of piconets in which connections exist between different piconets is called a *scatternet*.

A master or slave can become a slave in another piconet by being paged by the master of this other piconet. On the other hand, a device participating in one piconet can page the master or slave of another piconet. Since the paging device always starts out as master, a master-slave role switch is required if a slave role is desired. This is described in the Section 8.6.5 on page 175.

8.6.6.1 Inter-piconet communications

Time multiplexing must be used to switch between piconets. Devices may achieve the time multiplexing necessary to implement scatternet by using sniff mode or by remaining in an active ACL connection. For an ACL connection in piconets where the device is a slave in the **CONNECTION** state, it may choose not to listen in every master slot. In this case it should be recognized that the quality of service on this link may degrade abruptly if the slave is not present enough to match up with the master polling that slave. Similarly, in piconets where the device is master it may choose not to transmit in every master slot. In this case it is important to honor T_{poll} as much as possible. Devices in sniff mode may have sufficient time to visit another piconet in between sniff slots. When the device is a slave using sniff mode and there are not sufficient idle slots, the device may choose to not listen to all master transmission slots in the sniff_attempts period or during the subsequent sniff_timeout period. A master is not required to transmit during sniff slots and therefore has flexibility for scatternet. If SCO or eSCO links are established, other piconets shall only be visited in the non-reserved slots in between reserved slots. This is only possible if there is a single SCO logical transport using HV3 packets or eSCO links where at least four slots remain in between the reserved slots. Since the multiple piconets are not synchronized, guard time must be left to account for misalignment. This means that only 2 slots can effectively be used to visit another piconet in between the HV3 packets.

Since the clocks of two masters of different piconets are not synchronized, a slave device participating in two piconets shall maintain two offsets that, added to its own native clock, create the two master clocks. Since the two master clocks drift independently, the slave must regularly update the offsets in order to keep synchronization to both masters.

8.6.7 Hop sequence switching

Hop sequence adaptation is controlled by the master and can be set to either *enabled* or *disabled*. Once enabled, hop sequence adaptation shall apply to all logical transports on a physical link. Once enabled, the master may periodically update the set of *used* and *unused* channels as well as disable hop sequence adaptation on a physical link. When a master has multiple physical links the state of each link is independent of all other physical links.

When hop sequence adaptation is enabled, the *sequence selection* hop selection kernel input is set to *adapted channel hopping sequence* and the *AFH_channel_map* input is set to the appropriate set of *used* and *unused* channels. Additionally, the *same channel* mechanism shall be used. When hop sequence adaptation is enabled with all channels *used* this is known as AHS(79).

When hop sequence adaptation is disabled, the *sequence selection* input of the hop selection kernel is set to *basic channel hopping sequence* (the *AFH_channel_map* input is unused in this case) and the *same channel* mechanism shall not be used.

The hop sequence adaptation state shall be changed when the master sends the LMP_set_AFH PDU and a baseband acknowledgement is received. When the baseband acknowledgement is received prior to the hop sequence switch instant, *AFH_Instant*, (See Link Manager Protocol [Part C] Section 4.1.4 on page 237) the hop sequence proceeds as shown in Figure 8.9 on page 178.

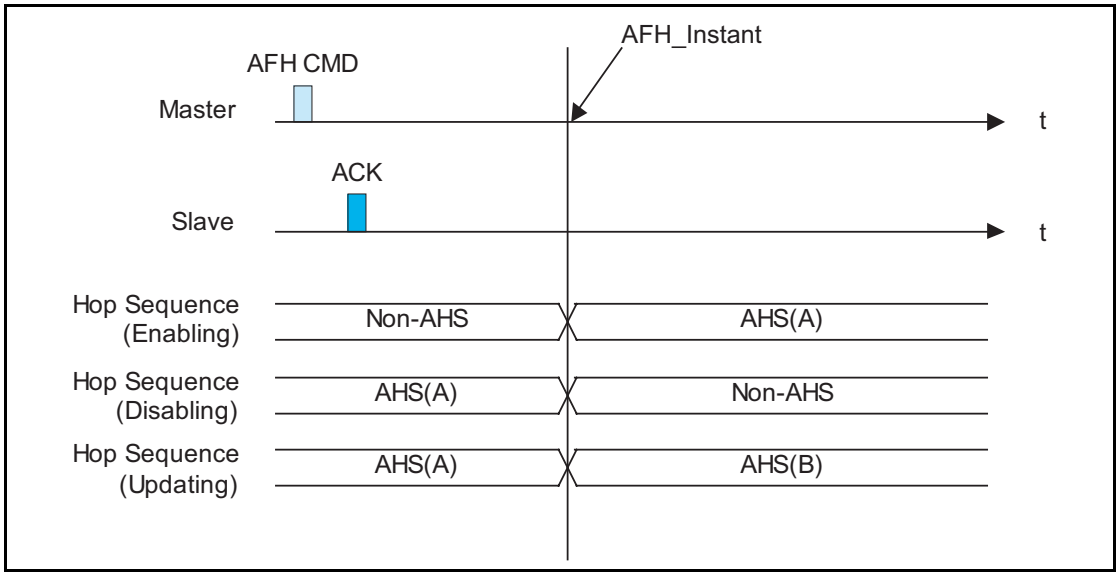


Figure 8.9: Successful hop sequence switching

When the baseband acknowledgement is not received prior to the *AFH_Instant* the master shall use a recovery hop sequence for the slave(s) that did not respond with an acknowledgement (this may be because the slave did not hear the master's transmission or the master did not hear the slave's transmission). When hop sequence adaptation is being enabled or disabled the recovery sequence shall be the *AFH_channel_map* specified in the LMP_set_AFH PDU. When the *AFH_channel_map* is being updated the master shall choose a recovery sequence that includes all of the RF channels marked as *used* in either the old or new *AFH_channel_map,* e.g. AHS(79). Once the baseband acknowledgement is received the master shall use the *AFH_channel_map* in the LMP_set_AFH PDU starting with the next transmission to the slave. See Figure 8.10 on page 179.

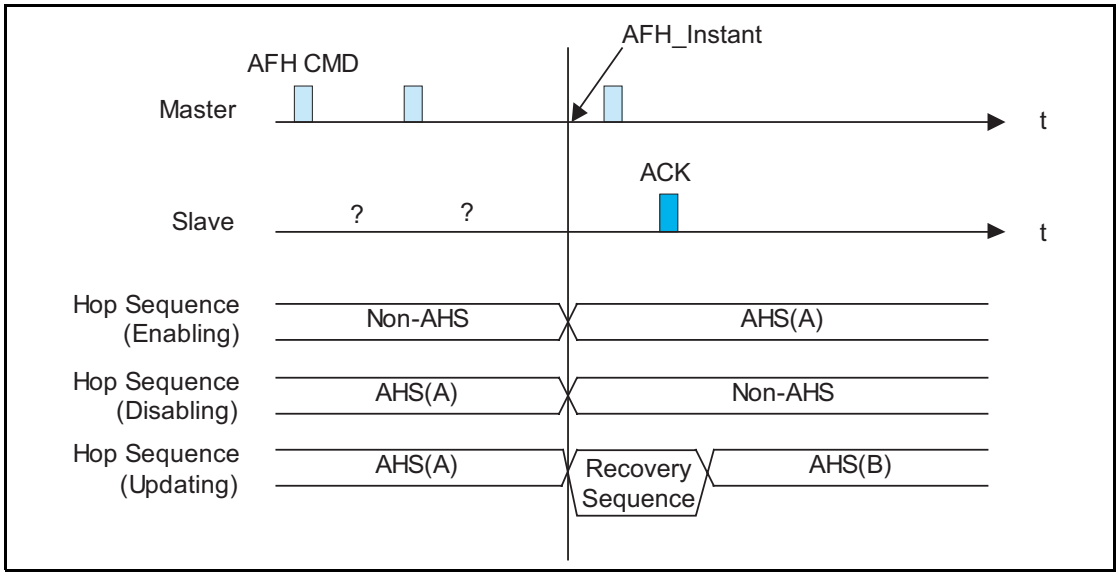


Figure 8.10: Recovery hop sequences

When the *AFH_Instant* occurs during a multi-slot packet transmitted by the master, the slave shall use the same hopping sequence parameters as the master used at the start of the multi-slot packet. An example of this is shown in Figure 8.11 on page 180. In this figure the *basic channel hopping sequence* is designated f. The first *adapted channel hopping sequence* is designated with f', and the second *adapted channel hopping sequence* is designated f".

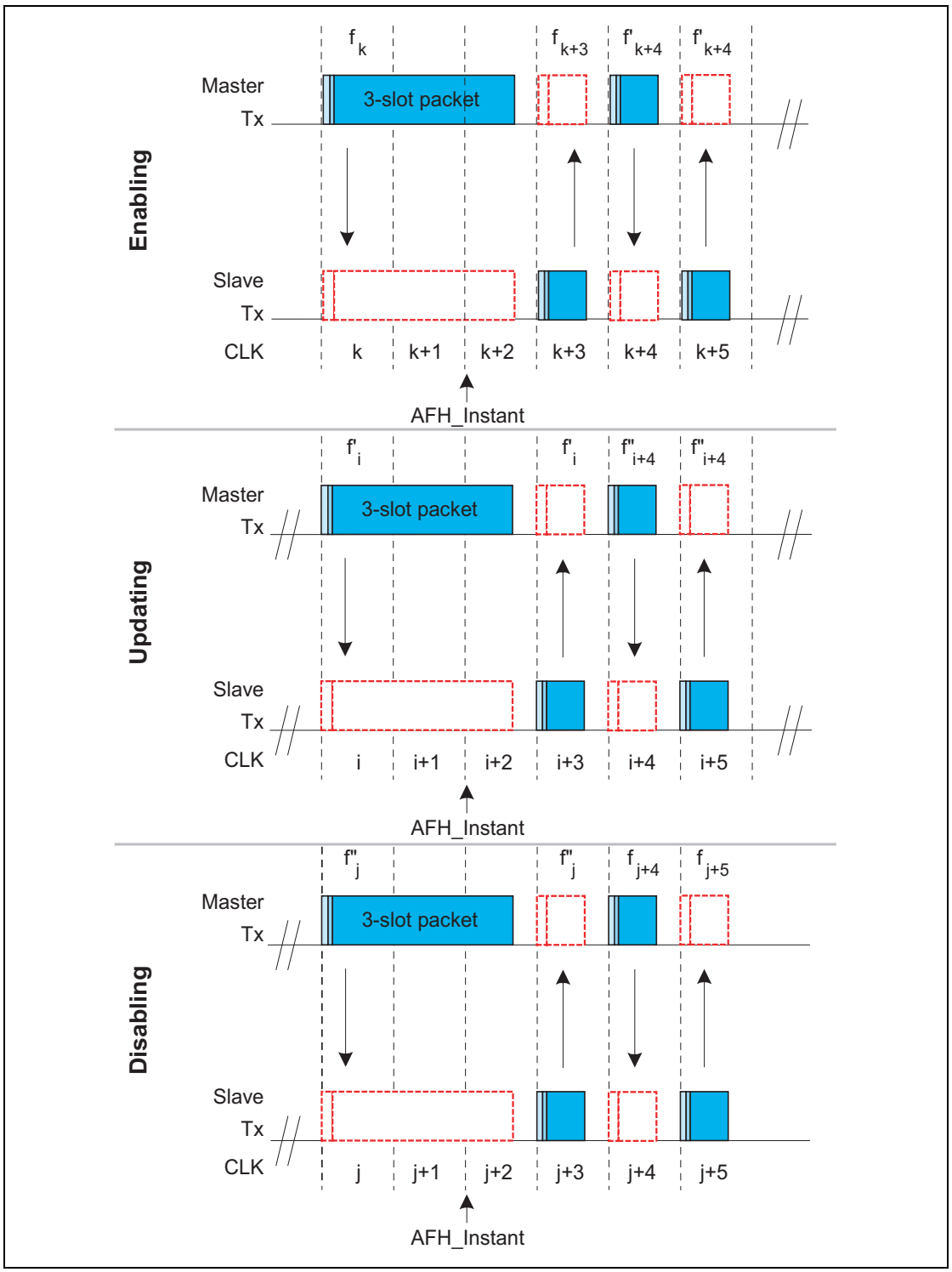


Figure 8.11: AFH_Instant changes during multi-slot packets transmitted by the master.

8.6.8 Channel classification and channel map selection

RF channels are classified as being *unknown, bad or good*. These classifications are determined individually by the master and slaves based on local information (e.g. active or passive channel assessment methods or from the Host via HCI). Information received from other devices via LMP (e.g. an *AFH_channel_map* from a master or a channel classification report from a slave) shall not be included in a device's channel classification.

The three possible channel classifications shall be as defined in Table 8.6 on page 181.

Classification	Definition
unknown	A channel shall be classified as *unknown* if the channel assessment measurements are insufficient to reliably classify the channel, and the channel is not marked as *bad* in the most recent HCI *Set_AFH_Channel_Classification*.
bad	A channel may be classified as *bad* if an ACL or synchronous throughput failure measure associated with it has exceeded a threshold (defined by the particular channel assessment algorithm employed). A channel may also be classified as *bad* if an interference-level measure associated with it, determining the interference level that the link poses upon other systems in the vicinity, has exceeded a threshold (defined by the particular channel assessment algorithm employed). A channel shall be classified as *bad* when it is marked as *bad* in the most recent HCI *Set_AFH_Channel_Classification* command.
good	A channel shall be classified as *good* if it is not either *unknown* or *bad*.

Table 8.6: Channel classification descriptions

A master with AFH enabled physical links shall determine an *AFH_channel_map* based on any combination of the following information:

- Channel classification from local measurements (e.g. active or passive channel assessment in the Controller), if supported and enabled. The Host may enable or disable local measurements using the HCI *Write_AFH_Channel_Classification_Mode* command, defined in the HCI Functional Specification [Part E] Section 7.3.58 on page 537 if HCI is present.
- Channel classification information from the Host using the HCI *Set_AFH_channel_classification* command, defined in the HCI Functional Specification [Part E] Section 7.3.58 on page 537 if HCI is present. Channels classified as *bad* in the most recent *AFH_Host_Channel_Classification* shall be marked as *unused* in the *AFH_channel_map*.
- Channel classification reports received from slaves in *LMP_channel_classification* PDUs, defined in the LMP Specification [Part C] Section 4.1.5 on page 240.

The algorithm used by the master to combine these information sources and generate the *AFH_channel_map* is not defined in the specification and will be implementation specific. At no time shall the number of channels used be less than N_{min}, defined in Section 2.3.1 on page 75.

If a master that determines that all channels should be *used* it may keep AFH operation enabled using an *AFH_channel_map* of 79 *used* channels, i.e. AHS(79).

8.6.9 Power Management

Features are provided to allow low-power operation. These features are both at the microscopic level when handling the packets, and at the macroscopic level when using certain operation modes.

8.6.9.1 Packet handling

In order to minimize power consumption, packet handling is minimized both at TX and RX sides. At the TX side, power is minimized by only sending useful data. This means that if only link control information needs to be exchanged, **NULL** packets may be used. No transmission is required if there is no link control information to be sent, or if the transmission would only involve a NAK (NAK is implicit on no reply). If there is data to be sent, the payload length shall be adapted in order to send only the valid data bytes. At the RX side, packet processing takes place in different steps. If no valid access code is found in the search window, the transceiver may return to sleep. If an access code is found, the receiver device shall start to process the packet header. If the HEC fails, the device may return to sleep after the packet header. A valid header indicates if a payload will follow and how many slots are involved.

8.6.9.2 Slot occupancy

As was described in Section 6.5 on page 118, the packet type indicates how many slots a packet may occupy. A slave not addressed in the packet header may go to sleep for the remaining slots the packet occupies. This can be read from the TYPE code.

8.6.9.3 Recommendations for low-power operation

The most common and flexible methods for reducing power consumption are the use of sniff and park. Hold can also be used by repeated negotiation of hold periods.

8.6.9.4 Enhanced Data Rate

Enhanced Data Rate provides power saving because of the ability to send a given amount of data in either fewer packets or with the same (or similar) number of packets but with shorter payloads.

8.7 SNIFF MODE

In sniff mode, the duty cycle of the slave's activity in the piconet may be reduced. If a slave is in active mode on an ACL logical transport, it shall listen in every ACL slot to the master traffic, unless that link is being treated as a scatternet link or is absent due to hold mode. With sniff mode, the time slots when a slave is listening are reduced, so the master shall only transmit to a slave in specified time slots. The sniff anchor points are spaced regularly with an interval of T_{sniff}.

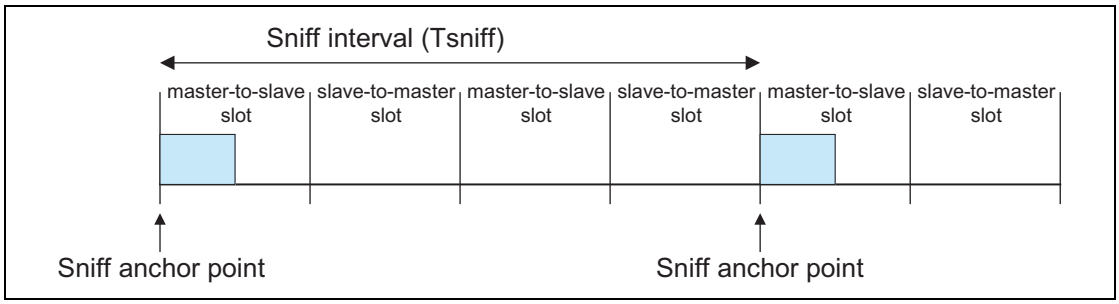


Figure 8.12: Sniff anchor points

The slave listens in master-to-slave transmission slots starting at the sniff anchor point. It shall use the following rules to determine whether to continue listening:

- If fewer than $N_{sniff\ attempt}$ master-to-slave transmission slots have elapsed since the sniff anchor point then the slave shall continue listening.
- If the slave has received a packet with a matching LT_ADDR that contains ACL data (DM, DH, DV, or AUX1 packets) in the preceding $N_{sniff\ timeout}$ master-to-slave transmission slots then it shall continue listening.
- If the slave has transmitted a packet containing ACL data (DM, DH, DV, or AUX1 packets) in the preceding $N_{sniff\ timeout}$ slave-to-master transmission slots then it shall continue listening.
- If the slave has received any packet with a matching LT_ADDR in the preceding $N_{sniff\ timeout}$ master-to-slave transmission slots then it may continue listening.
- A device may override the rules above and stop listening prior to $N_{sniff\ timeout}$ or the remaining $N_{sniff\ attempt}$ slots if it has activity in another piconet.

It is possible that activity from one sniff timeout may extend to the next sniff anchor point. Any activity from a previous sniff timeout shall not affect activity

after the next sniff anchor point. So in the above rules, only the slots since the last sniff anchor point are considered.

Note that $N_{sniff\ attempt}$ =1 and $N_{sniff\ timeout}$ =0 cause the slave to listen only at the slot beginning at the sniff anchor point, irrespective of packets received from the master.

$N_{sniff\ attempt}$ =0 shall not be used.

Sniff mode only applies to asynchronous logical transports and their associated LT_ADDR. sniff mode shall not apply to synchronous logical transports, therefore, both masters and slaves shall still respect the reserved slots and retransmission windows of synchronous links.

To enter sniff mode, the master or slave shall issue a sniff command via the LM protocol. This message includes the sniff interval T_{sniff} and an offset D_{sniff}. In addition, an initialization flag indicates whether initialization procedure 1 or 2 shall be used. The device shall use initialization 1 when the MSB of the current master clock (CLK_{27}) is 0; it shall use initialization 2 when the MSB of the current master clock (CLK_{27}) is 1. The slave shall apply the initialization method as indicated by the initialization flag irrespective of its clock bit value CLK_{27}. The sniff anchor point determined by the master and the slave shall be initialized on the slots for which the clock satisfies the applicable equation:

$CLK_{27-1} \bmod T_{sniff} = D_{sniff}$ for initialization 1

$(\overline{CLK}_{27}, CLK_{26-1}) \bmod T_{sniff} = D_{sniff}$ for initialization 2

this implies that D_{sniff} must be even

After initialization, the clock value CLK(k+1) for the next sniff anchor point shall be derived by adding the fixed interval T_{sniff} to the clock value of the current sniff anchor point:

$$CLK(k+1) = CLK(k) + T_{sniff}$$

8.7.1 Sniff Transition Mode

Sniff transition mode is a special mode which is used during the transition between sniff and active mode. It is required because during this transition it is unclear which mode (Sniff or Active) the slave is in and it is necessary to ensure that the slave is polled correctly regardless of which mode it is in.

In sniff transition mode the master shall maintain the active mode poll interval in case the slave is in active mode. In addition the master shall poll the slave at least once in the sniff attempt transmit slots starting at each sniff instant: note that this transmission counts for the active mode polling as well. The master must use its high power accurate clock when in Sniff Transition Mode.

The precise circumstances under which the master enters Sniff Transition Mode are defined in [Part C] Section 4.5.3.1 on page 279.

8.8 HOLD MODE

During the **CONNECTION** state, the ACL logical transport to a slave can be put in a **hold** mode. In **hold** mode the slave temporarily shall not support ACL packets on the channel. Any synchronous packet during reserved synchronous slots (from SCO and eSCO links) shall be supported. With the **hold** mode, capacity can be made free to do other things like scanning, paging, inquiring, or attending another piconet. The device in **hold** mode can also enter a low-power sleep mode. During **hold** mode, the slave device keeps its logical transport address(es) (LT_ADDR).

Prior to entering hold mode, master and slave agree on the time duration the slave remains in hold mode. A timer shall be initialized with the *holdTO* value. When the timer is expired, the slave shall wake up, synchronize to the traffic on the channel and will wait for further master transmissions.

8.9 PARK STATE

When a slave does not need to participate on the piconet channel, but still needs to remain synchronized to the channel, it can enter **PARK** state. **PARK** state is a state with very little activity in the slave. In the **PARK** state, the slave shall give up its logical transport address LT_ADDR. Instead, it shall receive two new addresses to be used in the **PARK** state

- PM_ADDR: 8-bit Parked Member Address
- AR_ADDR: 8-bit Access Request Address

The PM_ADDR distinguishes a parked slave from the other parked slaves. This address may be used in the master-initiated unpark procedure. In addition to the PM_ADDR, a parked slave may also be unparked by its 48-bit BD_ADDR. The all-zero PM_ADDR is a reserved address: if a parked device has the all-zero PM_ADDR it can only be unparked by the BD_ADDR. In that case, the PM_ADDR has no meaning. The AR_ADDR shall be used by the slave in the slave-initiated unpark procedure. All messages sent to the parked slaves are carried by broadcast packets.

The parked slave wakes up at regular intervals to listen to the channel in order to re-synchronize and to check for broadcast messages. To support the synchronization and channel access of the parked slaves, the master supports a beacon train described in the next section. The beacon structure is communicated to the slave when it is parked. When the beacon structure changes, the parked slaves are updated through broadcast messages.

The master shall maintain separate non-overlapping park beacon structures for each hop sequence. The beacon structures shall not overlap either their beacon slots or access windows.

In addition for using it for low power consumption, park is used to connect more than seven slaves to a single master. At any one time, only seven slaves can be in the **CONNECTION** state. However, by swapping active and parked slaves out respectively in the piconet, the number of slaves can be much larger (255 if the PM_ADDR is used, and an arbitrarily large number if the BD_ADDR is used).

8.9.1 Beacon train

To support parked slaves, the master establishes a beacon train when one or more slaves are parked. The beacon train consists of one beacon slot or a train of equidistant beacon slots which is transmitted periodically with a constant time interval. The beacon train is illustrated in Figure 8.13 on page 188. A train of N_B ($N_B \geq 1$) beacon slots is defined with an interval of T_B slots. The beacon slots in the train are separated by Δ_B. The start of the first beacon slot is referred to as the **beacon instant** and serves as the beacon timing reference. The beacon parameters N_B and T_B are chosen such that there are sufficient beacon slots for a parked slave to synchronize to during a certain time window in an error-prone environment.

When parked, the slave shall receive the beacon parameters through an LMP command. In addition, the timing of the beacon instant is indicated through the offset D_B. As with the SCO logical transport (see Section 8.6.2 on page 169), two initialization procedures 1 or 2 are used. The master shall use initialization 1 when the MSB of the current master clock (CLK_{27}) is 0; it shall use initialization 2 when the MSB of the current master clock (CLK_{27}) is 1. The chosen initialization procedure shall also be carried by an initialization flag in the LMP command. The slave shall apply the initialization method as indicated by the initialization flag irrespective of its clock bit CLK_{27}. The master-to-slave slot positioned at the beacon instant shall be initialized on the slots for which the clock satisfies the applicable equation:

$$CLK_{27-1} \bmod T_B = D_B \qquad \text{for initialization 1}$$

$$(\overline{CLK_{27}}, CLK_{26-1}) \bmod T_B = D_B \qquad \text{for initialization 2}$$

this implies that DB will be even

After initialization, the clock value CLK(k+1) for the next beacon instant shall be derived by adding the fixed interval T_B to the clock value of the current beacon instant:

$$CLK(k+1) = CLK(k) + T_B$$

The beacon train serves four purposes:

1. transmission of master-to-slave packets which the parked slaves can use for re-synchronization
2. carrying messages to the parked slaves to change the beacon parameters

3. carrying general broadcast messages to the parked slaves
4. unparking of one or more parked slaves

Since a slave can synchronize to any packet which is preceded by the proper channel access code, the packets carried on the beacon slots do not have to contain specific broadcast packets for parked slaves to be able to synchronize; any packet may be used. The only requirement placed on the beacon slots is that there is a master-to-slave transmission present on the hopping sequence associated with the park structure. If there is no information to be sent, **NULL** packets may be transmitted by the master. If there is indeed broadcast information to be sent to the parked slaves, the first packet of the broadcast message shall be repeated in every beacon slot of the beacon train. However, synchronous traffic in the synchronous reserved slots may interrupt the beacon transmission if it is on the same hopping sequence as the parked slaves. The master shall configure its park beacon structure so that reserved slots of synchronous logical transports do not cause slaves to miss synchronization on a beacon slot. For example, a master that has active slaves using AHS, and parked slaves using Non-AHS shall ensure that the Park beacons cannot be interrupted by AHS synchronous reserved slots.

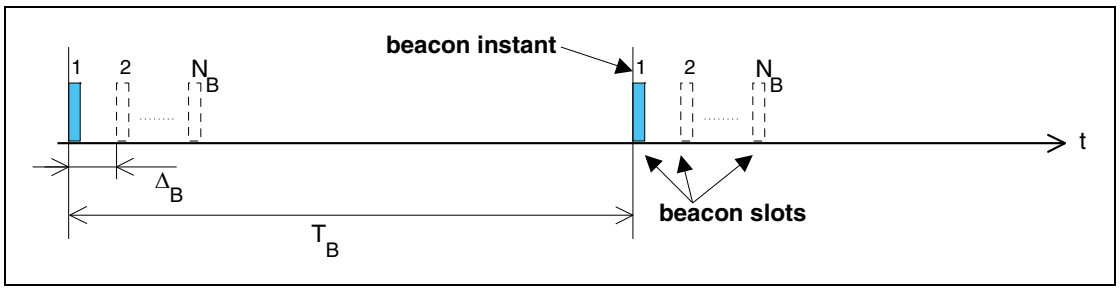


Figure 8.13: General beacon train format

The master can place parked slaves in any of the AFH operating modes, but shall ensure that all parked slaves use the same hop sequence. Masters should use AHS(79) or AHS when all the slaves in the Piconet are AFH capable.

A master that switches a slave between AFH enabled, AFH disabled or AHS(79) operation shall ensure that the AFH_Instant occurs before transmission of the beacon train using this hop sequence.

The master communicates with parked slaves using broadcast messages. Since these messages can be time - critical, an ongoing repetition train of broadcast message may be prematurely aborted by broadcast information destined to parked slaves in beacon slots and in access windows (see Section 8.9.2 on page 189).

8.9.2 Beacon access window

In addition to the beacon slots, an access window is defined where the parked slaves can send requests to be unparked. To increase reliability, the access window may be repeated M_{access} times ($M_{access} \geq 1$), see Figure 8.14 on page 189. The access window starts a fixed delay D_{access} after the beacon instant. The width of the access window is T_{access}.

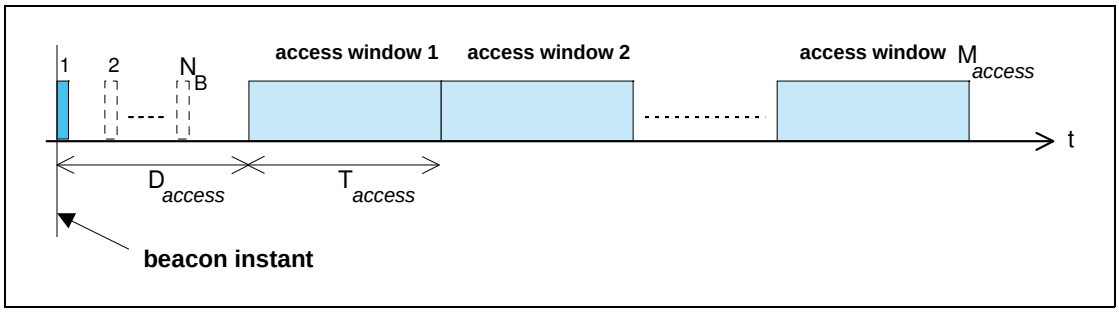


Figure 8.14: Definition of access window

The access window supports a polling slave access technique. The format of the polling technique is shown in Figure 8.15 on page 189. The same TDD structure is used as on the piconet channel, i.e. master-to-slave transmission is alternated with slave-to-master transmission. The slave-to-master slot is divided into two half slots of 312.5 µs each. The half slot a parked slave is allowed to respond in corresponds to its access request address (AR_ADDR), see also section 8.9.6 on page 192. For counting the half slots to determine the access request slot, the start of the access window is used, see Figure 8.15 on page 189. The slave shall only send an access request in the proper slave-to-master half slot if a broadcast packet has been received in the preceding master-to-slave slot. In this way, the master polls the parked slaves.

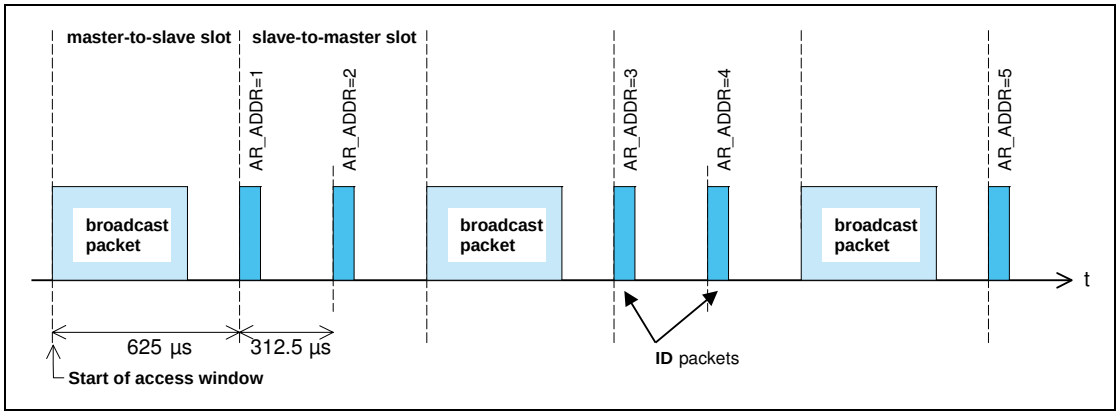


Figure 8.15: Access procedure applying the polling technique.

The slots of the access window may also be used for traffic on the piconet if required. For example, if a synchronous connection has to be supported, the slots reserved for the synchronous link may carry synchronous information

instead of being used for access requests, i.e. if the master-to-slave slot in the access window contains a packet different from a broadcast packet, the following slave-to-master slot shall not be used for slave access requests. If the master transmits a broadcast packet in the access window then it shall use the hop sequence associated with the park structure. Slots in the access window not affected by piconet traffic may still be used according to the defined access structure, (an example is shown in Figure 8.16 on page 190) the access procedure shall be continued as if no interruption had taken place.

When the slave is parked, the master shall indicate what type of access scheme will be used. For the polling scheme, the number of slave-to-master access slots N_{acc_slot} is indicated.

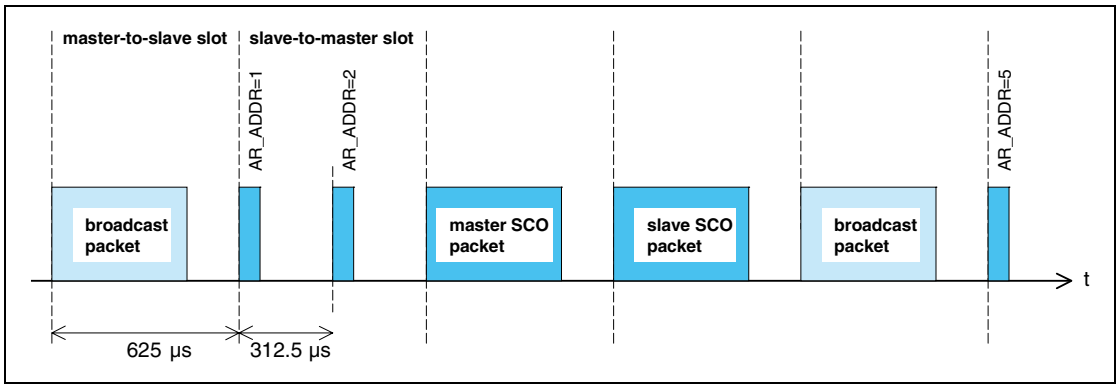


Figure 8.16: Disturbance of access window by SCO traffic

By default, the access window is always present. However, its activation depends on the master sending broadcast messages to the slave at the appropriate slots in the access window. A flag in a broadcast LMP message within the beacon slots may indicate that the access window(s) belonging to this instant will not be activated. This prevents unnecessary scanning of parked slaves that want to request access.

8.9.3 Parked slave synchronization

Parked slaves wake up periodically to re-synchronize to the channel. Any packet exchanged on the channel can be used for synchronization. Since master transmission is mandatory on the beacon slots, parked slaves will use the beacon train to re-synchronize. A parked slave may wake-up at the beacon instant to read the packet sent on the first beacon slot. If this fails, it may retry on the next beacon slot in the beacon train; in total, there are N_B opportunities per beacon instant to re-synchronize. During the search, the slave may increase its search window, see also Section 2.2.5.2 on page 74. The separation between the beacon slots in the beacon train Δ_B shall be chosen such that consecutive search windows will not overlap.

The parked slave may not wake up at every beacon instant. Instead, a sleep interval may be applied which is longer than the beacon interval T_B, see Figure 8.17 on page 191. The slave sleep window shall be a multiple N_{B_sleep} of T_B.

The precise beacon instant the slave may wake up on shall be indicated by the master with D_{B_sleep} which indicates the offset (in multiples of T_B) with respect to the beacon instant ($0 < D_{B\_sleep} < N_{B\_sleep}-1$). To initialize the wake-up period, the applicable equation shall be used:

$$CLK_{27-1} \bmod (N_{B\_sleep} \bullet T_B) = D_B + D_{B\_sleep} \bullet T_B \quad \text{for initialization 1}$$

$$(\overline{CLK_{27}}, CLK_{26-1}) \bmod (N_{B\_sleep} \bullet T_B) = D_B + D_{B\_sleep} \bullet T_B \quad \text{for initialization 2}$$

where initialization 1 shall be chosen by the master if the MSB in the current master clock is 0 and initialization 2 shall be chosen by the master if the MSB in the current master clock is 1.

When the master needs to send broadcast messages to the parked slaves, it may use the beacon slots for these broadcast messages. However, if $N_B < N_{BC}$, the slots following the last beacon slot in the beacon train shall be used for the remaining $N_{BC}-N_B$ broadcast packets. If $N_B > N_{BC}$, the broadcast message shall be repeated on all N_B beacon slots.

A parked slave shall read the broadcast messages sent in the beacon slot(s) it wakes up in. If the parked slave wakes up, the minimum wake-up activity shall be to read the channel access code for re-synchronization and the packet header to check for broadcast messages.

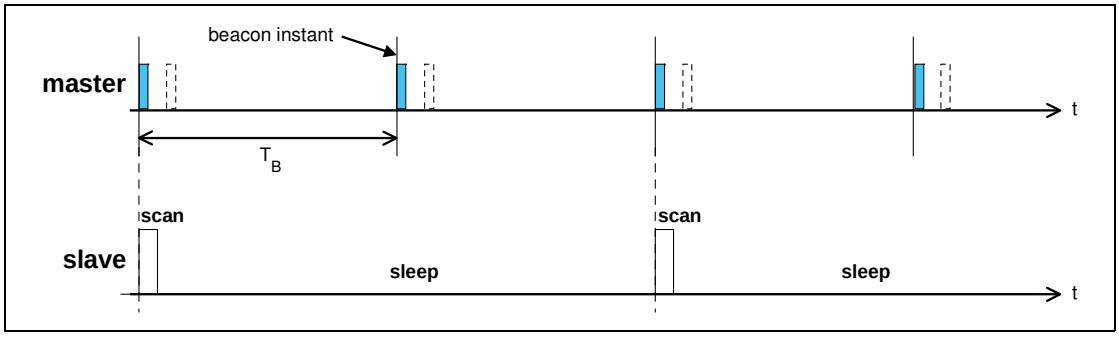


Figure 8.17: Extended sleep interval of parked slaves.

8.9.4 Parking

A master can park an active slave through the exchange of LMP commands. Before being put into park, the slave shall be assigned a PM_ADDR and an AR_ADDR. Every parked slave shall have a unique PM_ADDR or a PM_ADDR of 0. The AR_ADDR is not necessarily unique. The beacon parameters shall be given by the master when the slave is parked. The slave shall then give up its LT_ADDR and shall enter **PARK** state. A master can park only a single slave at a time. The park message is carried with a normal data packet and addresses the slave through its LT_ADDR.

8.9.5 Master-initiated unparking

The master can unpark a parked slave by sending a dedicated LMP unpark command including the parked slave's address. This message shall be sent in a broadcast packet on the beacon slots. The master shall use either the slave's PM_ADDR, or its BD_ADDR. The message also includes the logical transport address LT_ADDR the slave shall use after it has re-entered the piconet. The unpark message may include a number of slave addresses so that multiple slaves may be unparked simultaneously. For each slave, a different LT_ADDR shall be assigned.

After having received the unpark message, the parked slave matching the PM_ADDR or BD_ADDR shall leave the **PARK** state and enter the **CONNECTION** state. It shall keep listening to the master until it is addressed by the master through its LT_ADDR. The first packet sent by the master shall be a POLL packet. The return packet in response to the POLL packet confirms that the slave has been unparked. If no response packets from the slave is received for *newconnectionTO* number of slots after the end of beacon repetition period, the master shall unpark the slave again. The master shall use the same LT_ADDR on each unpark attempt until it has received a link supervision timeout for that slave or the unpark has completed successfully. If the slave does not receive the POLL packet for *newconnectionTO* number of slots after the end of beacon repetition period, it shall return to park, with the same beacon parameters. After confirming that the slave is in the **CONNECTION** state, the master decides in which mode the slave will continue.

When a device is unparked, the SEQN bit for the link shall be reset to 1 on both the master and the slave (see Section 7.6.2.1 on page 148).

8.9.6 Slave-initiated unparking

A slave can request access to the channel through the access window defined in section 8.9.2 on page 189. As shown in Figure 8.15 on page 189, the access window includes several slave-to-master half slots where the slave may send an access request message. The specific half slot the slave is allowed to respond in, corresponds to its access request address (AR_ADDR) which it received when it was parked. The order of the half slots (in Figure 8.15 the AR_ADDR numbers linearly increase from 1 to 5) is not fixed: an LMP command sent in the beacon slots may reconfigure the access window. When a slave desires access to the channel, it shall send an access request message in the proper slave-to-master half slot. The access request message of the slave is the **ID** packet containing the device access code (DAC) of the master (which is the channel access code without the trailer). The parked slave shall only transmit an access request message in the half slot, when in the preceding master-to-slave slot a broadcast packet has been received. This broadcast message may contain any kind of broadcast information not necessarily related to the parked slave(s). If no broadcast information is available, a broadcast **NULL** or broadcast **POLL** packet shall be sent.

After having sent an access request, the parked slave shall listen for an unpark message from the master. As long as no unpark message is received, the slave shall repeat the access requests in the subsequent access windows. After the last access window (there are M_{access} windows in total, see Section 8.9.2 on page 189), the parked slave shall listen for an additional N_{poll} time slots for an unpark message. If no unpark message is received within N_{poll} slots after the end of the last access window, the slave may return to sleep and retry an access attempt after the next beacon instant.

After having received the unpark message, the parked slave matching the PM_ADDR or BD_ADDR shall leave the **PARK** state and enter the **CONNECTION** state. It shall keep listening to the master until it is addressed by the master through its LT_ADDR. The first packet sent by the master shall be a POLL packet. The return packet in response to the POLL packet confirms that the slave has been unparked. After confirming that the slave is in the **CONNECTION** state, the master decides in which mode the slave will continue. If no response packet from the slave is received for *newconnectionTO* number of slots after N_{poll} slots after the end of the last access window, the master shall send the unpark message to the slave again. If the slave does not receive the POLL packet for *newconnectionTO* number of slots after N_{poll} slots after the end of the last access window, it shall return to park, with the same beacon parameters.

When a device is unparked, the SEQN bit for the link shall be reset to 1 on both the master and the slave (see Section 7.6.2.1 on page 148).

8.9.7 Broadcast scan window

In the beacon train, the master can support broadcast messages to the parked slaves. However, it may extend its broadcast capacity by indicating to the parked slaves that more broadcast information is following after the beacon train. This is achieved by an LMP command ordering the parked slaves (as well as the active slaves) to listen to the channel for broadcast messages during a limited time window. This time window starts at the beacon instant and continues for the period indicated in the LMP command sent in the beacon train.

8.9.8 Polling in the park state

In the **PARK** state, parked slaves may send access requests in the access window provided a broadcast packet is received in the preceding master-to-slave slot. Slaves in the **CONNECTION** state shall not send in the slave-to-master slots following the broadcast packet, since they are only allowed to send if addressed specifically.

9 AUDIO

On the air-interface, either a 64 kb/s log PCM (Pulse Code Modulation) format (A-law or μ-law) may be used, or a 64 kb/s CVSD (Continuous Variable Slope Delta Modulation) may be used. The latter format applies an adaptive delta modulation algorithm with syllabic companding.

The voice coding on the line interface is designed to have a quality equal to or better than the quality of 64 kb/s log PCM.

Table 9.1 on page 195 summarizes the voice coding schemes supported on the air interface. The appropriate voice coding scheme is selected after negotiations between the Link Managers.

Voice Codecs	
linear	CVSD
8-bit logarithmic	A-law
	μ-law

Table 9.1: Voice coding schemes supported on the air interface.

9.1 LOG PCM CODEC

Since the synchronous logical transports on the air-interface can support a 64 kb/s information stream, a 64 kb/s log PCM traffic can be used for transmission. Either A-law or μ-law compression may be applied. In the event that the line interface uses A-law and the air interface uses μ-law or vice versa, a conversion from A-law to μ-law shall be performed. The compression method shall follow ITU-T recommendations G. 711.

9.2 CVSD CODEC

A more robust format for voice over the air interface is delta modulation. This modulation scheme follows the waveform where the output bits indicate whether the prediction value is smaller or larger then the input waveform. To reduce slope overload effects, syllabic companding is applied: the step size is adapted according to the average signal slope. The input to the CVSD encoder shall be 64 ksamples/s linear PCM (typically 16 bits, but actual value is implementation specific). Block diagrams of the CVSD encoder and CVSD decoder are shown in Figure 9.1 on page 196, Figure 9.2 on page 196 and Figure 9.3 on page 196. The system shall be clocked at 64 kHz.

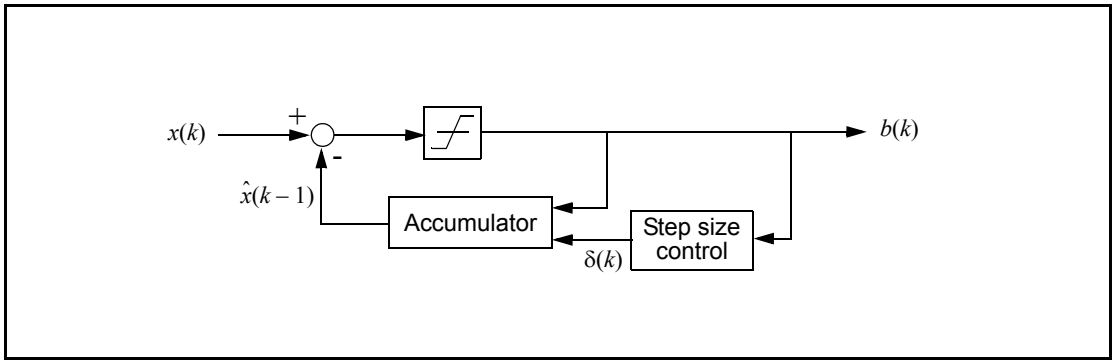


Figure 9.1: Block diagram of CVSD encoder with syllabic companding.

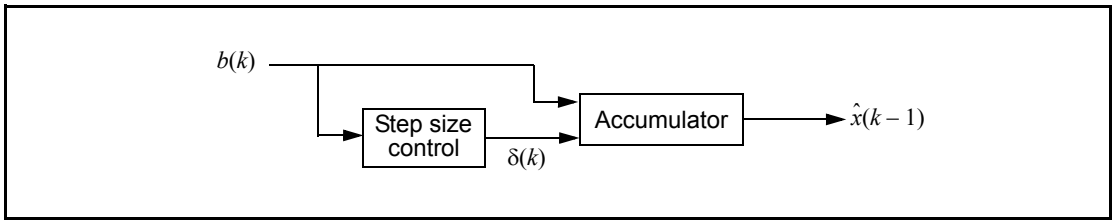


Figure 9.2: Block diagram of CVSD decoder with syllabic companding.

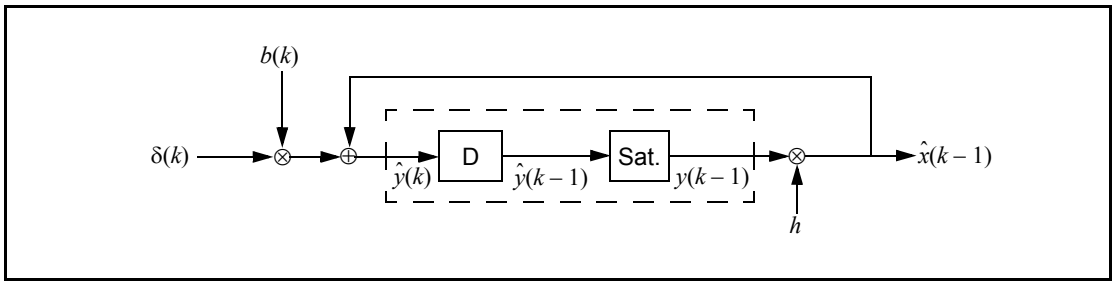


Figure 9.3: Accumulator procedure.

Let $\mathrm{sgn}(x) = 1$ for $x \geq 0$, otherwise $\mathrm{sgn}(x) = -1$. On air these numbers shall be represented by the sign bit; i.e. negative numbers are mapped on “1” and positive numbers are mapped on “0”.

Denote the CVSD encoder output bit $b(k)$, the encoder input $x(k)$, the accumulator contents $y(k)$, and the step size $\delta(k)$. Furthermore, let *h* be the decay factor for the accumulator, let β denote the decay factor for the step size, and, let α be the syllabic companding parameter. The latter parameter monitors the slope by considering the *K* most recent output bits

Let

$$\hat{x}(k) = hy(k). \qquad \text{(EQ 13)}$$

Then, the CVSD encoder internal state shall be updated according to the following set of equations:

$$b(k) = \mathrm{sgn}\{x(k) - \hat{x}(k-1)\}, \quad \text{(EQ 14)}$$

$$\alpha = \begin{cases} 1, & \text{if } J \text{ bits in the last } K \text{ output bits are equal,} \\ 0, & \text{otherwise,} \end{cases} \quad \text{(EQ 15)}$$

$$\delta(k) = \begin{cases} \min\{\delta(k-1) + \delta_{min}, \delta_{max}\}, & \alpha = 1, \\ \max\{\beta\delta(k-1), \delta_{min}\}, & \alpha = 0, \end{cases} \quad \text{(EQ 16)}$$

$$y(k) = \begin{cases} \min\{\hat{y}(k), y_{max}\}, & \hat{y}(k) \geq 0. \\ \max\{\hat{y}(k), y_{min}\}, & \hat{y}(k) < 0. \end{cases} \quad \text{(EQ 17)}$$

where

$$\hat{y}(k) = \hat{x}(k-1) + b(k)\delta(k). \quad \text{(EQ 18)}$$

In these equations, δ_{min} and δ_{max} are the minimum and maximum step sizes, and, y_{min} and y_{max} are the accumulator's negative and positive saturation values, respectively. Over air, the bits shall be sent in the same order they are generated by the CVSD encoder.

For a 64 kb/s CVSD, the parameters as shown in Table 9.2 shall be used. The numbers are based on a 16 bit signed number output from the accumulator. These values result in a time constant of 0.5 ms for the accumulator decay, and a time constant of 16 ms for the step size decay

Parameter	Value
h	$1 - \frac{1}{32}$
β	$1 - \frac{1}{1024}$
J	4
K	4
δ_{min}	10
δ_{max}	1280
y_{min}	-2^{15} or $-2^{15} + 1$
y_{max}	$2^{15} - 1$

Table 9.2: CVSD parameter values. The values are based on a 16 bit signed number output from the accumulator.

9.3 ERROR HANDLING

In the DV, HV3, EV3, EV5, 2-EV3, 3-EV3, 2-EV5 and 3-EV5 packets, the voice is not protected by FEC. The quality of the voice in an error-prone environment then depends on the robustness of the voice coding scheme and, in the case of eSCO, the retransmission scheme. CVSD, in particular, is rather insensitive to random bit errors, which are experienced as white background noise. However, when a packet is rejected because either the channel access code, the HEC test was unsuccessful, or the CRC has failed, measures have to be taken to “fill” in the lost speech segment.

The voice payload in the **HV2** and **EV4** packets are protected by a 2/3 rate FEC. For errors that are detected but cannot be corrected, the receiver should try to minimize the audible effects. For instance, from the 15-bit FEC segment with uncorrected errors, the 10-bit information part as found before the FEC decoder should be used. The **HV1** packet is protected by a 3 bit repetition FEC. For this code, the decoding scheme will always assume zero or one-bit errors. Thus, there exist no detectable but uncorrectable error events for **HV1** packets.

9.4 GENERAL AUDIO REQUIREMENTS

9.4.1 Signal levels

For A-law and μ-law log-PCM encoded signals the requirements on signal levels shall follow the ITU-T recommendation G.711.

Full swing at the 16 bit linear PCM interface to the CVSD encoder is defined to be 3 dBm0.

9.4.2 CVSD audio quality

For Bluetooth audio quality the requirements are put on the transmitter side. The 64 ksamples/s linear PCM input signal should have negligible spectral power density above 4 kHz. The power spectral density in the 4-32 kHz band of the decoded signal at the 64 ksample/s linear PCM output, should be more than 20 dB below the maximum in the 0-4 kHz range.

10 LIST OF FIGURES

11 LIST OF TABLES

APPENDIX A: GENERAL AUDIO RECOMMENDATIONS

MAXIMUM SOUND PRESSURE

It is the sole responsibility of each manufacturer to design their audio products in a safe way with regards to injury to the human ear. The Bluetooth Specification doesn't specify maximum sound pressure from an audio device.

OTHER TELEPHONY NETWORK REQUIREMENTS

It is the sole responsibility of each manufacturer to design the Bluetooth audio product so that it meets the regulatory requirements of all telephony networks that it may be connected to.

AUDIO LEVELS FOR BLUETOOTH

Audio levels shall be calculated as Send Loudness Rating, SLR, and Receive Loudness Rating, RLR. The calculation methods are specified in ITU-T Recommendation P.79.

The physical test set-up for Handsets and Headsets is described in ITU-T Recommendation P.51 and P.57

The physical test set-up for speakerphones and "Vehicle handsfree systems" is specified in ITU-T Recommendation P.34.

A general equation for computation of loudness rating (LR) for telephone sets is given by ITU-T recommendations P.79 and is given by

$$LR = -\frac{10}{m}\log_{10}\left(\sum_{i=N_1}^{N_2} 10^{m(s_i - w_i)/10}\right), \qquad \text{(EQ 19)}$$

where

m is a constant (~ 0.2).

w_i = weighting coefficient (different for the various LRs).

S_i = the sensitivity at frequency F_i, of the electro-acoustic path

N_1,N_2, consecutive filter bank numbers (Art. Index: 200-4000 Hz)

(EQ 19) on page 204 is used for calculating the (SLR) according to Figure 12.1 on page 205, and (RLR) according to Figure 12.2 on page 205. When calculating LRs one must only include those parts of the frequency band where an actual signal transmission can occur in order to ensure that the additive property of LRs is retained. Therefore ITU-T P.79 uses only the frequency band 200-4000 Hz in LR computations.

MICROPHONE PATH

SLR measurement model

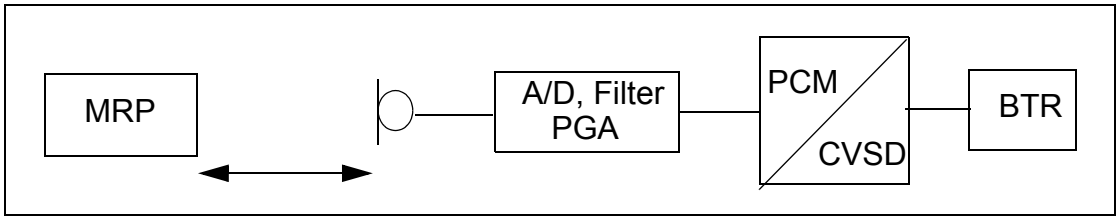


Figure 12.1: SLR measurement set-up.

LOUDSPEAKER PATH

RLR measurement model

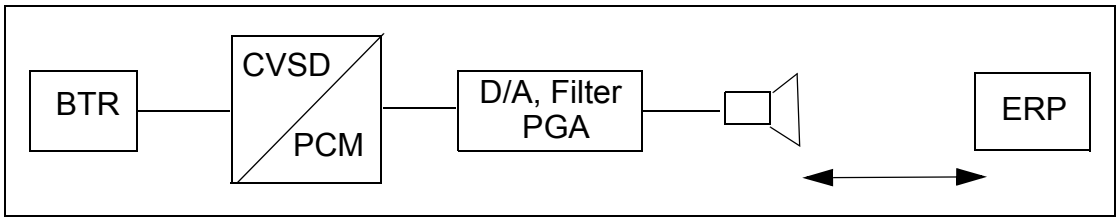


Figure 12.2: RLR measurement set-up.

BLUETOOTH VOICE INTERFACE

The specification for the Bluetooth voice interface should follow in the first place the *ITU-T Recommendations P.79*, which specifies the loudness ratings for telephone sets. These recommendations give general guidelines and specific algorithms used for calculating the loudness ratings of the audio signal with respect to Ear Reference Point (ERP).

For Bluetooth voice interface to the different cellular system terminals, loudness and frequency recommendations based on the cellular standards should be used. For example, GSM 03.50 gives recommendation for both the loudness ratings and frequency mask for a GSM terminal interconnection with Bluetooth.

1- The output of the CVSD decoder are 16-bit linear PCM digital samples, at a sampling frequency of 8 ksample/second. Bluetooth also supports 8-bit log PCM samples of A-law and μ-law type. The sound pressure at the ear reference point for a given 16-bit CVSD sample, should follow the sound pressure level given in the cellular standard specification.

2- A maximum sound pressure which can be represented by a 16-bit linear PCM sample at the output of the CVSD decoder should be specified according to the loudness rating, in ITU P.79 and at PGA value of 0 dB. Programmable Gain Amplifiers (PGAs) are used to control the audio level at the terminals by the user. For conversion between various PCM representations: A-law, μ-law and linear PCM, ITU-T G.711, G.712, G.714 give guidelines and PCM value relationships. Zero-code suppression based on ITU-T G.711 is also recommended to avoid network mismatches.

FREQUENCY MASK

For interfacing a Bluetooth terminal to a digital cellular mobile terminal, a compliance of the CVSD decoder signal to the frequency mask given in the cellular standard, is recommended to guarantee correct function of the speech coders. A recommendation for a frequency mask is given in the Table 12.1 below. The Figure 12.3 below shows a plot of the frequency mask for Bluetooth (solid line). The GSM frequency mask (dotted line) is shown in Figure 12.3 for comparison.

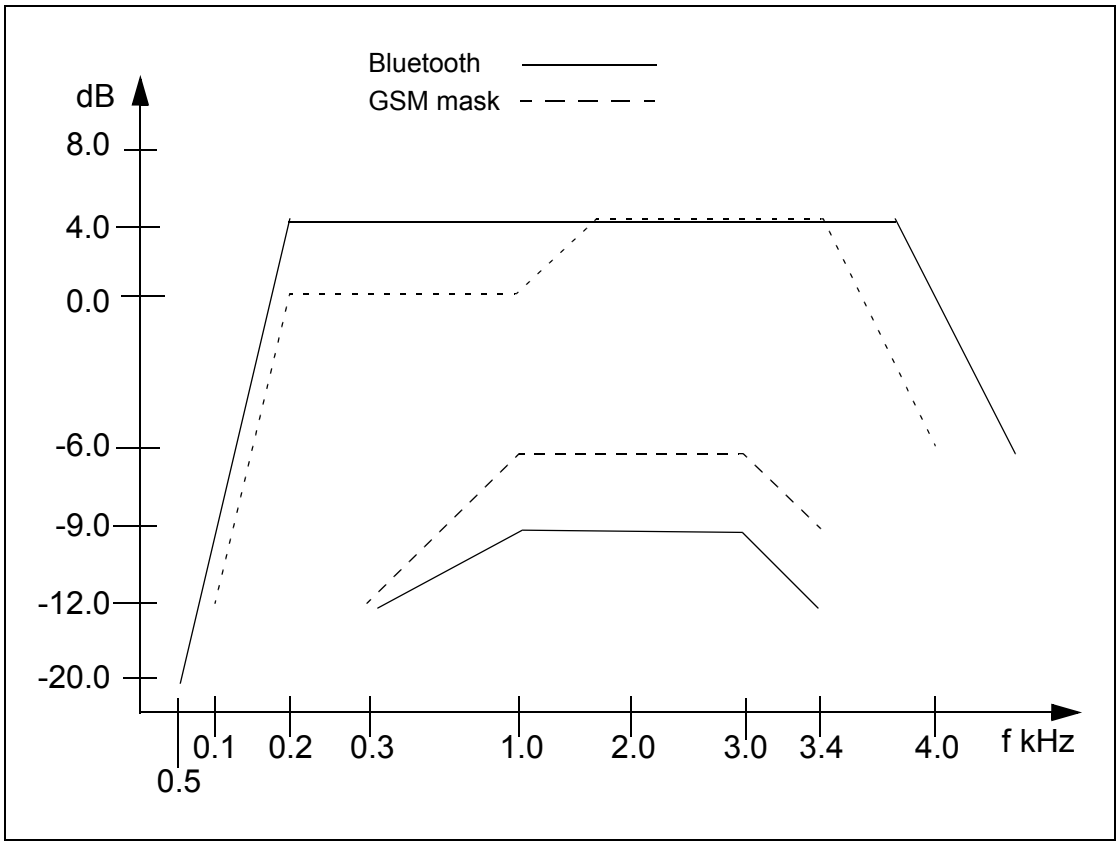


Figure 12.3: Plot of recommended frequency mask for Bluetooth. The GSM send frequency mask is given for comparison (dotted line)

Frequency (Hz)	Upper Limit (dB)	Lower Limit (dB)
50	-20	-
300	4	-12
1000	4	-9
2000	4	-9
3000	4	--9
3400	4	-12
4000	0	-

Table 12.1: Recommended Frequency Mask for Bluetooth

APPENDIX B: TIMERS

This appendix contains a list of Baseband timers related to inactivity timeout defined in this specification. Definitions and default values of the timers are listed below

All timer values are given in slots.

LIST OF TIMERS

inquiryTO

The *inquiryTO* defines the number of slots the **inquiry** substate will last. The timer value may be changed by the host. HCI provides a a command to change the timer value.

pageTO

The *pageTO* defines the number of slots the **page** substate can last before a response is received. The timer value may be changed by the host. HCI provides a a command to change the timer value.

pagerespTO

In the slave, it defines the number of slots the slave awaits the master's response, FHS packet, after sending the page acknowledgment ID packet. In the master, *pagerespTO* defines the number of slots the master should wait for the FHS packet acknowledgment before returning to **page** substate. Both master and slave units should use the same value for this timeout, to ensure common page/scan intervals after reaching *pagerespTO*.

The *pagerespTO*value is 8 slots.

newconnectionTO

Every time a new connection is started through paging, scanning, role switch or unparking, the master sends a POLL packet as the first packet in the new connection. Transmission and acknowledgment of this POLL packet is used to confirm the new connection. If the POLL packet is not received by the slave or the response packet is not received by the master for *newconnectionTO* number of slots, both the master and the slave will return to the previous substate.

newconnectionTO value is 32 slots.

supervisionTO

The *supervisionTO* is used by both the master and slave to monitor link loss. If a device does not receive any packets that pass the HEC check and have the proper LT_ADDR for a period of *supervisionTO*, it will reset the link. The supervision timer keeps running in hold mode, sniff mode and park state.

The *supervisionTO* value may be changed by the host. HCI provides a a command to change the timer value. At the baseband level a default value that is equivalent to 20 seconds will be used.

APPENDIX C: RECOMMENDATIONS FOR AFH OPERATION IN PARK, HOLD AND SNIFF

The three possible AFH operation modes for an AFH capable slave in park, hold and sniff are the same three AFH operation modes used during **CONNECTION** state:

- *Enabled* (using the same AHS as slaves in the **CONNECTION** state)
- *Enabled* (using AHS(79))
- *Disabled*

The master may place an AFH capable slave in any of the three AFH operating modes.

Operation at the Master

A master that has one or more slaves in park, hold or sniff and decides to update them simultaneously shall schedule an *AFH_Instant* for a time that allows it to update all these slaves (as well as its active slaves) with the new adaptation.

A master that has multiple slaves with non-overlapping “wake” times (e.g. slaves in sniff mode with different timing parameters) may keep them *enabled* on the same adaptation provided that its scheduling of the *AFH_Instant* allows enough time to update them all.

This timing is summarized in the figure below. In this example the master decides that a hop sequence adaptation is required. However it cannot schedule an *AFH_Instant* until it has informed its active slave, its slave in hold, its slave in sniff, and had time to un-park its parked slaves.

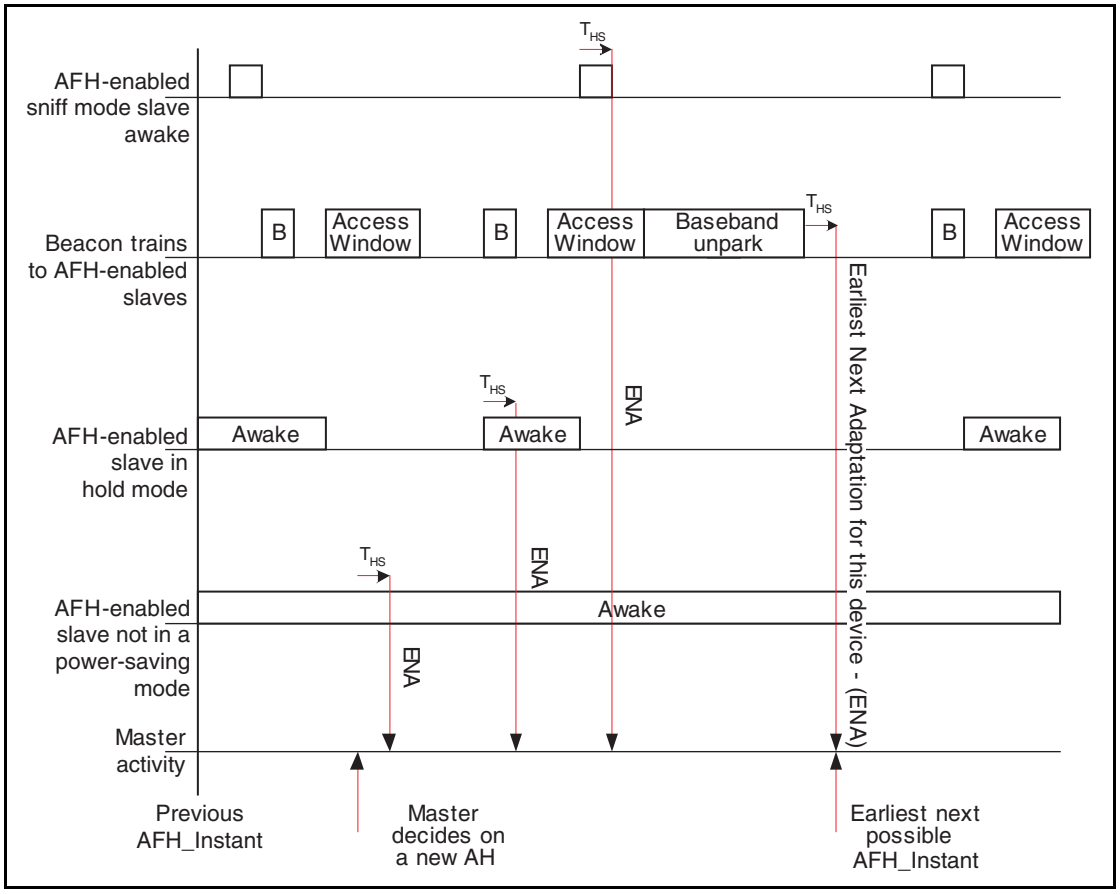


Figure 12.4: Timing constraint on AFH_Instant with slaves in park, hold and sniff

Operation in park

A slave that is in the Park state cannot send or receive any AFH LMP messages. Once the slave has left the Park state the master may subsequently update the slave's adaptation.

AFH Operation in Sniff

Once a slave has been placed in sniff mode, the master may periodically change its AHS without taking the slave out of sniff mode.

AFH Operation in Hold

A slave that is in hold mode cannot send or receive any LMP messages. Once the slave has left hold mode the master may subsequently update the slave's adaptation.

Core System Package [Controller volume]
Part C

LINK MANAGER PROTOCOL

This specification describes the Link Manager Protocol (LMP) which is used for link set-up and control. The signals are interpreted and filtered out by the Link Manager on the receiving side and are not propagated to higher layers.

CONTENTS

1 INTRODUCTION

The Link Manager Protocol (LMP) is used to control and negotiate all aspects of the operation of the Bluetooth connection between two devices. This includes the set-up and control of logical transports and logical links, and for control of physical links.

The Link Manager Protocol is used to communicate between the Link Managers (LM) on the two devices which are connected by the ACL logical transport. All LMP messages shall apply solely to the physical link and associated logical links and logical transports between the sending and receiving devices.

The protocol is made up of a series of messages which shall be transferred over the ACL-C logical link on the default ACL logical transport between two devices. LMP messages shall be interpreted and acted-upon by the LM and shall not be directly propagated to higher protocol layers.

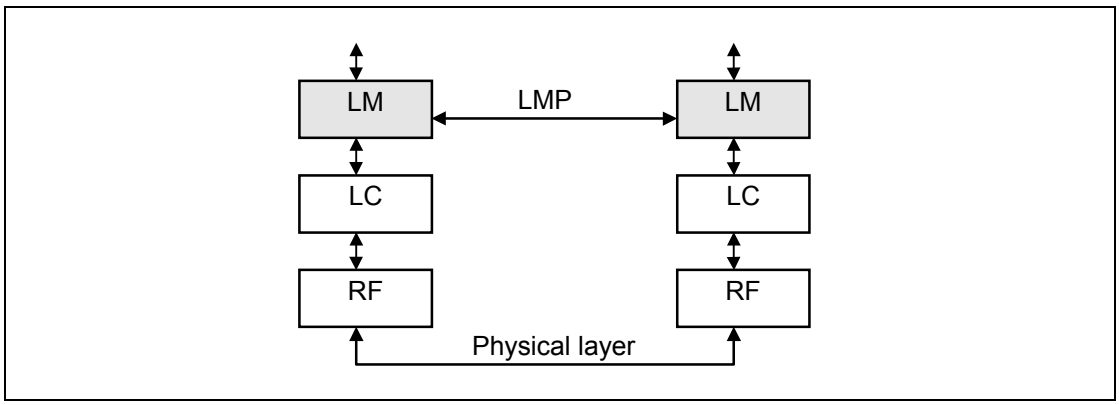


Figure 1.1: Link Manager Protocol signalling layer.

2 GENERAL RULES

2.1 MESSAGE TRANSPORT

LMP messages shall be exchanged over the ACL-C logical link that is carried on the default ACL logical transport (Baseband Specification, Section 4.4, on page 98). The ACL-C logical link is distinguished from the ACL-U (which carries L2CAP and user data) by the Logical Link Identifier (LLID) field carried in the payload header of variable-length packets (Baseband Specification, Section 6.6.2, on page 130).

The ACL-C has a higher priority than other traffic - see Baseband Specification, Section 5.6, on page 108.

The error detection and correction capabilities of the baseband ACL logical transport are generally sufficient for the requirements of the LMP. Therefore LMP messages do not contain any additional error detection information beyond what can be realized by means of sanity checks performed on the contents of LMP messages. Any such checks and protections to overcome undetected errors in LMP messages is an implementation matter.

2.2 SYNCHRONIZATION

This section is informative and explains why many of the LMP message sequences are defined as they are.

LMP messages are carried on the ACL-C logical link, which does not guarantee a time to deliver or acknowledge packets. LMP procedures take account of this when synchronizing state changes in the two devices. For example, criteria are defined that specify when a logical transport address (LT_ADDR) may be re-used after it becomes available due to a device leaving the piconet or entering the park state. Other LMP procedures, such as hold or role switch include the Bluetooth clock as a parameter in order to define a fixed synchronization point. The transitions into and out of sniff mode are protected with a transition mode.

The LC normally attempts to communicate with each slave no less often than every T_{poll} slots (see Section 4.1.8 on page 244) based on the T_{poll} for that slave.

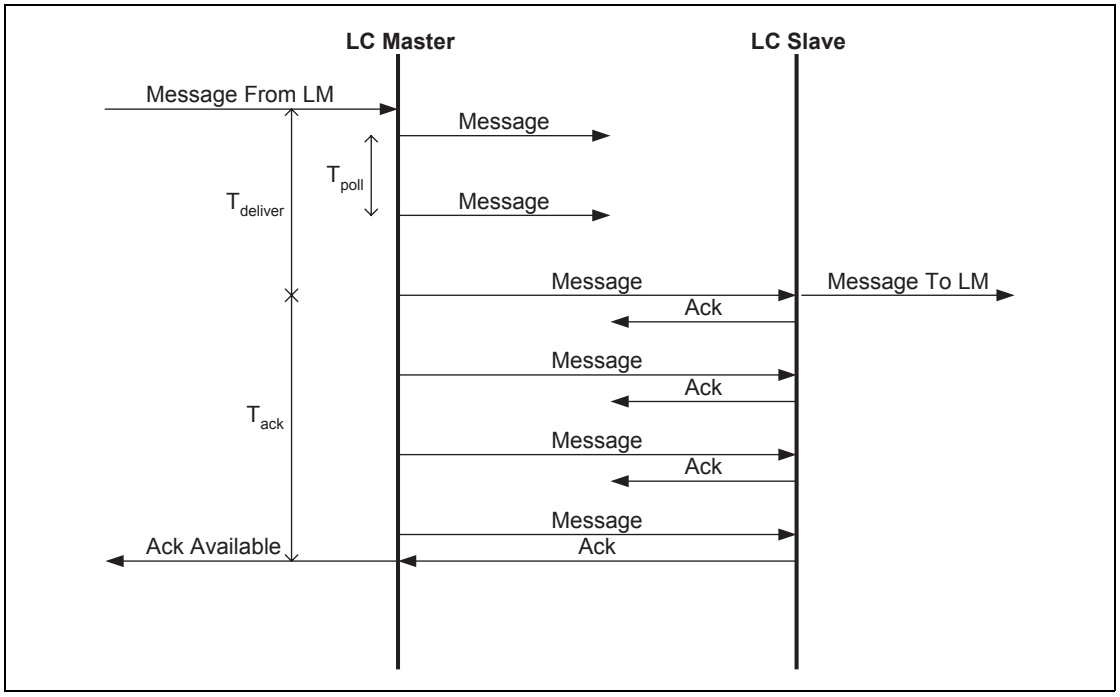


Figure 2.1: Transmission of a message from master to slave[1].

Figure 2.1 on page 220 illustrates the fundamental problem. It shows the transmission of a packet from the master to the slave in conditions of heavy interference for illustrative purposes. It is obvious that neither side can determine the value of either $T_{deliver}$ or T_{ack}. It is therefore not possible to use simple messages to identify uniquely the instant at which a state change occurs in the other device.

2.3 PACKET FORMAT

Each PDU is assigned either a 7 or a 15 bit opcode used to uniquely identify different types of PDUs, see Table 5.1 on page 295. The first 7 bits of the opcode and a transaction ID are located in the first byte of the payload body. If the initial 7 bits of the opcode have one of the special escape values 124-127 then an additional byte of opcode is located in the second byte of the payload and the combination uniquely identifies the PDU.

The FLOW bit in the packet header is always 1 and is ignored on reception.

If the PDU contains one or more parameters these are placed in the payload starting immediately after the opcode, i.e. at byte 2 if the PDU has a 7 bit opcode or byte 3 if the PDU has a 15 bit opcode. The number of bytes used

1. Note the diagram shows the limiting case where the master transmits the message at intervals of T_{poll}. In the case of heavy interference improved performance is gained by transmitting more often.

depends on the length of the parameters. All parameters have a little-endian format, i.e. the least significant byte is transferred first.

LMP messages shall be transmitted using DM1 packets, however if an HV1 SCO link is in use and the length of the payload is less than 9 bytes then DV packets may be used.

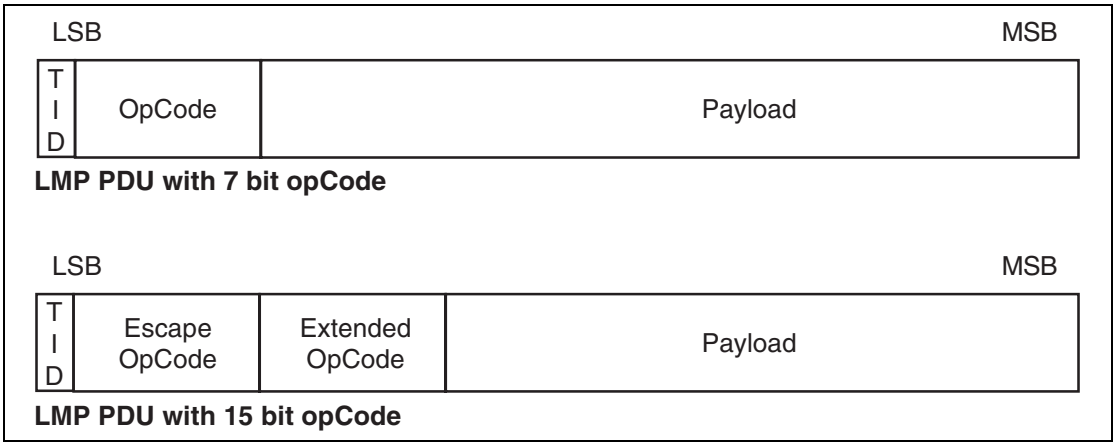


Figure 2.2: Payload body when LMP PDUs are sent.

2.4 TRANSACTIONS

The LMP operates in terms of transactions. A transaction is a connected set of message exchanges which achieve a particular purpose. All PDUs which form part of the same transaction shall have the same value for the transaction ID which is stored as part of the first byte of the opCode (see Section 2.3 on page 220).

The transaction ID is in the least significant bit. It shall be 0 if the PDU forms part of a transaction that was initiated by the master and 1 if the transaction was initiated by the slave.

Each sequence described in Section 4 on page 233 shall be defined as a transaction. For pairing, see Section 4.2.2 on page 251, and encryption, see Section 4.2.5 on page 257, all sequences belonging to each section shall be counted as one transaction and shall use the same transaction ID. For connection establishment, see Section 4.1.1 on page 233, LMP_host_connection_req and the response with LMP_accepted or LMP_not_accepted shall form one transaction and have the transaction ID of 0. LMP_setup_complete is a stand-alone PDU, which forms a transaction by itself.

For error handling, see Section 2.5 on page 222, the PDU to be rejected and LMP_not_accepted or LMP_not_accepted_ext shall form a single transaction.

2.4.1 LMP Response Timeout

The time between receiving a baseband packet carrying an LMP PDU and sending a baseband packet carrying a valid response PDU, according to the procedure rules in Section 4 on page 233, shall be less than the LMP Response Timeout. The value of this timeout is 30 seconds. Note that the LMP Response Timeout is applied not only to sequences described in Section 4 on page 233, but also to the series of the sequences defined as the transactions in Section 4 on page 233. It shall also be applied to the series of LMP transactions that take place during a period when traffic on the ACL-U logical link is disabled for the duration of the series of LMP transactions, for example during the enabling of encryption.

The LMP Response Timeout shall restart each time an LMP PDU which requires a reply is queued for transmission by the baseband.

2.5 ERROR HANDLING

If the LM receives a PDU with unrecognized opcode, it shall respond with LMP_not_accepted or LMP_not_accepted_ext with the error code *unknown LMP PDU.* The opcode parameter that is echoed back is the unrecognized opcode.

If the LM receives a PDU with invalid parameters, it shall respond with LMP_not_accepted or LMP_not_accepted_ext with the error code *invalid LMP parameters*.

If the maximum response time, see Section 2.4 on page 221, is exceeded or if a link loss is detected (see Baseband Specification, Section 3.1, on page 95), the party that waits for the response shall conclude that the procedure has terminated unsuccessfully.

Erroneous LMP messages can be caused by errors on the channel or systematic errors at the transmit side. To detect the latter case, the LM should monitor the number of erroneous messages and disconnect if it exceeds a threshold, which is implementation-dependent.

When the LM receives a PDU that is not allowed, and the PDU normally expects a PDU reply, for example LMP_host_connection_req or LMP_unit_key, the LM shall return PDU LMP_not_accepted or LMP_not_accepted_ext with the error code *PDU not allowed*. If the PDU normally doesn't expect a reply, for example LMP_sres or LMP_temp_key, the PDU will be ignored.

The reception of an optional PDU which is not supported shall be handled in one of two ways: if the LM simply does not know the opcode (e.g. it was added at a later version of the specification) it shall respond with LMP_not_accepted or LMP_not_accepted_ext with the error code *unknown LMP PDU*. If the LM recognizes the PDU as optional but unsupported then it shall reply with LMP_not_accepted or LMP_not_accepted_ext with the error code *unsupported LMP feature* if the PDU would normally generate a reply otherwise no reply is generated.

2.5.1 Transaction collision resolution

Since LMP PDUs are not interpreted in real time, collision situations can occur where both LMs initiate the same procedure and both cannot be completed. In this situation, the master shall reject the slave-initiated procedure by sending LMP_not_accepted or LMP_not_accepted_ext with the error code *LMP error transaction collision*. The master-initiated procedure shall then be completed.

Collision situations can also occur where both LMs initiate different procedures and both cannot be completed. In this situation, the master shall reject the slave-initiated procedure by sending LMP_not_accepted or LMP_not_accepted_ext with the error code *LMP error different transaction collision*. The master- initiated procedure shall then be completed.

2.6 PROCEDURE RULES

Each procedure is described and depicted with a sequence diagram. The following symbols are used in the sequence diagrams:

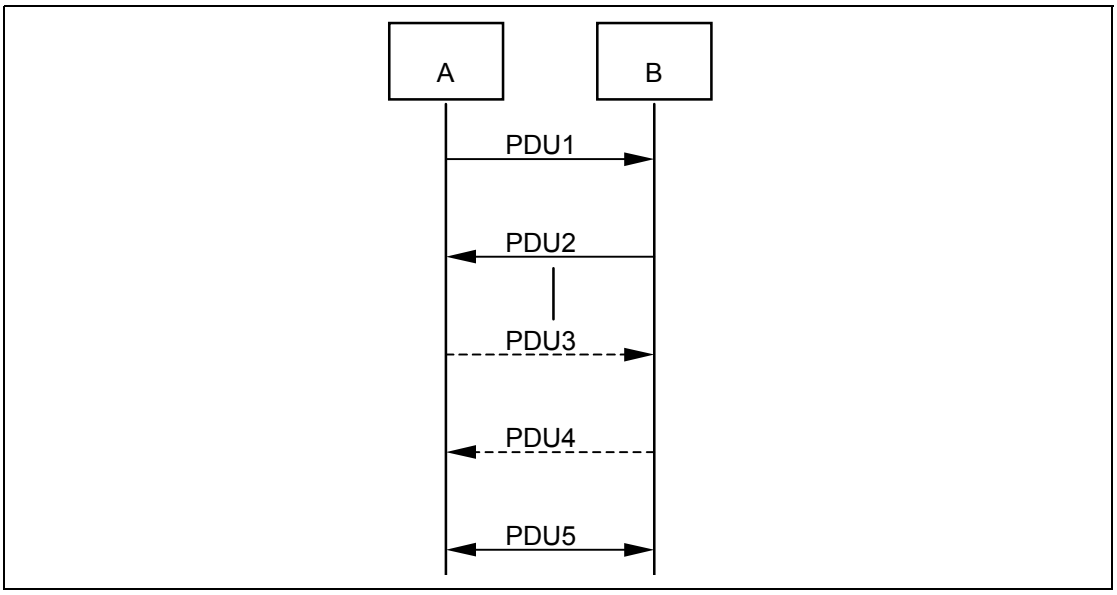


Figure 2.3: Symbols used in sequence diagrams.

PDU1 is a PDU sent from A to B. PDU2 is a PDU sent from B to A. PDU3 is a PDU that is optionally sent from A to B. PDU4 is a PDU that is optionally sent from B to A. PDU5 is a PDU sent from either A or B. A vertical line indicates that more PDUs can optionally be sent.

2.7 GENERAL RESPONSE MESSAGES

The PDUs LMP_accepted, LMP_accepted_ext, LMP_not_accepted and LMP_not_accepted_ext are used as response messages to other PDUs in a number of different procedures. LMP_accepted or LMP_accepted_ext includes the opcode of the message which is accepted. LMP_not_accepted or LMP_not_accepted_ext includes the opcode of the message which is not accepted and the error code why it is not accepted.

LMP_accepted_ext and LMP_not_accepted_ext shall be used when the opcode is 15 bits in length. LMP_accepted and LMP_not_accepted shall be used when the opcode is 7 bits in length.

M/O	PDU	Contents
M	LMP_accepted	op code
M	LMP_not_accepted	op code error code
O	LMP_accepted_ext	escape op code extended op code
O	LMP_not_accepted_ext	escape op code extended op code error code

Table 2.1: General response messages.

2.8 LMP MESSAGE CONSTRAINTS

This section is informative.

- No LMP message shall exceed the maximum payload length of a single DM1 packet i.e. 17 bytes in length (Baseband Specification, Section 6.5.4.1, on page 126).
- All LM messages are of fixed length apart from those sent using broadcast in park state.
- The LMP version number shall not be used to indicate the presence or absence of functionality.

3 DEVICE FEATURES

3.1 GENERAL DESCRIPTION

Each PDU is either Mandatory or Optional as defined by the M/O field in the tables of Section 4 on page 233. An M in this field shall indicate that support for the feature is mandatory. An O in this field shall indicate that support for the PDU is optional and it shall be followed by the number(s) of the feature(s) involved in brackets.

All features added after the 1.1 specification have associated LMP feature bits. Support of these features may be made “mandatory” by the qualification process but the LM still considers them to be optional since it must interoperate with older devices which do not support them.

The LM does not need to be able to transmit a PDU which is optional. Support of optional PDUs is indicated by a device's features mask. The features mask can be read (see Section 4.3.4 on page 264). An LM shall not send or be sent any PDU which is incompatible with the features signalled in its or its peer's features mask, as detailed in Section 3.2 .

3.2 FEATURE DEFINITIONS

Feature	Definition
Extended features	This feature indicates whether the device is able to support the extended features mask using the LMP sequences defined in Section 4.3.4 on page 264.
Timing accuracy	This feature indicates whether the LM supports requests for timing accuracy using the sequence defined in Section 4.3.1 on page 262.
Enhanced inquiry scan	This feature indicates whether the device is capable of supporting the enhanced inquiry scan mechanism as defined in Baseband Specification, Section 8.4.1, on page 164. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.
Interlaced inquiry scan	This feature indicates whether the device is capable of supporting the interlaced inquiry scan mechanism as defined in Baseband Specification, Section 8.4.1, on page 164. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.
Interlaced page scan	This feature indicates whether the device is capable of supporting the interlaced page scan mechanism as defined in Baseband Specification, Section 8.3.1, on page 154. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.

Table 3.1: Feature definitions.

Feature	Definition
RSSI with inquiry results	This feature indicates whether the device is capable of reporting the RSSI with inquiry results as defined in Baseband Specification, Section 8.4.2, on page 165. The presence of this feature has only local meaning and does not imply support for any additional LMP PDUs or sequences.
Paging parameter negotiation	This feature indicates whether the LM is capable of supporting the signaling of changes in the paging scheme as defined in Section 4.1.9 on page 246.
3 slot packets	This feature indicates whether the device supports the transmission and reception of both DM3 and DH3 packets for the transport of traffic on the ACL-U logical link.
5 slot packets	This feature indicates whether the device supports the transmission and reception of both DM5 and DH5 packets for the transport of traffic on the ACL-U logical link.
Flow control lag	This is defined as the total amount of ACL-U data that can be sent following the receipt of a valid payload header with the payload header flow bit set to 0 and is in units of 256 bytes. See further in Baseband Specification, Section 6.6.2, on page 130.
AFH capable slave	This feature indicates whether the device is able to support the Adaptive Frequency Hopping mechanism as a slave as defined in Baseband Specification, Section 2.3, on page 75 using the LMP sequences defined in Section 4.1.4 on page 237.
AFH classification slave	This feature indicates whether the device is able to support the AFH classification mechanism as a slave as defined in Baseband Specification, Section 8.6.8, on page 181 using the LMP sequences defined Section 4.1.5 on page 240.
AFH capable master	This indicates whether the device is able to support the Adaptive Frequency Hopping mechanism as a master as defined in Baseband Specification, Section 2.3, on page 75 using the LMP sequences defined in Section 4.1.4 on page 237.
AFH classification master	This feature indicate whether the device is able to support the AFH classification mechanism as a master as defined in Baseband Specification, Section 8.6.8, on page 181 using the LMP sequences defined Section 4.1.5 on page 240.
Power control	This feature indicates whether the device is capable of adjusting its transmission power. This feature indicates the ability to receive the LMP_incr_power and LMP_decr_power PDUs and transmit the LMP_max_power and LMP_min_power PDUs, using the sequences defined in Section 4.1.3 on page 235. These sequences may be used even if the remote device does not support the power control feature, as long as it supports the Power control requests feature.

Table 3.1: Feature definitions.

Feature	Definition
Power control requests	This feature indicates whether the device is capable of determining if the transmit power level of the other device should be adjusted and will send the LMP_incr_power and LMP_decr_power PDUs to request the adjustments. It indicates the ability to receive the LMP_max_power and LMP_min_power PDUs, using the sequences in Section 4.1.3 on page 235. These sequences may be used even if the remote device does not support the RSSI feature, as long as it supports the power control feature.
Channel Quality Driven Data Rate	This feature indicates whether the LM is capable of recommending a packet type (or types) depending on the channel quality using the LMP sequences defined in section Section 4.1.7 on page 243.
Broadcast encryption	This feature indicates whether the device is capable of supporting broadcast encryption as defined in [Part H] Section 4.2 on page 788 and also the sequences defined in Section 4.2.6 on page 261 and Section 4.2.4 on page 255. Note: Devices compliant to versions of this specification 1.1 and earlier may support broadcast encryption even though this feature bit is not set.
Encryption	This feature indicates whether the device supports the encryption of packet contents using the sequence defined in Section 4.2.5 on page 257.
Slot offset	This feature indicates whether the LM supports the transfer of the slot offset using the sequence defined in Section 4.4.1 on page 267.
Role switch	This feature indicates whether the device supports the change of master and slave roles as defined by baseband Section 8.6.5 on page 175 using the sequence defined in Section 4.4.2 on page 268.
Hold mode	This feature indicates whether the device is able to support Hold mode as defined in baseband Section 8.8 on page 185 using the LMP sequences defined in Section 4.5.1 on page 270.
Sniff mode	This feature indicates whether the device is able to support sniff mode as defined in baseband Section 8.7 on page 183 using the LMP sequences defined in Section 4.5.3 on page 278.
Park state	This feature indicates whether the device is able to support Park state as defined in baseband Section 8.9 on page 185 using the LMP sequences defined in Section 4.5.2 on page 272.
SCO link	This feature shall indicate whether the device is able to support the SCO logical transport as defined in Baseband Specification, Section 4.3, on page 98, the HV1 packet defined in Baseband Specification, Section 6.5.2.1, on page 123 and receiving the DV packet defined in Baseband Specification, Section 6.5.2.4, on page 123 using the LMP sequence in Section 4.6.1 on page 281.
HV2 packets	This feature indicates whether the device is capable of supporting the HV2 packet type as defined in baseband Section 6.5.2.2 on page 123 on the SCO logical transport.

Table 3.1: Feature definitions.

Feature	Definition
HV3 packets	This feature indicates whether the device is capable of supporting the HV3 packet type as defined in baseband Section 6.5.2.3 on page 123 on the SCO logical transport.
μ-law log synchronous data	This feature indicates whether the device is capable of supporting μ-Law Log format data as defined in Baseband Specification, Section 9.1, on page 195 on the SCO and eSCO logical transports.
A-law log synchronous data	This feature indicates whether the device is capable of supporting A-Law Log format data as defined in Baseband Specification, Section 9.1, on page 195 on the SCO and eSCO logical transports.
CVSD synchronous data	This feature indicates whether the device is capable of supporting CVSD format data as defined in Baseband Specification, Section 9.2, on page 195 on the SCO and eSCO logical transports.
Transparent synchronous data	This feature indicates whether the device is capable of supporting transparent synchronous data as defined in Baseband Specification, Section 6.4.3, on page 117 on the SCO and eSCO logical transports.
Extended SCO link	This feature indicates whether the device is able to support the eSCO logical transport as defined Baseband Specification, Section 5.5, on page 108 and the EV3 packet defined in Baseband Specification, Section 6.5.3.1, on page 124 using the LMP sequences defined in Section 4.6.2 on page 284.
EV4 packets	This feature indicates whether the device is capable of supporting the EV4 packet type defined in Baseband Specification, Section 6.5.3.2, on page 124 on the eSCO logical transport.
EV5 packets	This feature indicates whether the device is capable of supporting the EV5 packet type defined in Baseband Specification, Section 6.5.3.3, on page 124 on the eSCO logical transport.
Enhanced Data Rate ACL 2 Mbps mode	This feature indicates whether the device supports the transmission and reception of 2-DH1 packets for the transport of traffic on the ACL-U logical link.
Enhanced Data Rate ACL 3 Mbps mode	This feature indicates whether the device supports the transmission and reception of 3-DH1 packets for the transport of traffic on the ACL-U logical link. This feature bit shall only be set if the “Enhanced Data Rate ACL 2 Mbps mode” feature bit is set.
3-slot Enhanced Data Rate ACL packets	This feature indicates whether the device supports the transmission and reception of three-slot Enhanced Data Rate packets on the ACL-U logical link. This feature bit shall only be set if the “Enhanced Data Rate ACL 2 Mbps mode” feature bit is set.
5-slot Enhanced Data Rate ACL packets	This feature indicates whether the device supports the transmission and reception of 5-slot Enhanced Data Rate packets for the transport of traffic on the ACL-U logical link. This feature bit shall only be set if the “Enhanced Data Rate ACL 2 Mbps mode” feature bit is set.

Table 3.1: Feature definitions.

Feature	Definition
Enhanced Data Rate eSCO 2 Mbps mode	This feature indicates whether the device supports the transmission and reception of 2-EV3 packets for the transport of traffic on the eSCO logical transport.
Enhanced Data Rate eSCO 3 Mbps mode	This feature indicates whether the device supports the transmission and reception of 3-EV3 packets for the transport of traffic on the eSCO logical transport. This feature bit shall only be set if the “Enhanced Data Rate eSCO 2 Mbps mode” feature bit is set.
3-slot Enhanced Data Rate eSCO packets	This feature indicates whether the device supports the transmission and reception of 3-slot Enhanced Data Rate packets for the transport of traffic on the eSCO logical transport. This feature bit shall only be set if the “Enhanced Data Rate eSCO 2 Mbps mode” feature bit is set.

Table 3.1: Feature definitions.

3.3 FEATURE MASK DEFINITION

The features are represented as a bit mask when they are transferred in LMP messages. For each feature a single bit is specified which shall be set to 1 if the feature is supported and set to 0 otherwise. The single exception is the flow control lag which is coded as a 3 bit field with the least significant bit in byte 2 bit 4 and the most significant bit in byte 2 bit 6. All unknown or unassigned feature bits shall be set to 0.

No.	Supported feature	Byte	Bit
0	3 slot packets	0	0
1	5 slot packets	0	1
2	Encryption	0	2
3	Slot offset	0	3
4	Timing accuracy	0	4
5	Role switch	0	5
6	Hold mode	0	6
7	Sniff mode	0	7
8	Park state	1	0
9	Power control requests	1	1
10	Channel quality driven data rate (CQDDR)	1	2
11	SCO link	1	3
12	HV2 packets	1	4
13	HV3 packets	1	5
14	μ-law log synchronous data	1	6
15	A-law log synchronous data	1	7
16	CVSD synchronous data	2	0
17	Paging parameter negotiation	2	1
18	Power control	2	2
19	Transparent synchronous data	2	3
20	Flow control lag (least significant bit)	2	4
21	Flow control lag (middle bit)	2	5
22	Flow control lag (most significant bit)	2	6
23	Broadcast encryption	2	7
24	Reserved	3	0
25	Enhanced Data Rate ACL 2 Mbps mode	3	1

Table 3.2: Feature mask definition

No.	Supported feature	Byte	Bit
26	Enhanced Data Rate ACL 3Mbps mode	3	2
27	Enhanced inquiry scan	3	3
28	Interlaced inquiry scan	3	4
29	Interlaced page scan	3	5
30	RSSI with inquiry results	3	6
31	Extended SCO link (EV3 packets)	3	7
32	EV4 packets	4	0
33	EV5 packets	4	1
34	Reserved	4	2
35	AFH capable slave	4	3
36	AFH classification slave	4	4
37	Reserved	4	5
38	Reserved	4	6
39	3-slot Enhanced Data Rate ACL packets	4	7
40	5-slot Enhanced Data Rate ACL packets	5	6
41	reserved	5	1
42	reserved	5	2
43	AFH capable master	5	3
44	AFH classification master	5	4
45	Enhanced Data Rate eSCO 2 Mbps mode	5	5
46	Enhanced Data Rate eSCO 3 Mbps mode	5	6
47	3-slot Enhanced Data Rate eSCO packets	5	7
48	Reserved	6	0
63	Extended features	7	7

Table 3.2: Feature mask definition

3.4 LINK MANAGER INTEROPERABILITY POLICY

Link managers of any version will interoperate using the lowest common subset of functionality by reading the LMP features mask (defined in Table 3.2 on page 230).

An optional LMP PDU shall only be sent to a device if the corresponding feature bit is set in its feature mask. The exception to this are certain PDUs (see Section 4.1.1 on page 233) which can be sent before the features mask is requested. Note: a later version device with a restricted feature set is indistinguishable from an earlier version device with the same features.

4 PROCEDURE RULES

4.1 CONNECTION CONTROL

4.1.1 Connection establishment

After the paging procedure, LMP procedures with for clock offset request, LMP version, supported features, name request and detach may then be initiated.

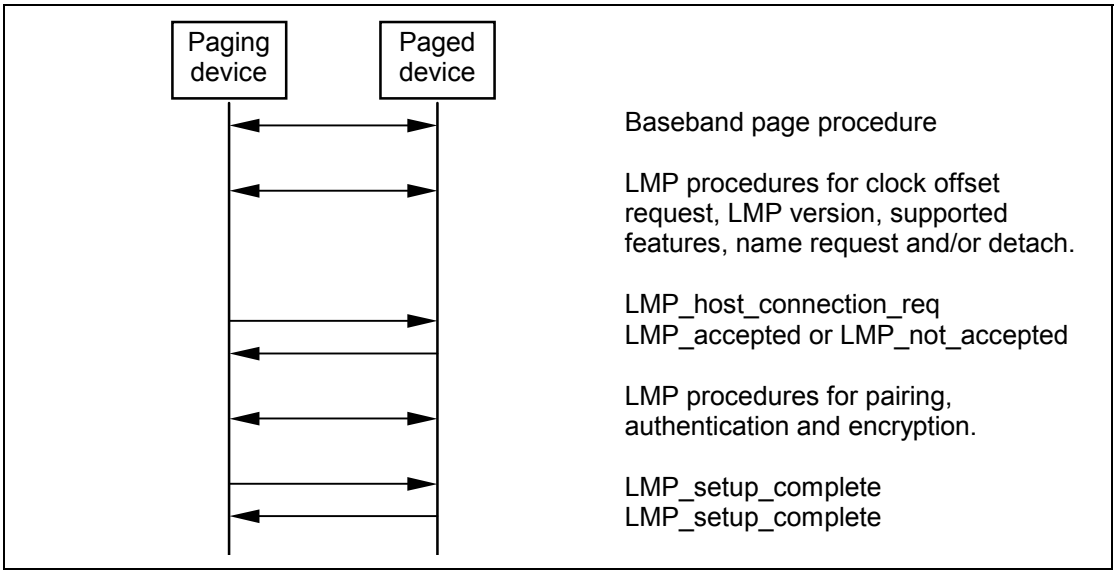


Figure 4.1: Connection establishment.

When the paging device wishes to create a connection involving layers above LM, it sends an LMP_host_connection_req PDU. When the other side receives this message, the host is informed about the incoming connection. The remote device can accept or reject the connection request by sending an LMP_accepted PDU or an LMP_not_accepted PDU. Alternatively, if the slave needs a role switch, see Section 4.4.2 on page 268, it sends an LMP_slot_offset PDU and LMP_switch_req PDU after it has received an LMP_host_connection_req PDU. If the role switch fails the LM shall continue with the creation of the connection unless this cannot be supported due to limited resources in which case the connection shall be terminated with an LMP_detach PDU with error code *other end terminated connection: low resources*. When the role switch has been successfully completed, the old slave will reply with an LMP_accepted PDU or an LMP_not_accepted PDU to the LMP_host_connection_req PDU (with the transaction ID set to 0).

If the paging device receives an LMP_not_accepted PDU in response to LMP_host_connection_req it shall immediately disconnect the link using the mechanism described in Section 4.1.2 on page 234.

If the LMP_host_connection_req PDU is accepted, LMP security procedures (pairing, authentication and encryption) may be invoked. When a device is not going to initiate any more security procedures during connection establishment it sends an LMP_setup_complete PDU. When both devices have sent LMP_setup_complete PDUs the traffic can be transferred on the ACL-U logical transport.

M/O	PDU	Contents
M	LMP_host_connection_req	-
M	LMP_setup_complete	-

Table 4.1: PDUs used for connection establishment.

4.1.2 Detach

The connection between two Bluetooth devices may be detached anytime by the master or the slave. An error code parameter is included in the message to inform the other party of why the connection is detached.

M/O	PDU	Contents
M	LMP_detach	error code

Table 4.2: PDU used for detach.

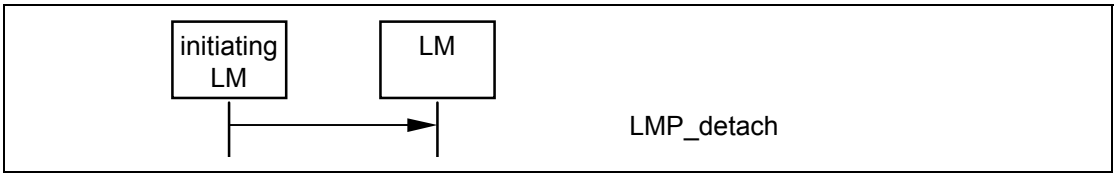
The initiating LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). The initiating LM then queues the LMP_detach for transmission and it shall start a timer for $6*T_{poll}$ slots where T_{poll} is the poll interval for the connection. If the initiating LM receives the baseband acknowledgement before the timer expires it starts a timer for $3*T_{poll}$ slots. When this timer expires, and if the initiating LM is the master, the LT_ADDR(s) may be re-used immediately. If the initial timer expires then the initiating LM drops the link and starts a timer for $T_{linksupervisiontimeout}$ slots after which the LT_ADDR(s) may be re-used if the initiating LM is the master.

When the receiving LM receives the LMP_detach, it shall start a timer for $6*T_{poll}$ slots if it is the master and $3*T_{poll}$ if it is the slave. On timer expiration, the link shall be detached and, if the receiving LM is the master, the LT_ADDR(s) may be re-used immediately. If the receiver never receives the LMP_detach then a link supervision timeout will occur, the link will be detached, and the LT_ADDR may be re-used immediately.

If at any time during this or any other LMP sequence the Link supervision time-out expires then the link shall be terminated immediately and the LT_ADDR(S) may be re-used immediately.

If the connection is in hold mode, the initiating LM shall wait for hold mode to end before initiating the procedure defined above. If the connection is in sniff mode or park state, the initiating LM shall perform the procedure to exit sniff

mode or park state before initiating the procedure defined above. If the procedure to exit sniff mode or park state does not complete within the LMP response timeout (30 seconds) the procedure defined above shall be initiated anyway.



Sequence 1: Connection closed by sending LMP_detach.

4.1.3 Power control

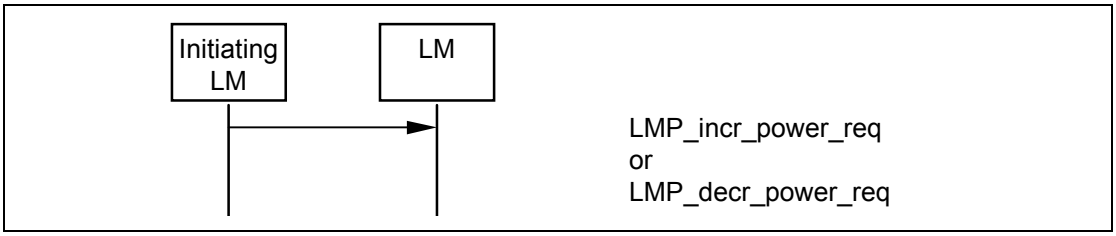
If the received signal characteristics differs too much from the preferred value of a Bluetooth device, it may request an increase or a decrease of the other device's TX power. The power adjustment requests may be made at anytime following a successful baseband paging procedure.

If a device does not support power control requests this is indicated in the supported features list and thus no power control requests shall be sent after the supported features response has been processed. Prior to this time, a power control adjustment might be sent and if the recipient does not support power control it is allowed to send LMP_max_power in response to LMP_incr_power_req and LMP_min_power in response to LMP_decr_power_req. Another possibility is to send LMP_not_accepted with the error code *unsupported LMP feature*.

Upon receipt of an LMP_incr_power_req PDU or LMP_decr_power_req PDU the output power shall be increased or decreased one step. See Radio Specification Section 3, on page 31 for the definition of the step size. The TX power is a property of the physical link, and affects all logical transports carried over the physical link. Power control requests carried over the default ACL-C logical link shall only affect the physical link associated with the default ACL-C logical link: they shall not affect the power level used on the physical links to other slaves.

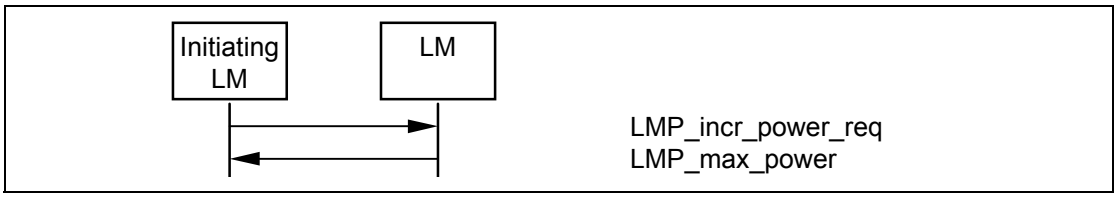
M/O	PDU	Contents
O(9)	LMP_incr_power_req	for future use (1 Byte)
O(9)	LMP_decr_power_req	for future use (1 Byte)
O(18)	LMP_max_power	-
O(18)	LMP_min_power	-

Table 4.3: PDUs used for power control.

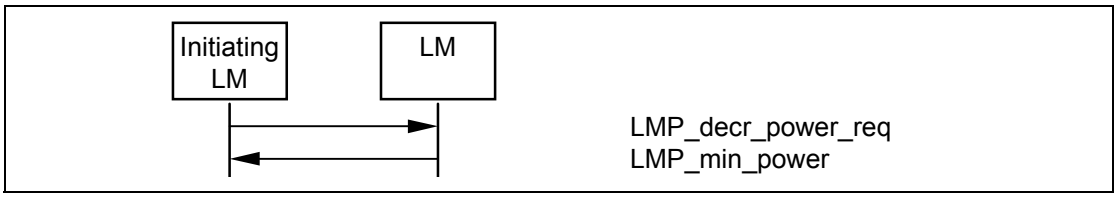


Sequence 2: A device requests a change of the other device's TX power.

If the receiver of LMP_incr_power_req is at maximum power LMP_max_power shall be returned. The device shall only request an increase again after having requested a decrease at least once. If the receiver of LMP_decr_power_req is at minimum power then LMP_min_power shall be returned and the device shall only request a decrease after having requested an increase at least once.



Sequence 3: The TX power cannot be increased.



Sequence 4: The TX power cannot be decreased.

One byte is reserved in LMP_incr/decr_power_req for future use. The parameter value shall be 0x00 and ignored upon receipt.

4.1.4 Adaptive frequency hopping

AFH is used to improve the performance of physical links in the presence of interference as well as reducing the interference caused by physical links on other devices in the ISM band. AFH shall only be used during the connection state.

M/O	PDU	Contents
O(35) Rx O(43) Tx	LMP_set_AFH	AFH_Instant, AFH_Mode, AFH_Channel_Map

Table 4.4: PDUs used for AFH

The LMP_set_AFH PDU contains three parameters: AFH_Instant, AFH_Mode, and AFH_Channel_Map. The parameter, AFH_Instant, specifies the instant at which the hopset switch will become effective. This is specified as a Bluetooth Clock value of the master's clock, that is available to both devices. The AFH instant is chosen by the master and shall be an even value at least $6*T_{poll}$ or 96 slots (whichever is greater) in the future, where T_{poll} is at least the longest poll interval for all AFH enabled physical links. The AFH_instant shall be within 12 hours of the current clock value. The parameter AFH_Mode, specifies whether AFH shall be enabled or disabled. The parameter AFH_Channel_Map, specifies the set of channels that shall be used if AFH is enabled.

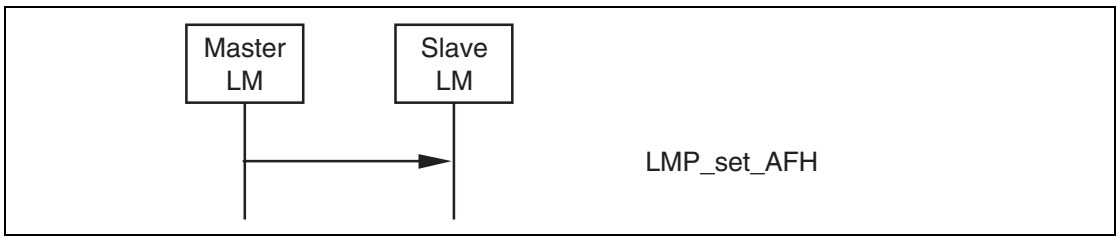
When the LMP_set_AFH PDU is received the AFH instant shall be compared with the current Bluetooth clock value. If it is in the past then the AFH_instant has passed and the slave shall immediately configure the hop selection kernel (see Baseband Specification, Section 2.6.3, on page 89) with the new AFH_mode and AFH_channel_map specified in the LMP_set_AFH PDU. If it is in the future then a timer shall be started to expire at the AFH instant. When this timer expires it shall configure the hop selection kernel with the new AFH_mode and AFH_channel_map.

The master shall not send a new LMP_set_AFH PDU to a slave until it has received the baseband acknowledgement for any previous LMP_set_AFH addressed to that slave and the instant has passed.

Role switch while AFH is enabled shall follow the procedures define by Baseband Specification, Section 8.6.5, on page 175.

4.1.4.1 Master enables AFH

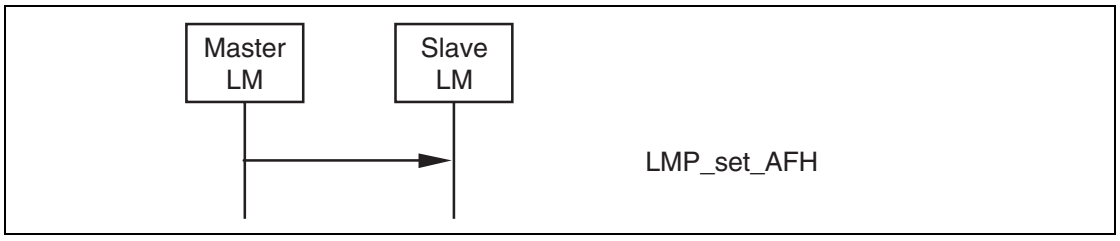
Prior to enabling AFH the master LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). The master shall then enable AFH on a physical link by sending the LMP_set_AFH PDU with AFH_mode set to AFH_enabled, the AFH_channel_map parameter containing the set of used and unused channels, and an AFH_instant. The LM shall not calculate the AFH instant until after traffic on the ACL-U logical link has been stopped. The master considers the physical link to be AFH_enabled once the baseband acknowledgement has been received and the AFH_instant has passed. Once the baseband acknowledgement has been received the master shall restart transmission on the ACL-U logical link.



Sequence 5: Master Enables AFH.

4.1.4.2 Master disables AFH

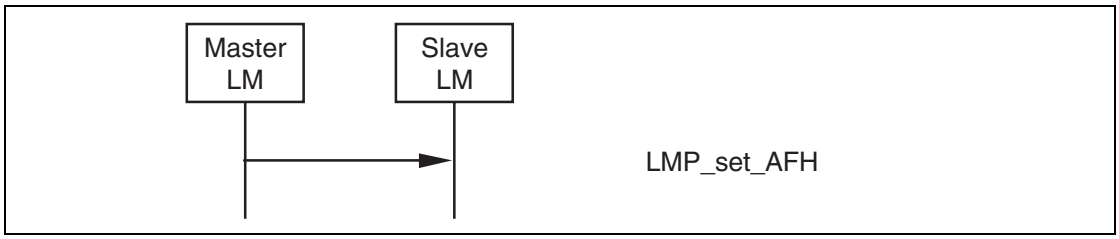
Prior to disabling AFH the master LM shall pause traffic on the ACL-U logical link (Baseband Specification, Section 5.3.1, on page 108). The master shall then disable AFH operation on a physical link by sending the LMP_set_AFH PDU with AFH_mode set to AFH_disabled and an AFH_instant. The AFH_channel_map parameter is not valid when AFH_mode is AFH_disabled. The LM shall not calculate the AFH instant until after traffic on the ACL-U logical link has been stopped. The master considers the physical link to have entered AFH_disabled operation once the baseband acknowledgement has been received and the AFH_instant has passed. Once the baseband acknowledgement has been received the master shall restart transmission on the ACL-U logical link.



Sequence 6: Master disables AFH.

4.1.4.3 Master updates AFH

A master shall update the AFH parameters on a physical link by sending the LMP_set_AFH PDU with AFH_mode set to AFH_enabled, an AFH_instant and a new AFH_channel_map. The master shall consider the slave to have the updated AFH parameters once the baseband acknowledgement has been received and the AFH_instant has passed.



Sequence 7: Master Updates AFH.

4.1.4.4 AFH operation in park, hold and sniff modes

A slave in Park, Hold or Sniff shall retain the AFH_mode and AFH_channel_map prior to entering those modes. A master may change the AFH_mode while a slave is in sniff.

A master that receives a request from an AFH_enabled slave to enter Park, Hold or Sniff and decides to operate the slave using a different hop sequence shall respond with an LMP_set_AFH PDU specifying the new hop sequence.

The master continues with the LMP signalling, for Park, Hold or Sniff initiation, once the baseband acknowledgement for the LMP_set_AFH PDU has been received. Optionally, the master may delay the continuation of this LMP signalling until after the instant. An AFH_capable_slave device shall support both of these cases.

A master that receives a request from an AFH_enabled slave to enter Park, Hold or Sniff and decides not to change the slave's hop sequence shall respond exactly as it would do without AFH. In this case, AFH operation has no effect on the LMP signalling.

4.1.5 Channel classification

A master may request channel classification information from a slave that is AFH_enabled.

A slave that supports the AFH_classification_slave feature shall perform channel classification and reporting according to its AFH_reporting_mode. The master shall control the AFH_reporting_mode using the LMP_channel_classification_req PDU. The slave shall report its channel classification using the LMP_channel_classification PDU.

The slave shall report pairs of channels as *good*, *bad* or *unknown*. See Table 5.2 on page 303 for the detailed format of the AFH_Channel_Classification parameter. When one channel in the n^{th} channel pair is good and the other channel is unknown the n^{th} channel pair shall be reported as good. When one channel in the n^{th} channel pair is bad and the other is unknown the n^{th} channel pair shall be reported as bad. It is implementation dependent what to report when one channel in a channel pair is good and the other is bad.

M/O	PDU	Contents
O(36) Rx O(44) Tx	LMP_channel_classification_req	AFH_Reporting_Mode, AFH_Min_Interval, AFH_Max_Interval
O(36) Tx O(44) Rx	LMP_channel_classification	AFH_Channel_Classification

Table 4.5: PDUs used for Channel Classification Reporting.

The LMP_channel_classification_req PDU contains three parameters: AFH_Reporting_Mode, AFH_Min_Interval, and AFH_Max_Interval. In the AFH_reporting_disabled state, the slave shall not generate any channel classification reports. The parameter AFH_min_interval, defines the minimum amount of time from the last LMP_channel_classification command that was sent before the next LMP_channel_classification PDU may be sent. The parameter AFH_max_interval, defines the maximum amount of time between the change in the radio environment being detected by a slave and its generation of an LMP_channel_classification PDU. The AFH_max_interval shall be equal to or larger than AFH_min_interval.

The AFH_reporting_mode parameter shall determine if the slave is in the AFH_reporting_enabled or AFH_reporting_disabled state. The default state, prior to receipt of any LMP_channel_classification_req PDUs, shall be AFH_reporting_disabled.

AFH_reporting_mode is implicitly set to the AFH_reporting_disabled state when any of the following occur:

- Establishment of a connection at the baseband level
- Master-slave role switch

- Entry to park state operation
- Entry to hold mode operation

AFH_reporting_mode is implicitly restored to its former value when any of the following occur:

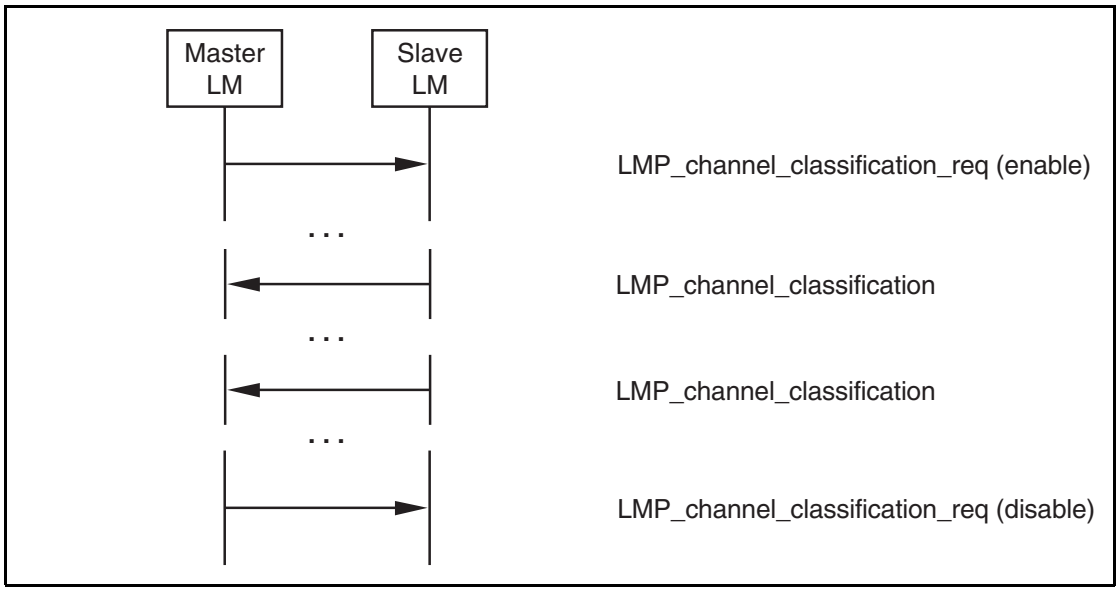
- Exit from park state operation
- Exit from hold mode
- Failure of Master-slave role switch

4.1.5.1 Channel classification reporting enabling and disabling

A master enables slave channel classification reporting by sending the LMP_channel_classification_req PDU with the AFH_reporting_mode parameter set to AFH_reporting_enabled.

When a slave has had classification reporting enabled by the master it shall send the LMP_channel_classification PDU according to the information in the latest LMP_channel_classification_req PDU. The LMP_channel_classification PDU shall not be sent if there has been no change in the slave's channel classification.

A master disables slave channel classification reporting by sending the LMP_channel_classification_req PDU with the AFH_reporting_mode parameter set to AFH_reporting_disabled.



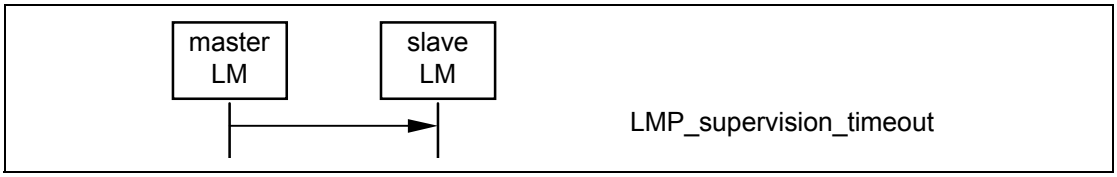
Sequence 8: Channel classification reporting.

4.1.6 Link supervision

Each physical link has a timer that is used for link supervision. This timer is used to detect physical link loss caused by devices moving out of range, or being blocked by interference, a device's power-down, or other similar failure cases. Link supervision is specified in Baseband Specification, Section 3.1, on page 95.

M/O	PDU	Contents
M	LMP_supervision_timeout	supervision timeout

Table 4.6: PDU used to set the supervision timeout.



Sequence 9: Setting the link supervision timeout.

4.1.7 Channel quality driven data rate change (CQDDR)

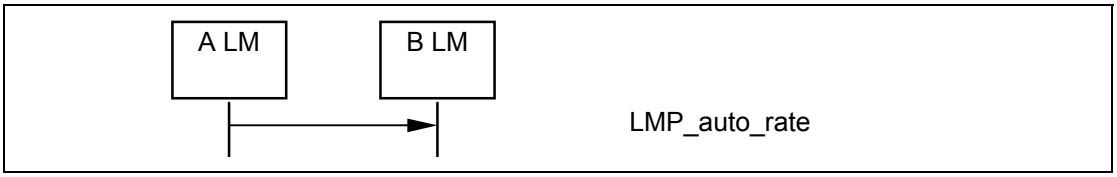
The data throughput for a given packet type depends on the quality of the RF channel. Quality measurements in the receiver of one device can be used to dynamically control the packet type transmitted from the remote device for optimization of the data throughput. Device A sends the LMP_auto_rate PDU once to notify device B to enable this feature. Once enabled, device B may change the packet type(s) that A transmits by sending the LMP_preferred_rate PDU. This PDU has a parameter which determines the preferred coding (with or without 2/3FEC) and optionally the preferred size in slots of the packets. Device A is not required to change to the packet type specified by this parameter. Device A shall not send a packet that is larger than max slots (see Section 4.1.10 on page 247) even if the preferred size is greater than this value.

The data rate parameter includes the preferred rate for Basic Rate and Enhanced Data Rate modes. When operating in Basic Rate mode, the device shall use bits 0-2 to determine the preferred data rate. When operating in Enhanced Data Rate mode, the device shall use bits 3-6 to determine the preferred data rate. For devices that support Enhanced Data Rate, the preferred rates for both Basic Rate and Enhanced Data Rate modes shall be valid at all times.

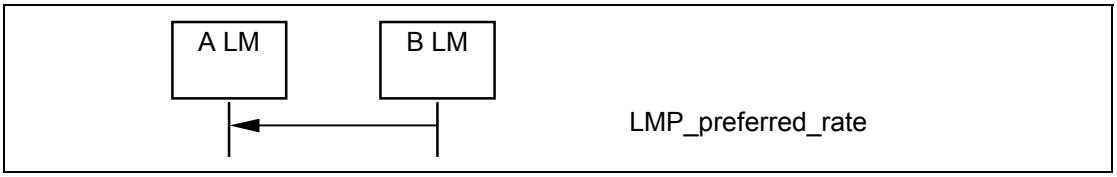
These PDUs may be sent at any time after connection setup is completed.

M/O	PDU	Contents
O(10)	LMP_auto_rate	-
O(10)	LMP_preferred_rate	data rate

Table 4.7: PDUs used for quality driven change of the data rate.



Sequence 10: A notifies B to enable CQDDR



Sequence 11: B sends A a preferred packet type

4.1.8 Quality of service (QoS)

The LM provides QoS capabilities. A poll interval, T_{poll}, that is defined as the maximum time between transmissions from the master to a particular slave on the ACL logical transport, is used to support bandwidth allocation and latency control - see Baseband Specification, Section 8.6.1, on page 169 for details. The poll interval is guaranteed in the active and sniff modes except when there are collisions with page, page scan, inquiry and inquiry scan, during time critical LMP sequences in the current piconet and any other piconets in which the Bluetooth device is a member, and during critical baseband sequences (such as the page response, initial connection state until the first POLL, and master slave switch). These PDUs maybe sent at anytime after connection setup is completed.

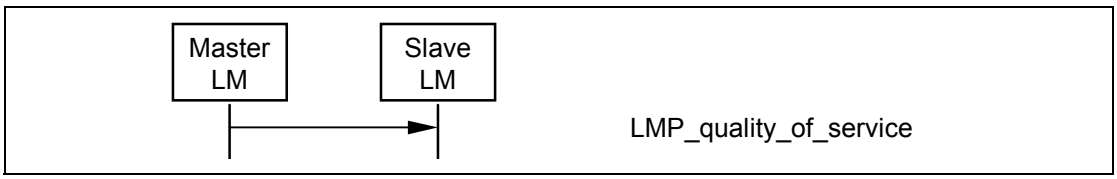
Master and slave negotiate the number of repetitions for broadcast packets (N_{BC}), see Baseband Specification, Section 7.6.5, on page 150.

M/O	PDU	Contents
M	LMP_quality_of_service	poll interval N_{BC}
M	LMP_quality_of_service_req	poll interval N_{BC}

Table 4.8: PDUs used for quality of service.

4.1.8.1 Master notifies slave of the quality of service

The master notifies the slave of the new poll interval and N_{BC} by sending the LMP_quality_of_service PDU. The slave cannot reject the notification.

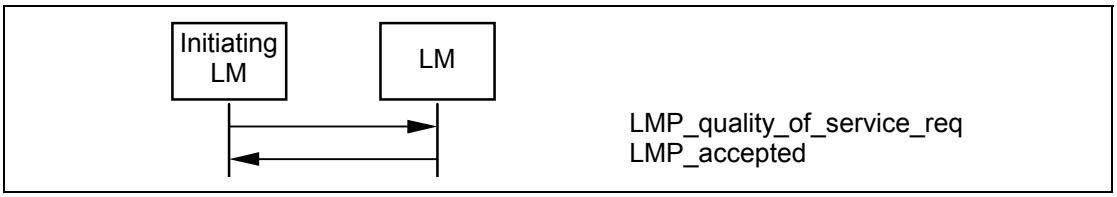


Sequence 12: Master notifies slave of quality of service.

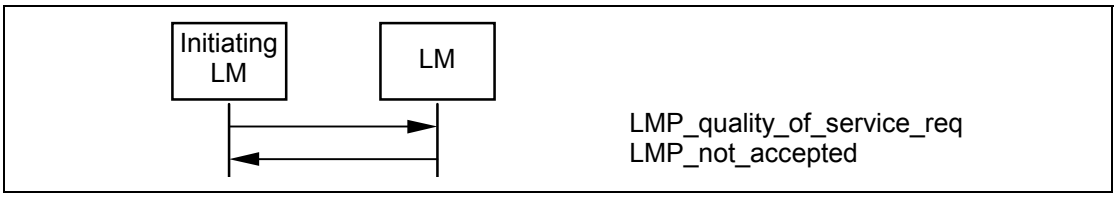
4.1.8.2 Device requests new quality of service

Either the master or the slave may request a new poll interval and N_{BC} by sending an LMP_quality_of_service_req PDU to the slave. The parameter N_{BC} is meaningful only when it is sent by a master to a slave. For transmission of LMP_quality_of_service_req PDUs from a slave, this parameter shall be ignored by the master. The request can be accepted or rejected. This allows the master and slave to dynamically negotiate the quality of service as needed.

The selected poll interval by the slave shall be less than or equal to the specified Access Latency for the outgoing traffic of the ACL link (see L2CAP "Quality of Service (QoS) Option" on page 60[vol. 4]).



Sequence 13: Device accepts new quality of service



Sequence 14: Device rejects new quality of service.

4.1.9 Paging scheme parameters

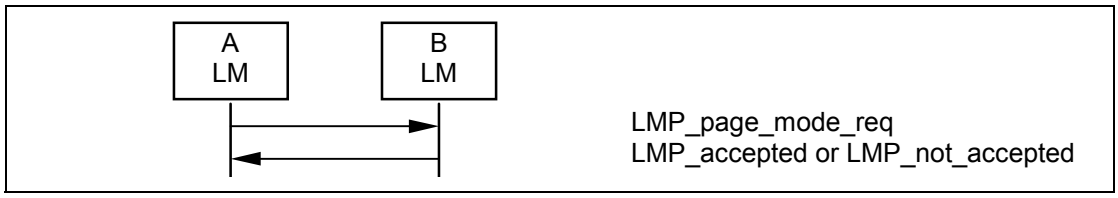
LMP provides a means to negotiate the paging scheme parameters that are used the next time a device is paged.

M/O	PDU	Contents
O(17)	LMP_page_mode_req	paging scheme paging scheme settings
O(17)	LMP_page_scan_mode_req	paging scheme paging scheme settings

Table 4.9: PDUs used to request paging scheme.

4.1.9.1 Page mode

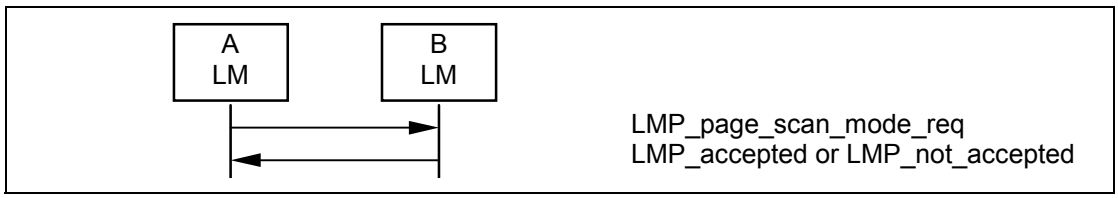
This procedure is initiated from device A and negotiates the paging scheme used when device A pages device B. Device A proposes a paging scheme including the parameters for this scheme and device B can accept or reject. On rejection the old setting will not be changed. A request to switch to a reserved paging scheme shall be rejected.



Sequence 15: Negotiation for page mode.

4.1.9.2 Page scan mode

This procedure is initiated from device A and negotiates the paging scheme and paging scheme settings used when device B pages device A. Device A proposes a paging scheme and paging scheme settings and device B may accept or reject. On reject the old setting is not changed. A request specifying the mandatory scheme shall be accepted. A request specifying a non-mandatory scheme shall be rejected. This procedure should be used when device A changes its paging scheme settings. A slave should also send this message to the master after connection establishment, to inform the master of the slave's current paging scheme and paging scheme settings.



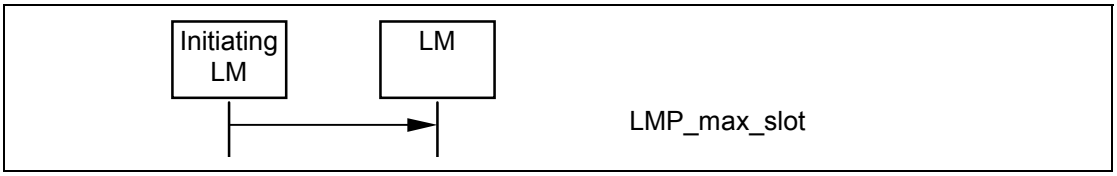
Sequence 16: Negotiation for page scan mode

4.1.10 Control of multi-slot packets

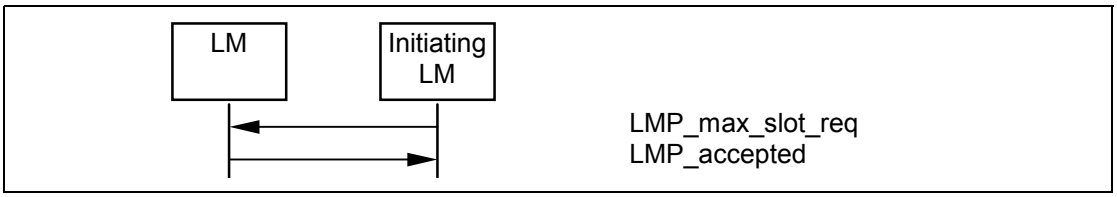
The number of consecutive slots used by a device on an ACL-U logical link can be limited. It does not affect traffic on the eSCO links where the packet sizes are defined as part of link setup. A device allows the remote device to use a maximum number of slots by sending the PDU LMP_max_slot providing max slots as parameter. Each device can request to use a maximal number of slots by sending the PDU LMP_max_slot_req providing max slots as parameter. After a new connection, as a result of page, page scan, role switch or unpark, the default value is 1 slot. These PDUs can be sent at anytime after connection setup is completed.

M/O	PDU	Contents
M	LMP_max_slot	max slots
M	LMP_max_slot_req	max slots

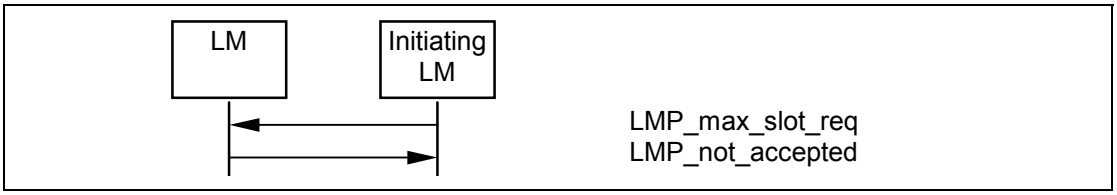
Table 4.10: PDUs used to control the use of multi-slot packets.



Sequence 17: Device allows Remote Device to use a maximum number of slots.



Sequence 18: Device requests a maximum number of slots. Remote Device accepts.



Sequence 19: Device requests a maximum number of slots. Remote Device rejects.

4.1.11 Enhanced Data Rate

A device may change the packet type table, ptt, to select which if any of the optional modulation schemes are to be used on an ACL logical transport.

Either the master or the slave may request a new packet type table and therefore the modulation scheme to be used on this ACL link. After a new baseband connection, as a result of page or page scan, the default value for ptt shall be 0.

The change of the modulation mode for an ACL logical transport shall not affect the packet types used for an associated SCO logical transport on the same LT_ADDR.

Note: Enhanced Data Rate eSCO links are negotiated using the LMP eSCO link_req as described in section 4.6.2.

Before changing the packet type table, the initiator shall finalize the transmission of the current ACL packet with ACL-U information and shall stop ACL-U transmissions. It shall then send the LMP_packet_type_table_req PDU.

If the receiver rejects the change, then it shall respond with an LMP_not_accepted_ext PDU.

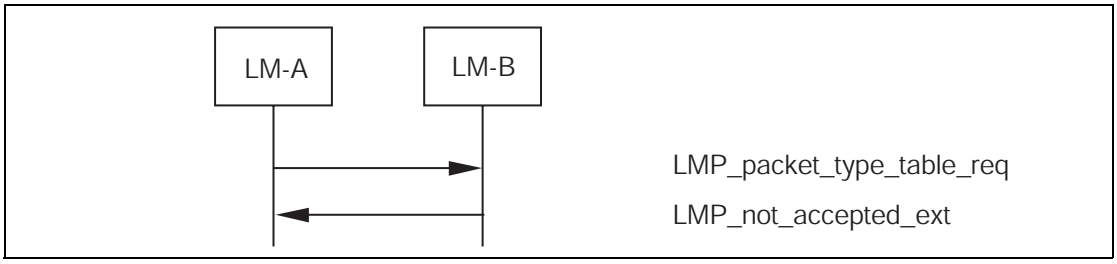
If the receiver accepts the change, then it shall finalize the transmission of the current ACL packet with ACL-U information and shall stop ACL-U transmissions, it shall change to the new packet type table and shall respond with an LMP_accepted_ext PDU. When it receives the baseband level acknowledgement for the LMP_accepted_ext PDU it shall restart ACL-U transmissions.

When the initiator receives an LMP_not_accepted_ext PDU the initiator shall restart ACL-U transmissions.

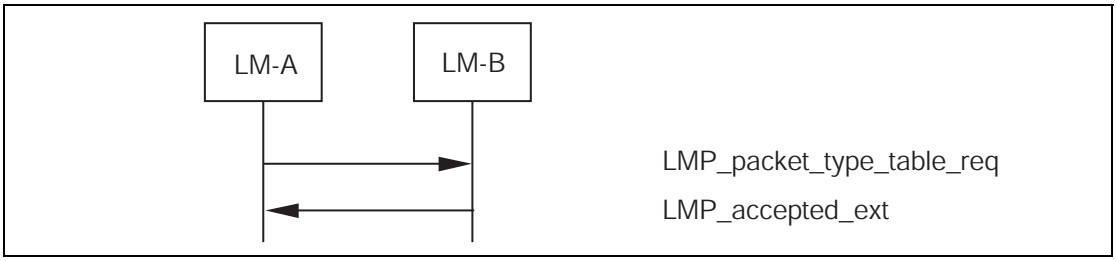
When the initiator receives an LMP_accepted_ext PDU it shall change the packet type table and restart ACL-U transmissions.

M/O	PDU	Contents
O(25)	LMP_packet_type_table_req	packet type table

Table 4.11: PDUs used for Enhanced Data Rate



Sequence 20: Packet type table change is rejected.



Sequence 21: Packet type table change is accepted.

4.2 SECURITY

4.2.1 Authentication

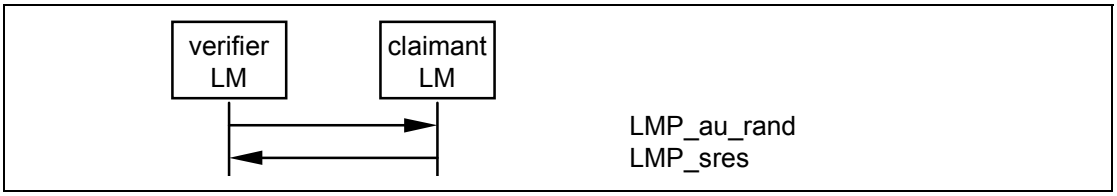
The authentication procedure is based on a challenge-response scheme as described in [Part H] Section 3.2.2 on page 780. The verifier sends an LMP_au_rand PDU that contains a random number (the challenge) to the claimant. The claimant calculates a response, that is a function of this challenge, the claimant's BD_ADDR and a secret key. The response is sent back to the verifier, that checks if the response was correct or not. The response shall be calculated as described in [Part H] Section 6.1 on page 801. A successful calculation of the authentication response requires that two devices share a secret key. This key is created as described in Section 4.2.2 on page 251. Both the master and the slave can be verifiers.

M/O	PDU	Contents
M	LMP_au_rand	random number
M	LMP_sres	authentication response

Table 4.12: PDUs used for authentication.

4.2.1.1 Claimant has link key

If the claimant has a link key associated with the verifier, it shall calculate the response and sends it to the verifier with LMP_sres. The verifier checks the response. If the response is not correct, the verifier can end the connection by sending an LMP_detach PDU with the error code *authentication failure,* see Section 4.1.2 on page 234.

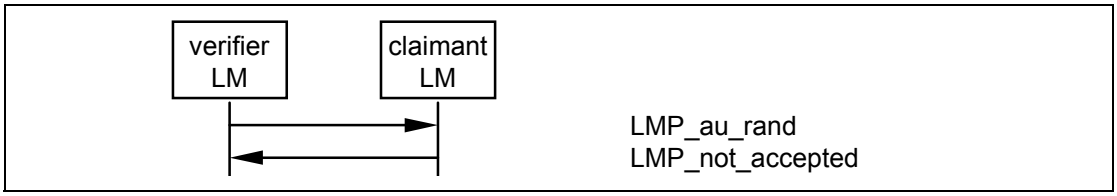


Sequence 22: Authentication. Claimant has link key.

Upon reception of an LMP_au_rand, an LM shall reply with LMP_sres before initiating its own authentication.
Note: there can be concurrent requests caused by the master and slave simultaneously initiating an authentication. The procedures in Section 2.5.1 on page 223 assures that devices will not have different Authenticated Ciphering Offset (ACO, see [Part H] Section 6.1 on page 801) when they calculate the encryption key.

4.2.1.2 Claimant has no link key

If the claimant does not have a link key associated with the verifier it shall send an LMP_not_accepted PDU with the error code *key missing* after receiving an LMP_au_rand PDU.



Sequence 23: Authentication fails. Claimant has no link key.

4.2.1.3 Repeated attempts

The scheme described in [Part H] Section 5.1 on page 799 shall be applied when an authentication fails. This will prevent an intruder from trying a large number of keys in a relatively short time.

4.2.2 Pairing

When two devices do not have a common link key an initialization key (K_{init}) shall be created based on a PIN, and a random number and a BD_ADDR. K_{init} shall be created as specified in [Part H] Section 6.3 on page 805. When both devices have calculated K_{init} the link key shall be created, and a mutual authentication is performed. The pairing procedure starts with a device sending an LMP_in_rand PDU; this device is referred to as the "initiating LM" or "initiator" in Section 4.2.2.1 on page 251 - Section 4.2.2.5 on page 253. The other device is referred to as the "responding LM" or "responder". The PDUs used in the pairing procedure are:

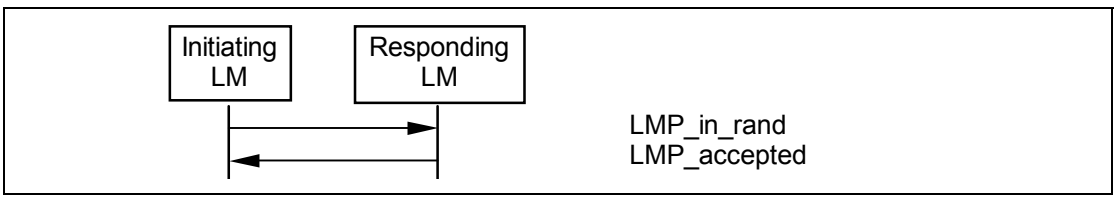
M/O	PDU	Contents
M	LMP_in_rand	random number
M	LMP_au_rand	random number
M	LMP_sres	authentication response
M	LMP_comb_key	random number
M	LMP_unit_key	key

Table 4.13: PDUs used for pairing

All sequences described in Section 4 on page 233, including the mutual authentication after the link key has been created, shall form a single transaction. The transaction ID from the first LMP_in_rand shall be used for all subsequent sequences.

4.2.2.1 Responder accepts pairing

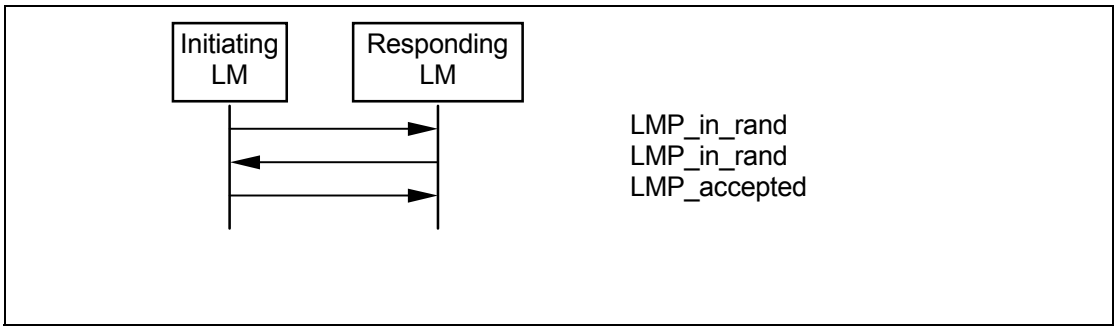
When the initiator sends an LMP_in_rand PDU and the responder shall reply with an LMP_accepted PDU. Both devices shall then calculate K_{init} based on the BD_ADDR of the responder and the procedure continues with creation of the link key; see Section 4.2.2.4 on page 253.



Sequence 24: Pairing accepted. Responder has a variable PIN. Initiator has a variable or fixed PIN.

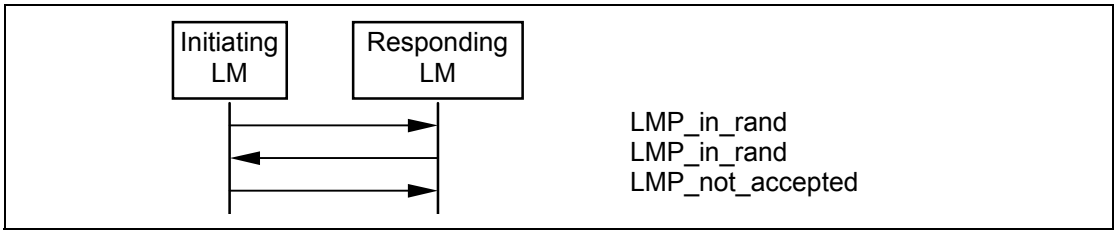
4.2.2.2 Responder has a fixed PIN

If the responder has a fixed PIN it shall generate a new random number and send it back in an LMP_in_rand PDU. If the initiator has a variable PIN it shall accept the LMP_in_rand PDU and shall respond with an LMP_accepted PDU. Both sides shall then calculate K_{init} based on the last IN_RAND and the BD_ADDR of the initiator. The procedure continues with creation of the link key; see Section 4.2.2.4 on page 253.



Sequence 25: Responder has a fixed PIN and initiator has a variable PIN.

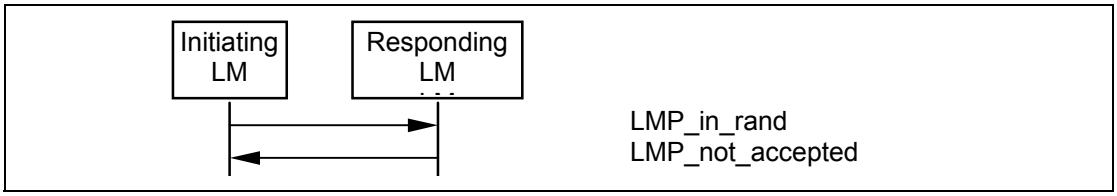
If the responder has a fixed PIN and the initiator also has a fixed PIN, the second LMP_in_rand shall be rejected by the initiator sending an LMP_not_accepted PDU with the error code *pairing not allowed*.



Sequence 26: Both devices have a fixed PIN.

4.2.2.3 Responder rejects pairing

If the responder rejects pairing it shall send an LMP_not_accepted PDU with the error code *pairing not allowed* after receiving an LMP_in_rand PDU.



Sequence 27: Responder rejects pairing.

4.2.2.4 Creation of the link key

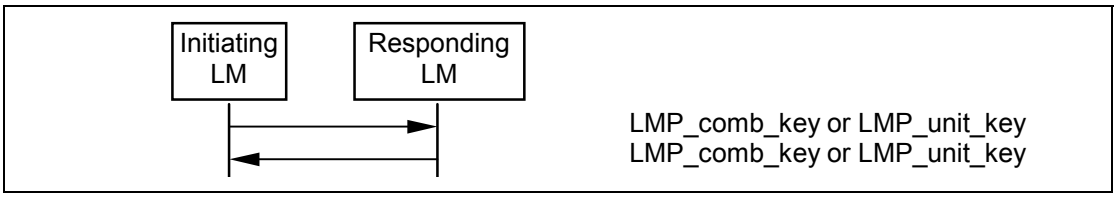
When K_{init} is calculated in both devices the link key shall be created. This link key will be used in the authentication between the two devices for all subsequent connections until it is changed; see Section 4.2.3 on page 254 and Section 4.2.4 on page 255. The link key created in the pairing procedure will either be a combination key or one of the device's unit keys. The following rules shall apply to the selection of the link key:

- if one device sends an LMP_unit_key PDU and the other device sends LMP_comb_key, the unit key will be the link key.
- if both devices send an LMP_unit_key PDU, the master's unit key will be the link key.
- if both devices send an LMP_comb_key PDU, the link key shall be calculated as described in [Part H] Section 3.2 on page 779.

The content of the LMP_unit_key PDU is the unit key bitwise XORed with K_{init}. The content of the LMP_comb_key PDU is LK_RAND bitwise XORed with K_{init}. Any device configured to use a combination key shall store the link key.

The use of unit keys is deprecated since it is implicitly insecure.

When the link key, combination or unit key, has been created mutual authentication shall be performed to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication is performed with the initiator as the verifier. When finalized an authentication in the reverse direction is performed.



Sequence 28: Creation of the link key.

4.2.2.5 Repeated attempts

When the authentication after creation of the link key fails because of an incorrect authentication response, the same scheme as in Section 4.2.1.3 on page 250 shall be used. This prevents an intruder from trying a large number of different PINs in a relatively short time.

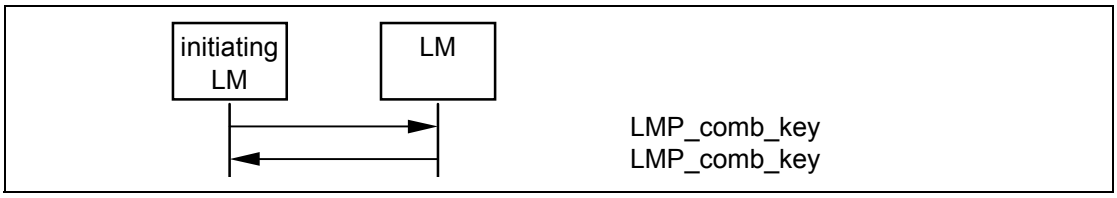
4.2.3 Change link key

If the link key is derived from combination keys and the current link is the semi-permanent link key, the link key can be changed. If the link key is a unit key, the devices shall go through the pairing procedure in order to change the link key. The contents of the LMP_comb_key PDU is protected by a bitwise XOR with the current link key.

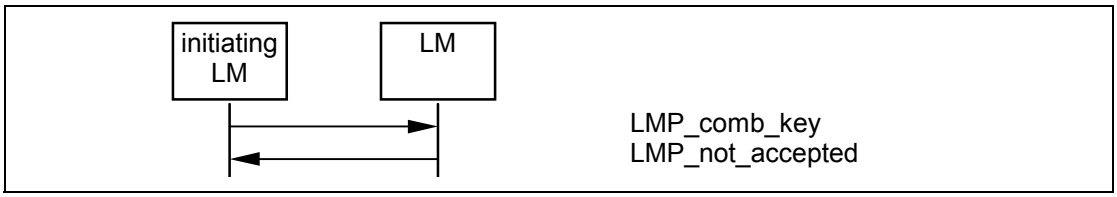
M/O	PDU	Contents
M	LMP_comb_key	random number

Table 4.14: PDUs used for change of link key.

All sequences described in Section 4.2.3 , including the mutual authentication after the link key has been changed, shall form a single transaction. The transaction ID from the first LMP_comb_key PDU shall be used for all subsequent sequences.



Sequence 29: Successful change of the link key.



Sequence 30: Change of the link key not possible since the other device uses a unit key.

If the change of link key is successful the new link key shall be stored and the old link key shall be discarded. The new link key shall be used as link key for all the following connections between the two devices until the link key is changed again. The new link key also becomes the current link key. It will remain the current link key until the link key is changed again, or until a temporary link key is created, see Section 4.2.4 on page 255.

When the new link key has been created mutual authentication shall be performed to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication is performed with the device that initiated change link key as verifier. When finalized an authentication in the reverse direction is performed.

4.2.4 Change current link key type

The current link key can be a semi-permanent link key or a temporary link key. It may be changed temporarily, but the change shall only be valid for the current connection, see [Part H] Section 3.1 on page 777. Changing to a temporary link key is necessary if the piconet is to support encrypted broadcast. The current link key may not be changed before the connection establishment procedure has completed. This feature is only supported if broadcast encryption is supported as indicated by the LMP features mask.

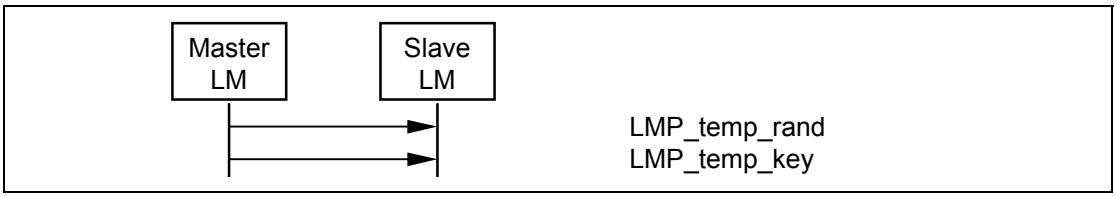
M/O	PDU	Contents
O(23)	LMP_temp_rand	random number
O(23)	LMP_temp_key	key
O(23)	LMP_use_semi_permanent_key	-

Table 4.15: PDUs used to change the current link key.

4.2.4.1 Change to a temporary link key

The master starts by creating the master key K_{master} as specified in Security Specification (EQ 4), on page 784. Then the master shall generate a random number, RAND, and shall send it to the slave in an LMP_temp_rand PDU. Both sides then calculate an overlay denoted OVL as OVL= E_{22}(current link key, RAND, 16). The master shall then send K_{master} protected by a modulo-2 addition with OVL to the slave in an LMP_temp_key PDU. The slave calculates K_{master}, based on OVL, that becomes the current link key. It shall be the current link key until the devices fall back to the semi-permanent link key, see section 4.2.4.2 on page 256.

Note: the terminology in this section is the same as used in [Part H] Section 3.2.8 on page 784.



Sequence 31: Change to a temporary link key.

All sequences described in Section 4.2.4.1 on page 255, including the mutual authentication after K_{master} has been created, shall form a single transaction. The transaction ID shall be set to 0.

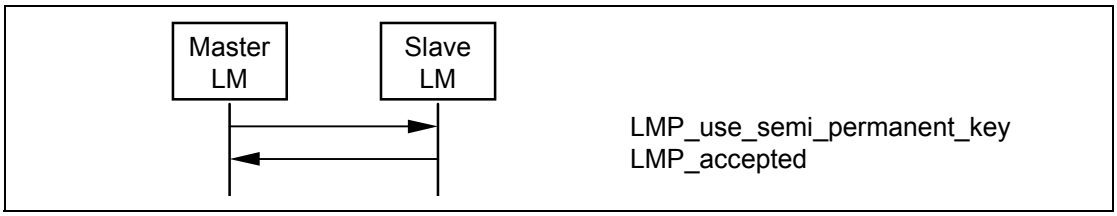
When the devices have changed to the temporary key, a mutual authentication shall be made to confirm that the same link key has been created in both devices. The first authentication in the mutual authentication shall be per-

formed with the master as verifier. When finalized an authentication in the reverse direction is performed.

Should the mutual authentication fail at either side, the LM of the verifier should start the detach procedure. This will allow the procedure to succeed even though one of the devices may be erroneous.

4.2.4.2 Make the semi-permanent link key the current link key

After the current link key has been changed to K_{master}, this change can be undone and the semi-permanent link key becomes the current link key again. If encryption is used on the link, the procedure to go back to the semi-permanent link key shall be immediately followed by the master stopping encryption using the procedure described in Section 4.2.5.4 on page 260. Encryption may be restarted by the master according to the procedures in section 4.2.5.1 on page 257 subsection 3. This is to assure that encryption with encryption parameters known by other devices in the piconet is not used when the semi-permanent link key is the current link key.



Sequence 32: Link key changed to the semi-permanent link key.

4.2.5 Encryption

If at least one authentication has been performed encryption may be used. In order for the master to use the same encryption parameters for all slaves in the piconet it shall issue a temporary key, K_{master}. The master shall make this key the current link key for all slaves in the piconet before encryption is started, see Section 4.2.4 on page 255. This is required if broadcast packets are to be encrypted.

M/O	PDU	Contents
O	LMP_encryption_mode_req	encryption mode
O	LMP_encryption_key_size_req	key size
O	LMP_start_encryption_req	random number
O	LMP_stop_encryption_req	-

Table 4.16: PDUs used for handling encryption.

All sequences described in Section 4.2.5 shall form a single transaction. The transaction ID from the LMP_encryption_mode_req PDU shall be used for all subsequent sequences.

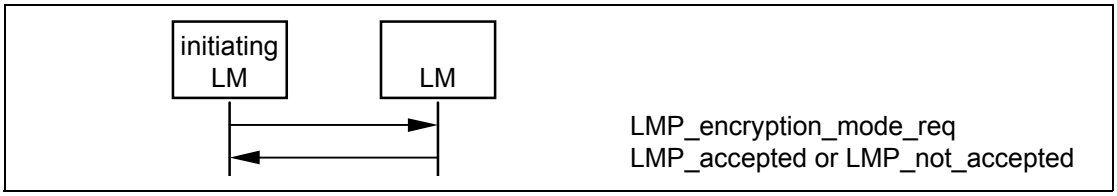
4.2.5.1 Encryption mode

The master and the slave must agree upon whether to use encryption (encryption mode=1 in lmp_encryption_mode_req) or not (encryption mode=0). If the semi-permanent key is used (Key_Flag=0x00) encryption shall only apply to point-to-point packets. If the master link key is used (Key_Flag=0x01) encryption shall apply to both point-to-point packets and broadcast packets. If master and slave agree on the encryption mode, the master continues to give more detailed information about the encryption.

Devices should never send LMP_encryption_mode_req with an encryption mode value of 2 however for backwards compatibility if the LMP_encryption_mode_req is received with an encryption mode value of 2 then it should be treated the same as an encryption mode value of 1.

The initiating LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). The initiating device shall then send the LMP_encryption_mode_req PDU. If the responding device accepts the change in encryption mode then it shall complete the transmission of the current packet on the ACL logical transport and shall then suspend transmission on the ACL-U logical link. The responding device shall then send the LMP_accepted PDU.

ACL-U logical link traffic shall only be resumed after the attempt to encrypt or decrypt the logical transport is completed i.e. at the end of Sequence 33, 34 or 35.

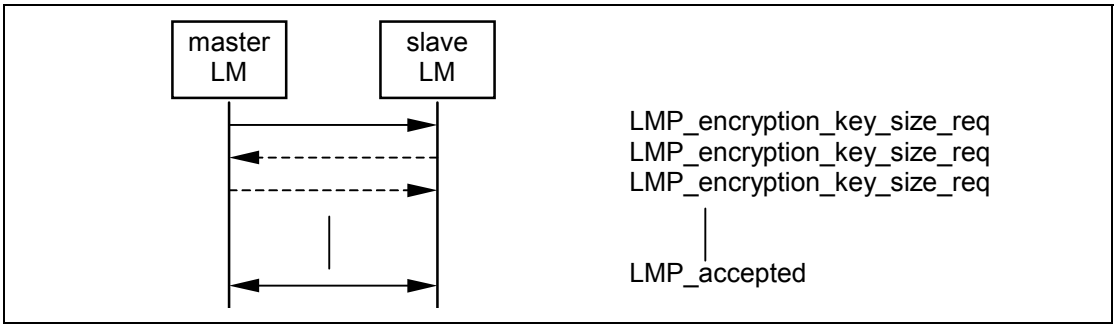


Sequence 33: Negotiation for encryption mode.

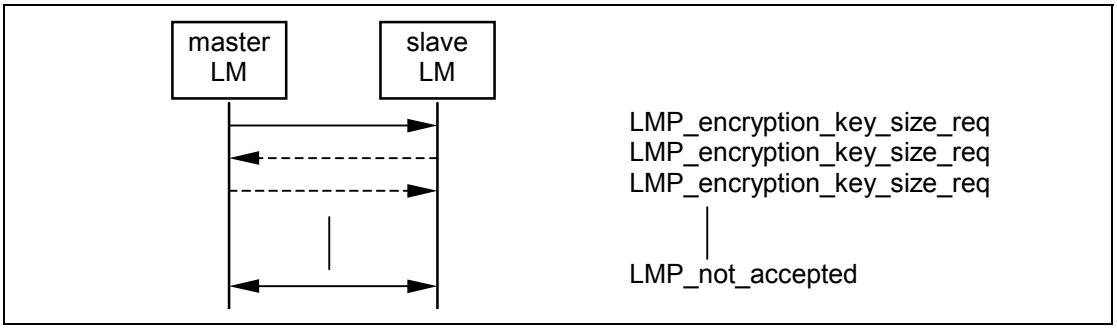
After a device has sent an LMP_encryption_mode_req PDU it shall not send an LMP_au_rand PDU before encryption is started. After a device has received an LMP_encryption_mode_req PDU and sent an LMP_accepted PDU it shall not send an LMP_au_rand PDU before encryption is started. If an LMP_au_rand PDU is sent violating these rules, the claimant shall respond with an LMP_not_accepted PDU with the error code *PDU not allowed*. This assures that devices will not have different ACOs when they calculate the encryption key. If the encryption mode is not accepted or the encryption key size negotiation results in disagreement the devices may send an LMP_au_rand PDU again.

4.2.5.2 Encryption key size

Note: this section uses the same terms as in [Part H] Section 4.1 on page 788. The master sends an LMP_encryption_key_size_req PDU including the suggested key size $L_{sug, m}$, that is initially equal to $L_{max, m}$. If $L_{min, s} \leq L_{sug, m}$ and the slave supports $L_{sug, m}$ it shall respond with an LMP_accepted PDU and $L_{sug, m}$ shall be used as the key size. If both conditions are not fulfilled the slave sends back an LMP_encryption_key_size_req PDU including the slave's suggested key size $L_{sug, s}$. This value shall be the slave's largest supported key size that is less than $L_{sug, m}$. Then the master performs the corresponding test on the slave's suggestion. This procedure is repeated until a key size agreement is reached or it becomes clear that no such agreement can be reached. If an agreement is reached a device sends an LMP_accepted PDU and the key size in the last LMP_encryption_key_size_req PDU shall be used. After this, encryption is started; see Section 4.2.5.3 on page 259. If an agreement is not reached a device sends an LMP_not_accepted PDU with the error code *unsupported parameter value* and the devices shall not communicate using encryption.



Sequence 34: Encryption key size negotiation successful.



Sequence 35: Encryption key size negotiation failed.

4.2.5.3 Start encryption

To start encryption, the master issues the random number EN_RAND and calculates the encryption key. See [Part H] Section 3.2.5 on page 782. The random number shall be the same for all slaves in the piconet when broadcast encryption is used. The master then sends an LMP_start_encryption_req PDU, that includes EN_RAND. The slave shall calculate the encryption key when this message is received and shall acknowledge with an LMP_accepted PDU.

Sequence 36: Start of encryption.

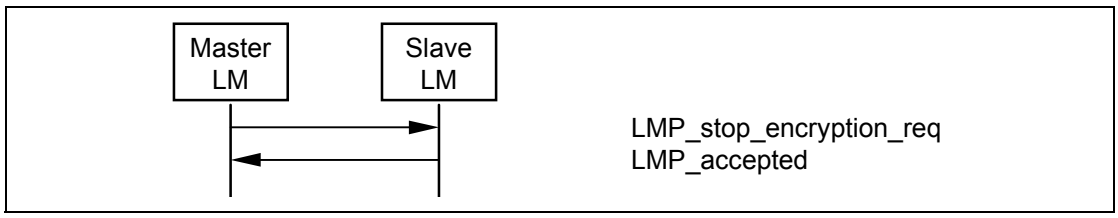
Starting encryption shall be performed in three steps:

1. Master is configured to transmit unencrypted packets and to receive encrypted packets.
2. Slave is configured to transmit and receive encrypted packets.
3. Master is configured to transmit and receive encrypted packets.

Between step 1 and step 2, master-to-slave transmission is possible. This is when an LMP_start_encryption_req PDU is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3, slave-to-master
transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

4.2.5.4 Stop encryption

To stop encryption a device shall send an LMP_encryption_mode_req PDU with the parameter encryption mode equal to 0 (no encryption). The other device responds with an LMP_accepted PDU or an LMP_not_accepted PDU (the procedure is described in Sequence 33 in section 4.2.5.1 on page 257). If accepted, encryption shall be stopped by the master sending an LMP_stop_encryption_req PDU and the slave shall respond with an LMP_accepted PDU according to Sequence 37.



Sequence 37: Stop of encryption.

Stopping encryption shall be performed in three steps, similar to the procedure for starting encryption.

1. Master is configured to transmit encrypted packets and to receive unencrypted packets.
2. Slave is configured to transmit and receive unencrypted packets.
3. Master is configured to transmit and receive unencrypted packets.

Between step 1 and step 2 master to slave transmission is possible. This is when an LMP_stop_encryption_req PDU is transmitted. Step 2 is triggered when the slave receives this message. Between step 2 and step 3 slave to master transmission is possible. This is when an LMP_accepted PDU is transmitted. Step 3 is triggered when the master receives this message.

4.2.5.5 Change encryption mode, key or random number

If the encryption key or encryption random number need to be changed or if the current link key needs to be changed according to the procedures in Section 4.2.4 on page 255, encryption shall be stopped and re-started after completion, using the procedures in Section 4.2.5 on page 257, subsections 3-4 for the new parameters to take effect.

4.2.6 Request supported encryption key size

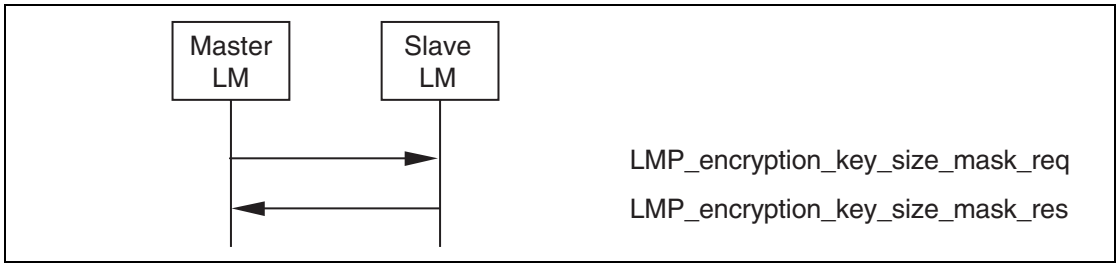
When broadcast encryption is supported via the LMP features mask, it is possible for the master to request a slave's supported encryption key sizes.

M/O	PDU	Contents
O(23)	LMP_encryption_key_size_mask_req	
O(23)	LMP_encryption_key_size_mask_res	key size mask

Table 4.17: PDUs used for encryption key size request

The master shall send an LMP_key_size_req PDU to the slave to obtain the slaves supported encryption key sizes.

The slave shall return a bit mask indicating all broadcast encryption key sizes supported. The least significant bit shall indicate support for a key size of 1, the next most significant bit shall indicate support for a key size of 2 and so on up to a key size of 16. In all cases a bit set to 1 shall indicate support for a key size; a bit set to 0 shall indicate that the key size is not supported.



Sequence 38: Request for supported encryption key sizes.

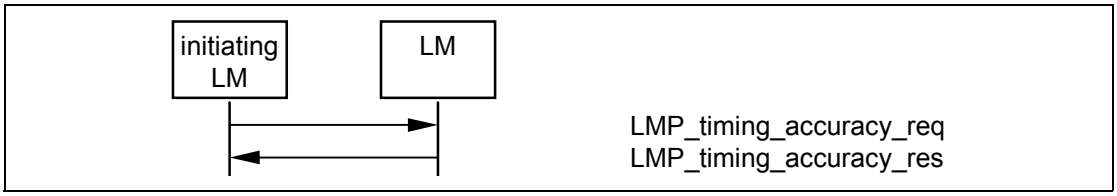
4.3 INFORMATIONAL REQUESTS

4.3.1 Timing accuracy

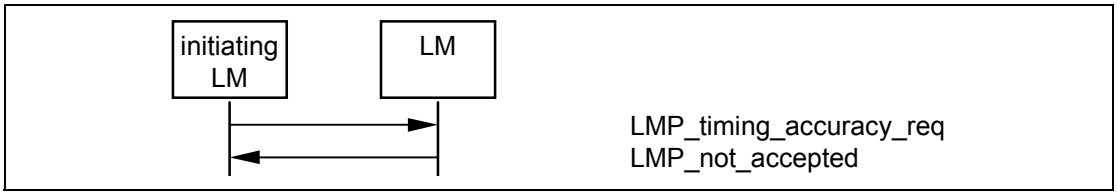
LMP supports requests for the timing accuracy. This information can be used to minimize the scan window during piconet physical channel re-synchronization (see Baseband Specification, Section 2.2.5.2, on page 74). The timing accuracy parameters returned are the long term drift measured in ppm and the long term jitter measured in μs of the worst case clock used. These parameters are fixed for a certain device and shall be identical when requested several times. Otherwise, the requesting device shall assume worst case values (drift=250ppm and jitter=10μs).

M/O	PDU	Contents
O(4)	LMP_timing_accuracy_req	-
O(4)	LMP_timing_accuracy_res	drift jitter

Table 4.18: PDUs used for requesting timing accuracy information.



Sequence 39: The requested device supports timing accuracy information.



Sequence 40: The requested device does not support timing accuracy information.

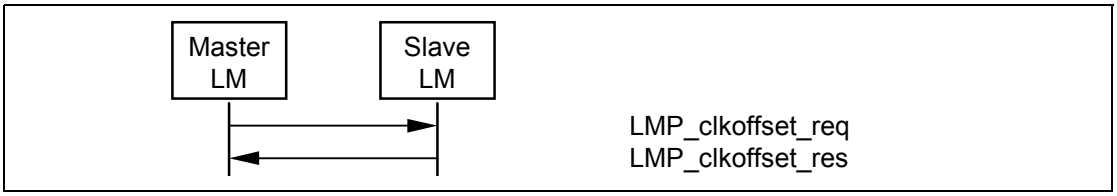
4.3.2 Clock offset

The clock offset can be used to speed up the paging time the next time the same device is paged. The master can request the clock offset at anytime following a successful baseband paging procedure (i.e., before, during or after connection setup). The clock offset shall be defined by the following equation:

$(CLKN_{16\text{-}2\ slave} - CLKN_{16\text{-}2\ master})$ mod 2**15.

M/O	PDU	Contents
M	LMP_clkoffset_req	-
M	LMP_clkoffset_res	clock offset

Table 4.19: PDUs used for clock offset request.



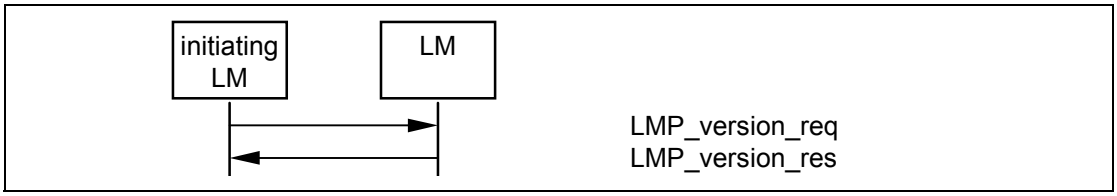
Sequence 41: Clock offset requested.

4.3.3 LMP version

LMP supports requests for the version of the LM protocol. The LMP_version_req and LMP_version_res PDUs contain three parameters: VersNr, CompId and SubVersNr. VersNr specifies the version of the Bluetooth LMP specification that the device supports. CompId is used to track possible problems with the lower Bluetooth layers. All companies that create a unique implementation of the LM shall have their own CompId. The same company is also responsible for the administration and maintenance of the SubVersNr. It is recommended that each company has a unique SubVersNr for each RF/BB/LM implementation. For a given VersNr and CompId, the values of the SubVersNr shall increase each time a new implementation is released. For both CompId and SubVersNr the value 0xFFFF means that no valid number applies. There is no ability to negotiate the version of the LMP. The sequence below is only used to exchange the parameters. LMP version can be requested at anytime following a successful baseband paging procedure.

M/O	PDU	Contents
M	LMP_version_req	VersNr CompId SubVersNr
M	LMP_version_res	VersNr CompId SubVersNr

Table 4.20: PDUs used for LMP version request.



Sequence 42: Request for LMP version.

4.3.4 Supported features

The supported features may be requested at anytime following a successful baseband paging procedure by sending the LMP_features_req PDU. Upon reception of an LMP_features_req PDU, the receiving device shall return an LMP_features_res PDU.

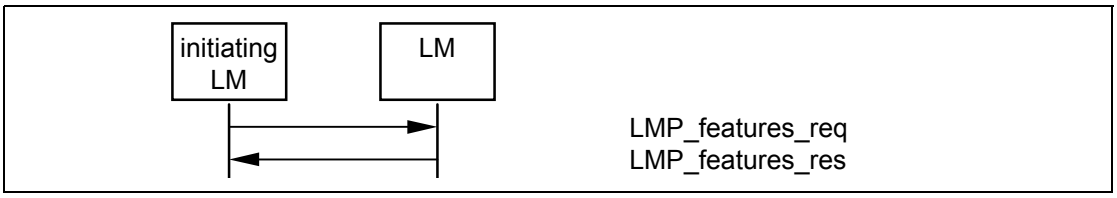
The number of features bits required will in the future exceed the size of a single page of features. An extended features mask is therefore provided to allow support for more than 64 features. Support for the extended features mask is indicated by the presence of the appropriate bit in the LMP features mask. The LMP_features_req_ext and LMP_features_res_ext PDUs operate in precisely the same way as the LMP_features_req and LMP_features_res PDUs except that they allow the various pages of the extended features mask to be requested. The LMP_features_req_ext may be sent at any time following the exchange of the LMP_features_req and LMP_features_rsp PDUs.

The LMP_features_req_ext PDU contains a feature page index that specifies which page is requested and the contents of that page for the requesting device. Pages are numbered from 0-255 with page 0 corresponding to the normal features mask. Each page consists of 64 bits. If a device does not support any page number it shall return a mask with every bit set to 0. It also contains the maximum features page number containing any non-zero bit for this device. The recipient of an LMP_features_req_ext PDU shall respond with an LMP_features_res_ext PDU containing the same page number and the appropriate features page along with its own maximum features page number.

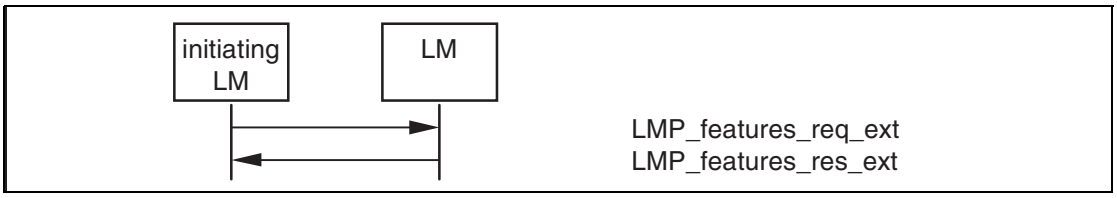
If the extended features request is not supported then all bits in all extended features pages for that device shall be assumed to be zero.

M/O	PDU	Contents
M	LMP_features_req	features
M	LMP_features_res	features
O(63)	LMP_features_req_ext	features page max supported page extended features
O(63)	LMP_features_res_ext	features page max supported page extended features

Table 4.21: PDUs used for features request.



Sequence 43: Request for supported features.



Sequence 44: Request for extended features.

4.3.5 Name request

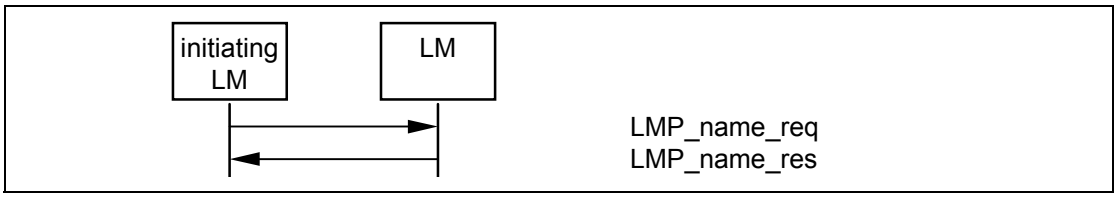
LMP supports name request to another device. The name is a user-friendly name associated with the device and consists of a maximum of 248 bytes coded according to the UTF-8 standard. The name is fragmented over one or more DM1 packets. When an LMP_name_req PDU is sent, a name offset indicates which fragment is expected. The corresponding LMP_name_res PDU carries the same name offset, the name length indicating the total number of bytes in the name of the device and the name fragment, where:

- name fragment(N) = name(N + name offset), if (N + name offset) < name length
- name fragment(N) = 0,otherwise.

Here $0 \leq N \leq 13$. In the first sent LMP_name_req PDU, name offset=0. Sequence 45 is then repeated until the initiator has collected all fragments of the name. The name request may be made at any time following a successful baseband paging procedure.

M/O	PDU	Contents
M	LMP_name_req	name offset
M	LMP_name_res	name offset name length name fragment

Table 4.22: PDUs used for name request.



Sequence 45: Device's name requested and it responses.

4.4 ROLE SWITCH

4.4.1 Slot offset

With LMP_slot_offset the information about the difference between the slot boundaries in different piconets is transmitted. The LMP_slot_offset PDU may be sent anytime after the baseband paging procedure has completed. This PDU carries the parameters slot offset and BD_ADDR. The slot offset shall be the time in microseconds between the start of a master transmission in the current piconet to the start of the next following master transmission in the piconet where the BD_ADDR device (normally the slave) is master at the time that the request is interpreted by the BD_ADDR device.

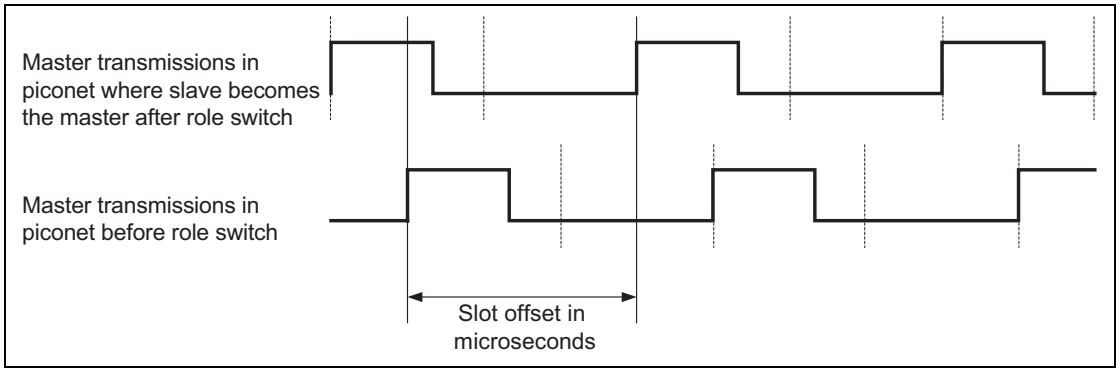
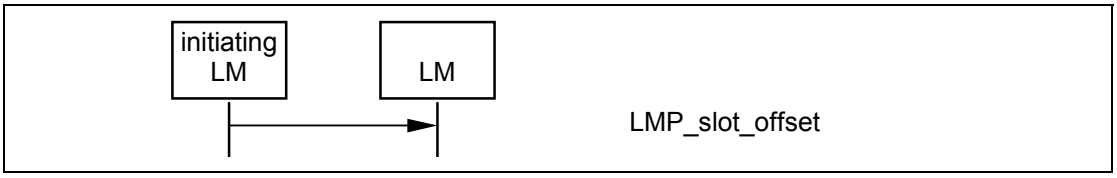


Figure 4.2: Slot offset for role switch.

See Section 4.4 on page 267 for the use of LMP_slot_offset in the context of the role switch. In the case of role switch the BD_ADDR is that of the slave device.

M/O	PDU	Contents
O(3)	LMP_slot_offset	slot offset BD_ADDR

Table 4.23: PDU used for slot offset information.



Sequence 46: Slot offset information is sent.

4.4.2 Role switch

Since the paging device always becomes the master of the piconet, a switch of the master slave role is sometimes needed, see Baseband Specification, Section 8.6.5, on page 175. The LMP_switch_req PDU may be sent anytime after the baseband paging procedure has completed.

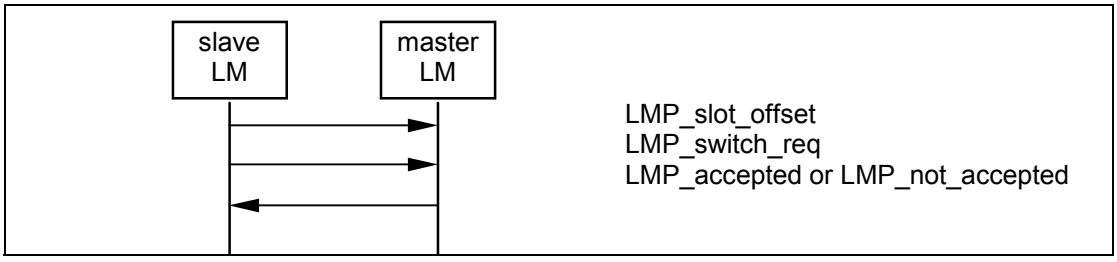
Support for LMP_slot_offset is mandatory if LMP_switch_req is supported.

The LMP_slot_offset shall be sent only if the ACL logical transport is in active mode. The LMP_switch_req shall be sent only if the ACL logical transport is in active mode, when encryption is disabled, and all synchronous logical transports on the same physical link are disabled. Additionally, LMP_slot_offset or LMP_switch_req shall not be initiated or accepted while a synchronous logical transport is being negotiated by LM.

M/O	PDU	Contents
O(5)	LMP_switch_req	switch instant
O(5)	LMP_slot_offset	slot offset BD_ADDR

Table 4.24: PDUs used for role switch.

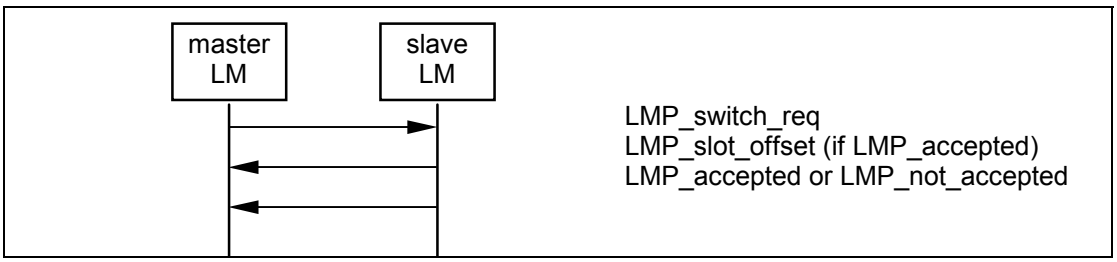
The initiating LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). It shall then send an LMP_slot_offset PDU immediately followed by an LMP_switch_req PDU. If the master accepts the role switch it shall pause traffic on the ACL-U logical link (seeBaseband Specification, Section 5.3.1, on page 108) and respond with an LMP_accepted PDU. When the role switch has been completed at the baseband level (successfully or not) both devices re-enable transmission on the ACL-U logical link. If the master rejects the role switch it responds with an LMP_not_accepted PDU and the slave re-enables transmission on the ACL-U logical link. The transaction ID for all PDUs in the sequence shall be set to 1.



Sequence 47: Role switch (slave initiated).

If the master initiates the role switch it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108) and send an LMP_switch_req PDU. If the slave accepts the role switch it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108) and responds with an LMP_slot_offset PDU immediately followed by an

LMP_accepted PDU. When the role switch has been completed at the baseband (successfully or not) both devices re-enable transmission on the ACL-U logical link. If the slave rejects the role switch it responds with an LMP_not_accepted PDU and the master re-enables transmission on the ACL-U logical link. The transaction ID for all PDUs in the sequence shall be set to 0.



Sequence 48: Role switch (master initiated).

The LMP_switch_req PDU contains a parameter, switch instant, which specifies the instant at which the TDD switch is performed. This is specified as a Bluetooth clock value of the master's clock, that is available to both devices. This instant is chosen by the sender of the message and shall be at least $2*T_{poll}$ or 32 (whichever is greater) slots in the future. The switch instant shall be within 12 hours of the current clock value to avoid clock wrap.

The sender of the LMP_switch_req PDU selects the switch instant and queues the LMP_switch_req PDU to LC for transmission and starts a timer to expire at the switch instant. When the timer expires it initiates the mode switch. In the case of a master initiated switch if the LMP_slot_offset PDU has not been received by the switch instant the role switch is carried out without an estimate of the slave's slot offset. If an LMP_not_accepted PDU is received before the timer expires then the timer is stopped and the role switch shall not be initiated.

When the LMP_switch_req is received the switch instant is compared with the current master clock value. If it is in the past then the instant has been passed and an LMP_not_accepted PDU with the error code *instant passed* shall be returned. If it is in the future then an LMP_accepted PDU shall be returned assuming the role switch is allowed and a timer is started to expire at the switch instant. When this timer expires the role switch shall be initiated.

After a successful role switch the supervision timeout and poll interval (T_{poll}) shall be set to their default values. The authentication state and the ACO shall remain unchanged. Adaptive Frequency Hopping shall follow the procedures described in Baseband Specification, Section 8.6.5, on page 175. The default value for max_slots shall be used.

4.5 MODES OF OPERATION

4.5.1 Hold mode

The ACL logical transport of a connection between two Bluetooth devices can be placed in hold mode for a specified hold time. See Baseband Specification, Section 8.8, on page 185 for details.

M/O	PDU	Contents
O(6)	LMP_hold	hold time, hold instant
O(6)	LMP_hold_req	hold time, hold instant

Table 4.25: PDUs used for hold mode.

The LMP_hold and LMP_hold_req PDUs both contain a parameter, hold instant, that specifies the instant at which the hold becomes effective. This is specified as a Bluetooth clock value of the master's clock, that is available to both devices. The hold instant is chosen by the sender of the message and should be at least $6*T_{poll}$ slots in the future. The hold instant shall be within 12 hours of the current clock value to avoid clock wrap.

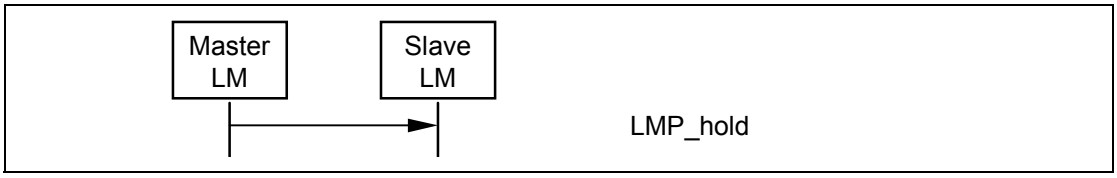
4.5.1.1 Master forces hold mode

The master may force hold mode if there has previously been a request for hold mode that has been accepted. The hold time included in the PDU when the master forces hold mode shall not be longer than any hold time the slave has previously accepted when there was a request for hold mode.

The master LM shall first pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). It shall select the hold instant and queue the LMP_hold PDU to its LC for transmission. It shall then start a timer to wait until the hold instant occurs. When this timer expires then the connection shall enter hold mode. If the baseband acknowledgement for the LMP_hold PDU is not received then the master may enter hold mode, but it shall not use its low accuracy clock during the hold.

When the slave LM receives an LMP_hold PDU it compares the hold instant with the current master clock value. If it is in the future then it starts a timer to expire at this instant and enters hold mode when it expires.

When the master LM exits from Hold mode it re-enables transmission on the ACL-U logical link.



Sequence 49: Master forces slave into hold mode.

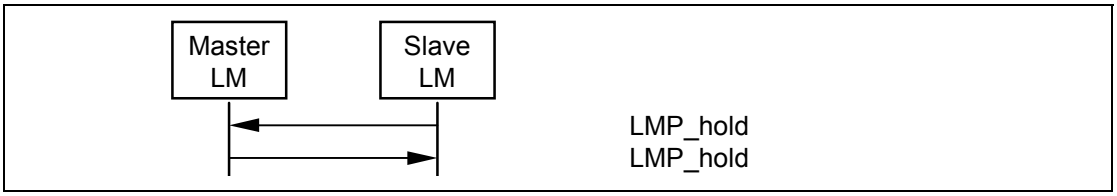
4.5.1.2 Slave forces hold mode

The slave may force hold mode if there has previously been a request for hold mode that has been accepted. The hold time included in the PDU when the slave forces hold mode shall not be longer than any hold time the master has previously accepted when there was a request for hold mode.

The slave LM shall first complete the transmission of the current packet on the ACL logical transport and then shall suspend transmission on the ACL-U logical link. It shall select the hold instant and queue the LMP_hold PDU to its LC for transmission. It shall then wait for an LMP_hold PDU from the master acting according to the procedure described in Section 4.5.1.1 .

When the master LM receives an LMP_hold PDU it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). It shall then inspect the hold instant. If this is less than $6*T_{poll}$ slots in the future it shall modify the instant so that it is at least $6*T_{poll}$ slots in the future. It shall then send an LMP_hold PDU using the mechanism described in Section 4.5.1.1 .

When the master and slave LMs exit from Hold mode they shall re-enable transmission on the ACL-U logical link.



Sequence 50: Slave forces master into hold mode.

4.5.1.3 Master or slave requests hold mode

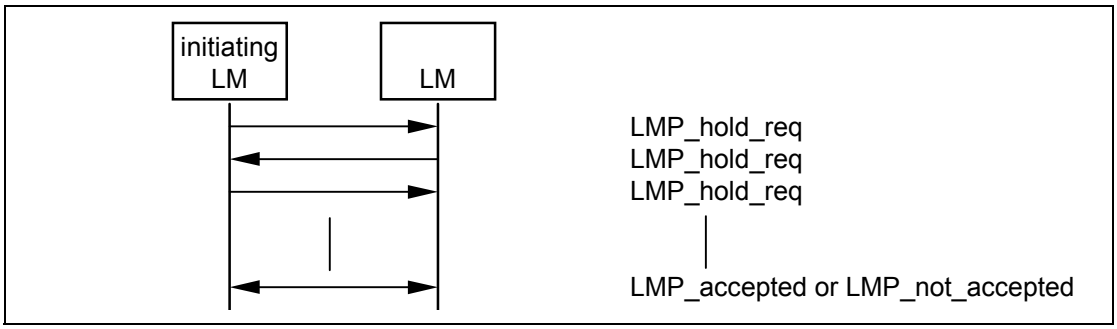
The master or the slave can request to enter hold mode. Upon receipt of the request, the same request with modified parameters can be returned or the negotiation can be terminated. If an agreement is seen an LMP_accepted PDU terminates the negotiation and the ACL link is placed in hold mode. If no agreement is seen, an LMP_not_accepted PDU with the error code *unsupported parameter value* terminates the negotiation and hold mode is not entered.

The initiating LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). On receiving an LMP_hold_req PDU the receiving LM shall complete the transmission of the current packet on the ACL logical transport and then shall suspend transmission on the ACL-U logical link.

The LM sending the LMP_hold_req PDU selects the hold instant, that shall be at least $9*T_{poll}$ slots in the future. If this is a response to a previous LMP_hold_req PDU and the contained hold instant is at least $9*T_{poll}$ slots in the

future then this shall be used. The LMP_hold_req PDU shall then be queued to its LC for transmission and a timer shall be started to expire at this instant and the connection enters hold mode when it expires unless an LMP_not_accepted or LMP_hold_req PDU is received by its LM before that point. If the LM receiving LMP_hold_req PDU agrees to enter hold mode it shall return an LMP_accepted PDU and shall start a timer to expire at the hold instant. When this timer expires it enters hold mode.

When each LM exits from Hold mode it shall re-enable transmission on the ACL-U logical link.



Sequence 51: Negotiation for hold mode.

4.5.2 Park state

If a slave does not need to participate in the channel, but should still remain synchronized to the master, it may be placed in park state. See Baseband Specification, Section 8.9, on page 185 for details.

Note: to keep a parked slave connected the master shall periodically unpark and repark the slave if the supervision timeout is not set to zero (see Baseband Specification, Section 3.1, on page 95).

All PDUs sent from the master to parked slaves are carried on the PSB-C logical link (LMP link of parked slave broadcast logical transport). These PDUs, LMP_set_broadcast_scan_window, LMP_modify_beacon, LMP_unpark_BD_addr_req and LMP_unpark_PM_addr_req, are the only PDUs that shall be sent to a slave in park state and the only PDUs that shall be broadcast. To increase reliability for broadcast, the packets are as short as possible. Therefore the format for these LMP PDUs are somewhat different. The parameters are not always byte-aligned and the length of the PDUs is variable.

The messages for controlling park state include parameters, defined in Baseband Specification, Section 8.9, on page 185. When a slave is placed in park state it is assigned a unique PM_ADDR, that can be used by the master to unpark that slave. The all-zero PM_ADDR has a special meaning; it is not a valid PM_ADDR. If a device is assigned this PM_ADDR, it shall be identified with its BD_ADDR when it is unparked by the master.

M/O	PDU	Contents
O(8)	LMP_park_req	timing control flags D_B T_B N_B Δ_B PM_ADDR AR_ADDR N_{Bsleep} D_{Bsleep} D_{access} T_{access} $N_{acc\text{-}slots}$ N_{poll} M_{access} access scheme
O(8)	LMP_set_broadcast_scan_window	timing control flags D_B (optional) broadcast scan window
O(8)	LMP_modify_beacon	timing control flags D_B (optional) T_B N_B Δ_B D_{access} T_{access} $N_{acc\text{-}slots}$ N_{poll} M_{access} access scheme

Table 4.26: PDUs used for park state.

M/O	PDU	Contents
O(8)	LMP_unpark_PM_ADDR_req	timing control flags D_B (optional) LT_ADDR 1st unpark LT_ADDR 2nd unpark PM_ADDR 1st unpark PM_ADDR 2nd unpark LT_ADDR 3rd unpark LT_ADDR 4th unpark PM_ADDR 3rd unpark PM_ADDR 4th unpark LT_ADDR 5th unpark LT_ADDR 6th unpark PM_ADDR 5th unpark PM_ADDR 6th unpark LT_ADDR 7th unpark PM_ADDR 7th unpark
O(8)	LMP_unpark_BD_ADDR_req	timing control flags D_B (optional) LT_ADDR LT_ADDR (optional) BD_ADDR BD_ADDR (optional)

Table 4.26: PDUs used for park state.

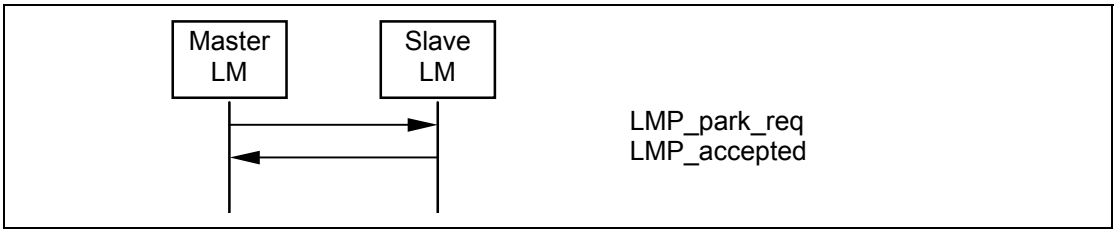
4.5.2.1 Master requests slave to enter park state

The master can request park state. The master LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108) and then send an LMP_park_req PDU. If the slave agrees to enter park state it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108). and then respond with an LMP_accepted PDU.

When the slave queues an LMP_accepted PDU it shall start a timer for 6*Tpoll slots. If the baseband acknowledgement is received before this timer expires it shall enter park state immediately otherwise it shall enter park state when the timer expires.

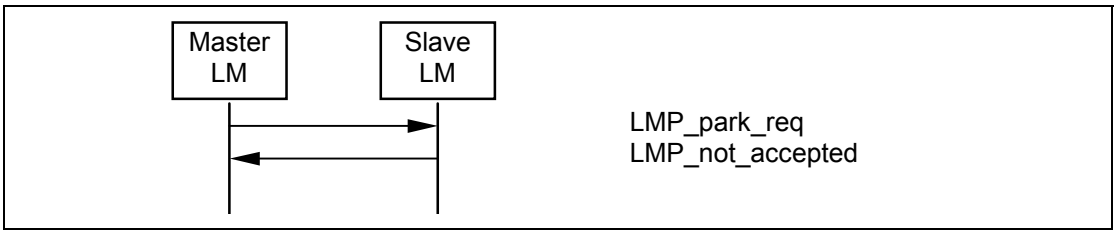
When the master receives an LMP_accepted PDU it shall start a timer for 6*Tpoll slots. When this timer expires the slave is in park state and the LT_ADDR may be re-used.

If the master never receives an LMP_accepted PDU then a link supervision timeout will occur.



Sequence 52: Slave accepts to enter park state.

If the slave rejects the attempt to enter park state it shall respond with an LMP_not_accepted PDU and the master shall re-enable transmission on the ACL-U logical link.



Sequence 53: Slave rejects to enter into park state

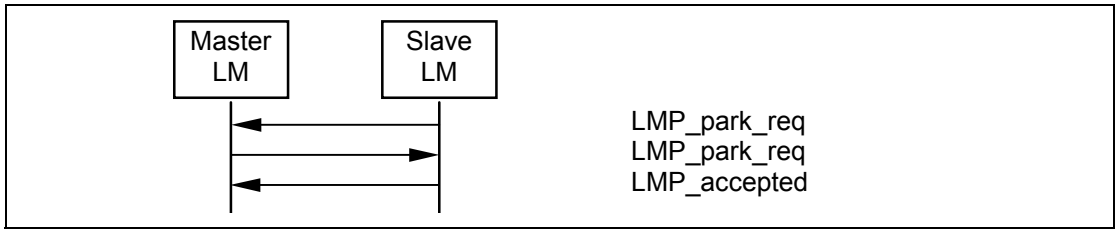
4.5.2.2 Slave requests to enter park state

The slave can request park state. The slave LM shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108) and then send an LMP_park_req PDU. When sent by the slave, the parameters PM_ADDR and AR_ADDR are not valid and the other parameters represent suggested values. If the master accepts the slave's request to enter park state it shall pause traffic on the ACL-U logical link (see Baseband Specification, Section 5.3.1, on page 108) and then send an LMP_park_req PDU, where the parameter values may be different from the values in the PDU sent from the slave. If the slave can accept these parameter it shall respond with an LMP_accepted PDU.

When the slave queues an LMP_accepted PDU for transmission it shall start a timer for 6*Tpoll slots. If the baseband acknowledgement is received before this timer expires it shall enter park state immediately otherwise it shall enter park state when the timer expires.

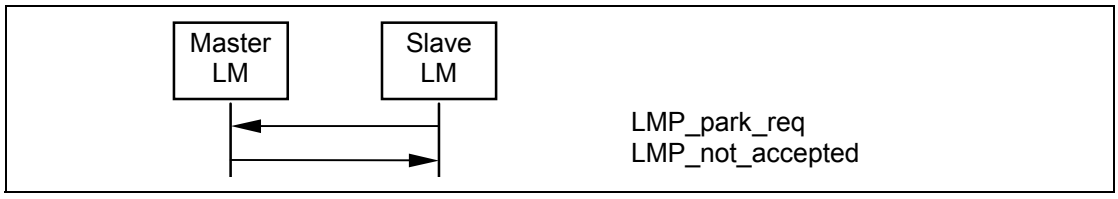
When the master receives an LMP_accepted PDU it shall start a timer for 6*Tpoll slots. When this timer expires the slave is in park state and the LT_ADDR may be re-used.

If the master never receives the LMP_accepted PDU then a link supervision timeout will occur.



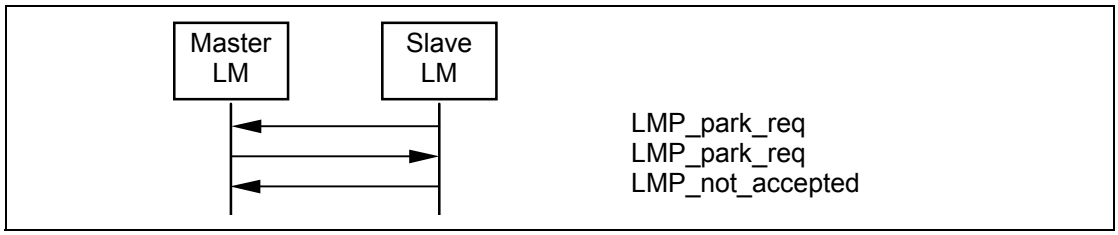
Sequence 54: Slave requests to enter park state and accepts master's beacon parameters.

If the master does not agree that the slave enters park state it shall send an LMP_not_accepted PDU. The slave shall then re-enable transmission on the ACL-U logical link.



Sequence 55: Master rejects slave's request to enter park state

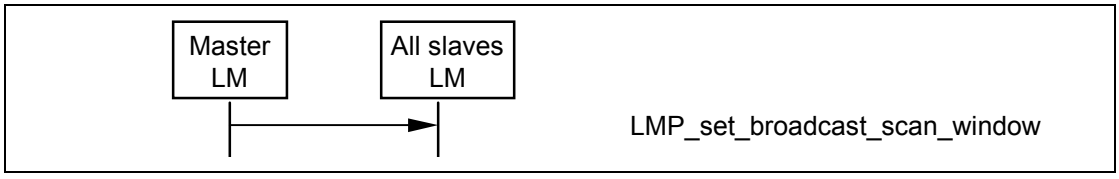
If the slave does not accept the parameters in the LMP_park_req PDU sent from the master it shall respond with an LMP_not_accepted PDU and both devices shall re-enable transmission on the ACL-U logical link.



Sequence 56: Slave requests to enter park state, but rejects master's beacon parameters.

4.5.2.3 Master sets up broadcast scan window

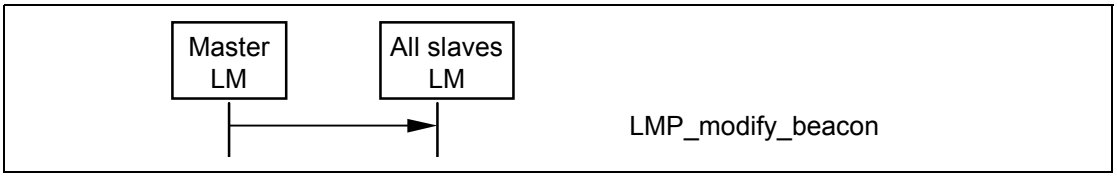
If more broadcast capacity is needed than the beacon train, the master may indicate to the slaves that more broadcast information will follow the beacon train by sending an LMP_set_broadcast_scan_window PDU. This message shall be sent in a broadcast packet at the beacon slot(s). The scan window shall start in the beacon instant and shall only be valid for the current beacon.



Sequence 57: Master notifies all slaves of increase in broadcast capacity.

4.5.2.4 Master modifies beacon parameters

When the beacon parameters change the master notifies the parked slaves of this by sending an LMP_modify_beacon PDU. This PDU shall be sent in a broadcast packet.



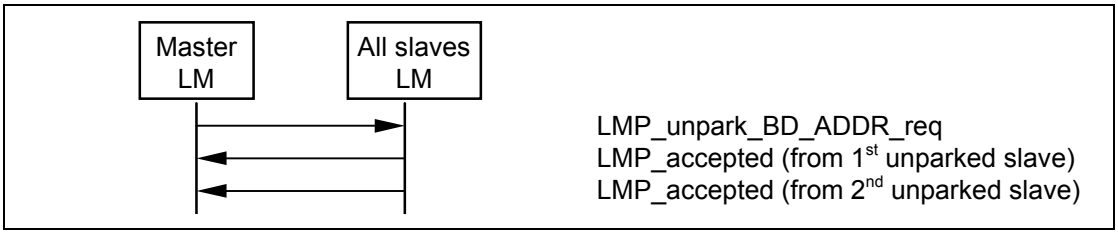
Sequence 58: Master modifies beacon parameters.

4.5.2.5 Unparking slaves

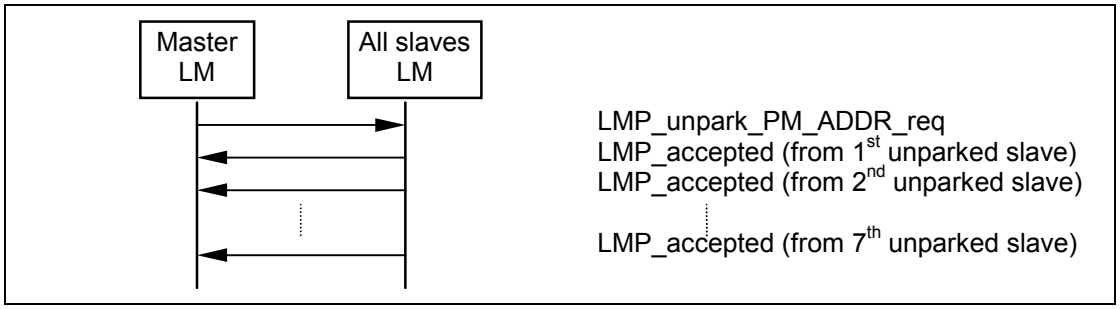
The master can unpark one or many slaves by sending a broadcast LMP message including the PM_ADDR or the BD_ADDR of the device(s) to be unparked. Broadcast LMP messages are carried on the PSB-C logical link. See Baseband Specification, Section 8.9.5, on page 192 for further details. This message also includes the LT_ADDR that the master assigns to the slave(s). After sending this message, the master shall check the success of the unpark by polling each unparked slave by sending POLL packets, so that the slave is granted access to the channel. The unparked slave shall then send a response with an LMP_accepted PDU. If this message is not received from the slave within a certain time after the master sent the unpark message, the unpark failed and the master shall consider the slave as still being in park state.

One PDU is used where the parked device is identified with the PM_ADDR, and another PDU is used where it is identified with the BD_ADDR. Both messages have variable length depending on the number of slaves the master unparks. For each slave the master wishes to unpark an LT_ADDR followed by the PM_ADDR or BD_ADDR of the device that is assigned this LT_ADDR is included in the payload. If the slaves are identified with the PM_ADDR a maximum of 7 slaves can be unparked with the same message. If they are identified with the BD_ADDR a maximum of 2 slaves can be unparked with the same message.

After a successful unparking, both devices re-enable transmission on the ACL-U logical link.



Sequence 59: Master unparks slaves addressed with their BD_ADDR.



Sequence 60: Master unparks slaves addressed with their PM_ADDR.

4.5.3 Sniff mode

To enter sniff mode, master and slave negotiate a sniff interval T_{sniff} and a sniff offset, D_{sniff}, that specifies the timing of the sniff slots. The offset determines the time of the first sniff slot; after that the sniff slots follow periodically with the sniff interval T_{sniff}. To avoid clock wrap-around during the initialization, one of two options is chosen for the calculation of the first sniff slot. A timing control flag in the message from the master indicates this.Only bit1 of the timing control flag is valid.

When the ACL logical transport is in sniff mode the master shall only start a transmission in the sniff slots. Two parameters control the listening activity in the slave: the sniff attempt and the sniff timeout. The sniff attempt parameter determines for how many slots the slave shall listen when the slave is not treating this as a scatternet link, beginning at the sniff slot, even if it does not receive a packet with its own LT_ADDR. The sniff timeout parameter determines for how many additional slots the slave shall listen when the slave is not treating this as a scatternet link if it continues to receive only packets with its own LT_ADDR. It is not possible to modify the sniff parameters while the device is in sniff mode.

M/O	PDU	Contents
O(7)	LMP_sniff_req	timing control flags D_{sniff} T_{sniff} sniff attempt sniff timeout
O(7)	LMP_unsniff_req	-

Table 4.27: PDUs used for sniff mode.

4.5.3.1 Master or slave requests sniff mode

Either the master or the slave may request entry to sniff mode.

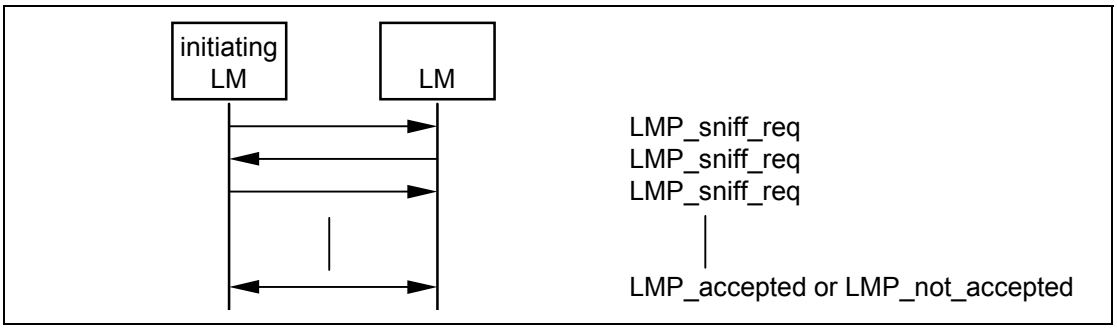
The process is initiated by sending an LMP_sniff_req PDU containing a set of parameters. The receiving LM shall then decide whether to reject the attempt by sending an LMP_not_accepted PDU, to suggest different parameters by replying with an LMP_sniff_req PDU or to accept the request.

Before the first time that the master sends LMP_sniff_req it shall enter sniff transition mode. If the master receives or sends an LMP_not_accepted PDU it shall exit from sniff transition mode. If the master receives an LMP_sniff_req PDU it shall enter sniff transition mode.

If the master decides to accept the request it shall send an LMP_accepted PDU. When the master receives the baseband acknowledgement for this PDU it shall exit sniff transition mode and enter sniff mode.

If the master receives an LMP_accepted PDU the master shall exit from sniff transition mode and enter sniff mode.

If the slave receives an LMP_sniff_req PDU it must decide whether to accept the request. If the slave does not wish to enter sniff mode then it replies with an LMP_not_accepted PDU. If it is happy to enter sniff mode but requires a different set of parameters it shall respond with an LMP_sniff_req PDU containing the new parameters. If the slave decides that the parameters are acceptable then it shall send an LMP_accepted PDU and enter sniff mode. If the slave receives an LMP_not_accepted PDU it shall terminate the attempt to enter sniff mode.



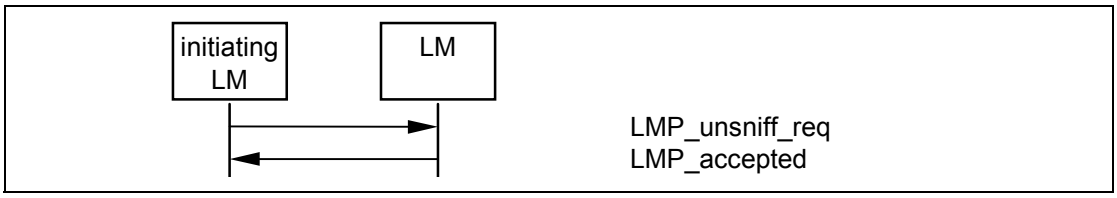
Sequence 61: Negotiation for sniff mode.

4.5.3.2 Moving a slave from sniff mode to active mode

Sniff mode may be exited by either the master or the slave sending an LMP_unsniff_req PDU. The requested device must reply with an LMP_accepted PDU.

If the master requests an exit from sniff mode it shall enter sniff transition mode and then send an LMP_unsniff_req PDU. When the slave receives the LMP_unsniff_req it shall exit from sniff mode and reply with an LMP_accepted PDU. When the master receives the LMP_accepted PDU it shall exit from sniff transition mode and enter active mode.

If the slave requests an exit from sniff mode it shall send an LMP_unsniff_req PDU. When the master receives the LMP_unsniff_req PDU it shall enter sniff transition mode and then send an LMP_accepted PDU. When the slave receives the LMP_accepted PDU it shall exit from sniff mode and enter active mode. When the master receives the baseband acknowledgement for the LMP_accepted PDU it shall leave sniff transition mode and enter active mode.



Sequence 62: Slave moved from sniff mode to active mode.

4.6 LOGICAL TRANSPORTS

When a connection is first established between two devices the connection consists of the default ACL logical links: ACL-C (for LMP messages) and ACL-U (for L2CAP data.) One or more synchronous logical transports (SCO or eSCO) may then be added. A new logical transport shall not be created if it would cause all slots to be allocated to reserved slots on secondary LT_ADDRs.

4.6.1 SCO logical transport

The SCO logical transport reserves slots separated by the SCO interval, T_{sco}. The first slot reserved for the SCO logical transport is defined by T_{sco} and the SCO offset, D_{sco}. See Baseband Specification, Section 8.6.2, on page 169 for details. A device shall initiate a request for HV2 or HV3 packet type only if the other device supports it (bits 12, 13) in its features mask. A device shall initiate CVSD, μ-law or A-law coding or uncoded (transparent) data only if the other device supports the corresponding feature. To avoid problems with a wrap-around of the clock during initialization of the SCO logical transport, the timing control flags parameter is used to indicate how the first SCO slot shall be calculated. Only bit1 of the timing control flags parameter is valid. The SCO link is distinguished from all other SCO links by an SCO handle. The SCO handle zero shall not be used.

M/O	PDU	Contents
O(11)	LMP_SCO_link_req	SCO handle timing control flags D_{sco} T_{sco} SCO packet air mode
O(11)	LMP_remove_SCO_link_req	SCO handle error

Table 4.28: PDUs used for managing the SCO links.

4.6.1.1 Master initiates an SCO link

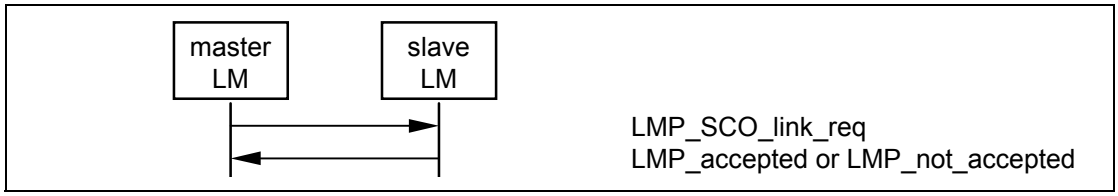
When establishing an SCO link the master sends a request, a LMP_SCO_link_req PDU, with parameters that specify the timing, packet type and coding that will be used on the SCO link. Each of the SCO packet types supports three different voice coding formats on the air-interface: μ-law log PCM, A-law log PCM and CVSD. The air coding by log PCM or CVSD may be deactivated to achieve a transparent synchronous data link at 64 kbits/s.

The slots used for the SCO links are determined by three parameters controlled by the master: T_{sco}, D_{sco} and a flag indicating how the first SCO slot is calculated. After the first slot, the SCO slots follow periodically at an interval of T_{sco}.

If the slave does not accept the SCO link, but is willing to consider another possible set of SCO parameters, it can indicate what it does not accept in the error code field of LMP_not_accepted PDU. The master may then issue a new request with modified parameters.

The SCO handle in the message shall be different from existing SCO link(s).

If the SCO packet type is HV1 the LMP_accepted shall be sent using the DM1 packet.

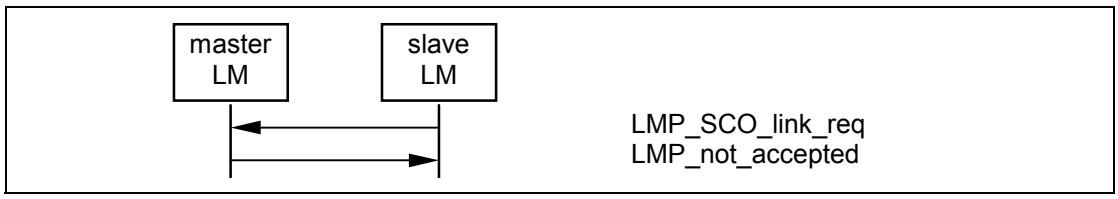


Sequence 63: Master requests an SCO link.

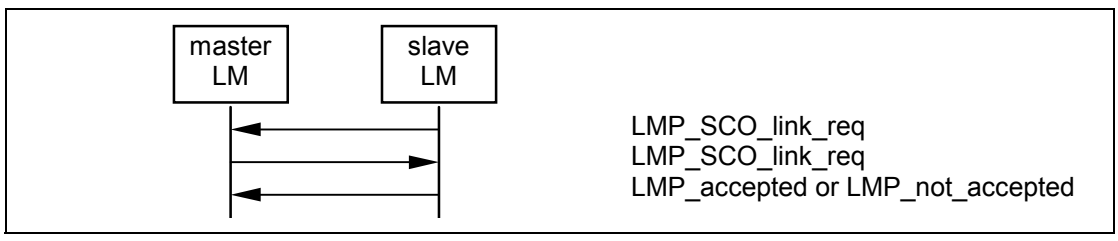
4.6.1.2 Slave initiates an SCO link

The slave may initiate the establishment of an SCO link. The slave sends an LMP_SCO_link_req PDU, but the parameters timing control flags and D_{sco} are invalid as well as the SCO handle, that shall be zero. If the master is not capable of establishing an SCO link, it replies with an LMP_not_accepted PDU. Otherwise it sends back an LMP_SCO_link_req PDU. This message includes the assigned SCO handle, D_{sco} and the timing control flags. The master should try to use the same parameters as in the slave request; if the master cannot meet that request, it is allowed to use other values. The slave shall then reply with LMP_accepted or LMP_not_accepted PDU.

If the SCO packet type is HV1 the LMP_accepted shall be sent using the DM1 packet.



Sequence 64: Master rejects slave's request for an SCO link.



Sequence 65: Master accepts slave's request for an SCO link.

4.6.1.3 Master requests change of SCO parameters

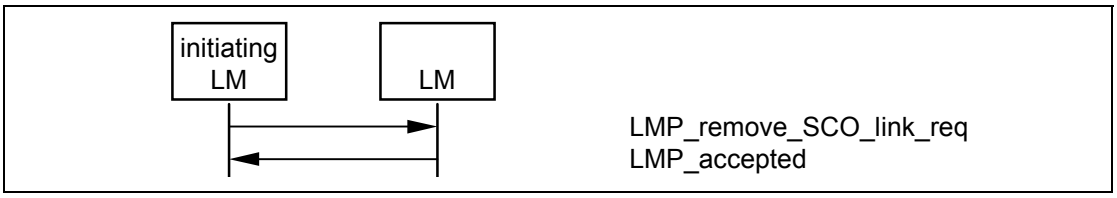
The master sends an LMP_SCO_link_req PDU, where the SCO handle is the handle of the SCO link the master wishes to change parameters for. If the slave accepts the new parameters, it replies with an LMP_accepted PDU and the SCO link will change to the new parameters. If the slave does not accept the new parameters, it shall reply with an LMP_not_accepted PDU and the SCO link is left unchanged. When the slave replies with an LMP_not_accepted PDU it shall indicate in the error code parameter what it does not accept. The master may then try to change the SCO link again with modified parameters. The sequence is the same as in Section 4.6.1.1 on page 281.

4.6.1.4 Slave requests change of SCO parameters

The slave sends an LMP_SCO_link_req PDU, where the SCO handle is the handle of the SCO link to be changed. The parameters timing control flags and D_{sco} are not valid in this PDU. If the master does not accept the new parameters it shall reply with an LMP_not_accpeted PDU and the SCO link is left unchanged. If the master accepts the new parameters it shall reply with an LMP_SCO_link_req PDU containing the same parameters as in the slave request. When receiving this message the slave replies with an LMP_not_accepted PDU if it does not accept the new parameters. The SCO link is then left unchanged. If the slave accepts the new parameters it replies with an LMP_accepted PDU and the SCO link will change to the new parameters. The sequence is the same as in Section 4.6.1.2 on page 282.

4.6.1.5 Remove an SCO link

Master or slave can remove the SCO link by sending a request including the SCO handle of the SCO link to be removed and an error code indicating why the SCO link is removed. The receiving side shall respond with an LMP_accepted PDU.



Sequence 66: SCO link removed.

4.6.2 eSCO logical transport

After an ACL link has been established, one or more extended SCO (eSCO) links can be set up to the remote device. The eSCO links are similar to SCO links using timing control flags, an interval T_{eSCO} and an offset D_{eSCO}. Only bit1 of the timing control flags parameter is valid. As opposed to SCO links, eSCO links have a configurable data rate that may be asymmetric, and can be set up to provide limited retransmissions of lost or damaged packets inside a retransmission window of size W_{eSCO}. The D_{eSCO} shall be based on CLK.

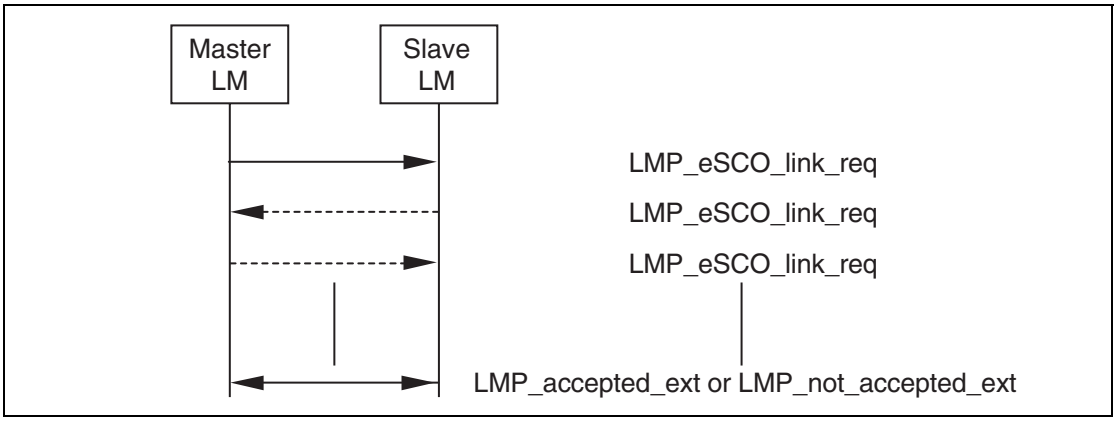
M/O	PDU	Contents
O(31)	LMP_eSCO_link_req	eSCO handle eSCO LT_ADDR timing control flags D_{eSCO} T_{eSCO} W_{eSCO} eSCO packet type M->S eSCO packet type S->M packet length M->S packet length S->M air mode negotiation state
O(31)	LMP_remove_eSCO_link_req	eSCO handle error

Table 4.29: PDUs used for managing the eSCO links

The parameters D_{eSCO}, T_{eSCO}, W_{eSCO}, eSCO packet type M->S, eSCO packet type S->M, packet length M->S, packet length S->M are henceforth referred to as the negotiable parameters.

4.6.2.1 Master initiates an eSCO link

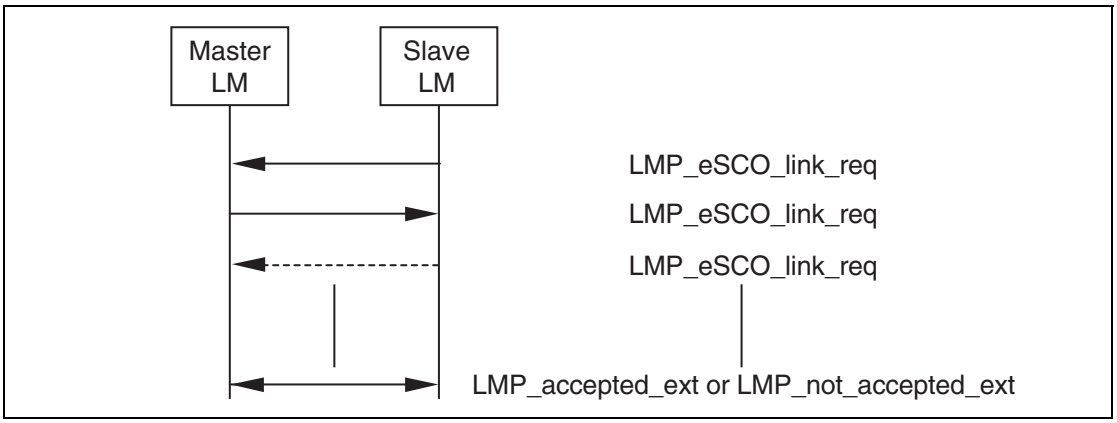
When establishing an eSCO link the master sends an LMP_eSCO_link_req PDU specifying all parameters. The slave may accept this with an LMP_accepted_ext PDU, reject it with an LMP_not_accepted_ext PDU, or respond with its own LMP_eSCO_link_req specifying alternatives for some or all parameters. The slave shall not negotiate the eSCO handle or eSCO LT_ADDR parameters. The negotiation of parameters continues until the master or slave either accepts the latest parameters with an LMP_accepted_ext PDU, or terminates the negotiation with an LMP_not_accepted_ext PDU. The negotiation shall use the procedures defined in Section 4.6.2.5 on page 287.



Sequence 67: Master requests an eSCO link.

4.6.2.2 Slave initiates an eSCO link

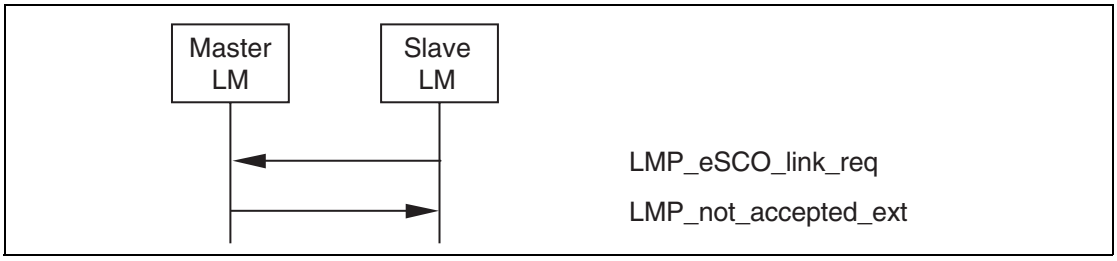
When attempting to establish an eSCO link the slave shall send an LMP_eSCO_link_req PDU specifying all parameters, with the exception of eSCO LT_ADDR and eSCO handle, which are invalid. The latter shall be set to zero. The master may respond to this with an LMP_eSCO_link_req PDU, filling in these missing parameters, and potentially changing the other requested parameters. The slave can accept this with an LMP_accepted_ext PDU, or respond with a further LMP_eSCO_link_req PDU specifying alternatives for some or all of the parameters. The negotiation of parameters continues until the master or slave either accepts the latest parameters with an LMP_accepted_ext PDU, or terminates the negotiation with an LMP_not_accepted_ext PDU.



Sequence 68: Slave requests an eSCO link.

Note that the slave should use the initialization flag appropriate to the master's Bluetooth clock. See Baseband section section 8.6.3.

The master may reject the request immediately with an LMP_not_accepted_ext PDU. The negotiation shall use the procedures defined in Section 4.6.2.5 on page 287.



Sequence 69: Master rejects slave's request for an eSCO link.

4.6.2.3 Master or slave requests change of eSCO parameters

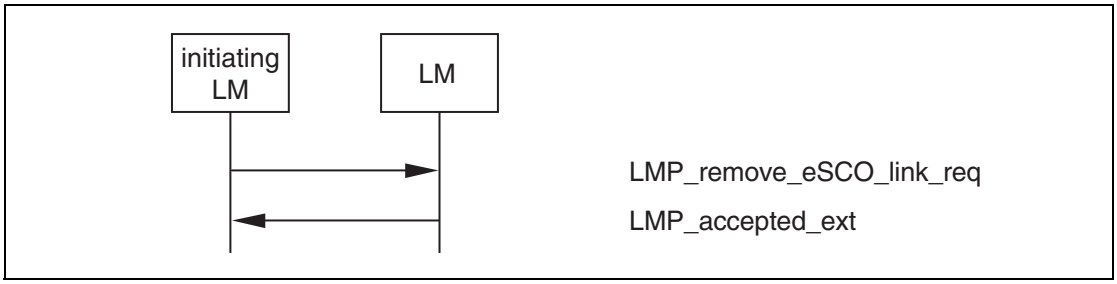
The master or slave may request a renegotiation of the eSCO parameters. The master or slave shall send an LMP_eSCO_link_req PDU with the eSCO handle of the eSCO link the device wishes to renegotiate. The remote device may accept the changed parameters immediately with LMP_accepted_ext PDU, or the negotiation may be continued with further LMP_eSCO_link_req PDUs until the master or slave accepts the latest parameters with an LMP_accepted_ext PDU or terminates the negotiation with an LMP_not_accepted_ext PDU. In the case of termination with an LMP_not_accepted_ext PDU, the eSCO link continues on the previously negotiated parameters.

The sequence is the same as in Section 4.6.2.2 on page 285.

During re-negotiation, the eSCO LT_ADDR and eSCO handle shall not be re-negotiated and shall be set to the originally negotiated values. The negotiation shall use the procedures defined in Section 4.6.2.5 on page 287.

4.6.2.4 Remove an eSCO link

Either the master or slave may remove the eSCO link by sending a request including the eSCO handle of the eSCO link to be removed and a error code indicating why the eSCO link is removed. The receiving side shall respond with an LMP_accepted_ext PDU.



Sequence 70: eSCO link removed

4.6.2.5 Rules for the LMP negotiation and renegotiation

Rule 1: the negotiation_state shall be set to 0 by the initiating LM. After the initial LMP_eSCO_link_req is sent the negotiation_state shall not be set to 0.

Rule 2: if the bandwidth (defined as 1600 times the packet length in bytes divided by T_{eSCO} in slots) for either RX or TX or the air_mode cannot be accepted the device shall send LMP_not_accepted_ext with the appropriate error code.

Rule 3: Bandwidth and air_mode are not negotiable and shall not be changed for the duration of the negotiation. Once one side has rejected the negotiation (with LMP_not_accepted_ext) a new negotiation may be started with different bandwidth and air_mode parameters.

Rule 4: if the parameters will cause a latency violation (T_{eSCO} + W_{eSCO} + reserved synchronous slots > allowed local latency) the device should propose new parameters that shall not cause a reserved slot violation or latency violation for the device that is sending the parameters. In this case the negotiation_state shall be set to 3. Otherwise the device shall send LMP_not_accepted_ext.

Rule 5: once a device has received an LMP_eSCO_link_req with the negotiation_state set to 3 (latency violation), the device shall not propose any combination of packet type, T_{eSCO}, and W_{eSCO} that will give an equal or larger latency than the combination that caused the latency violation for the other device.

Rule 6: if the parameters cause both a reserved slot violation and a latency violation the device shall set the negotiation_state to 3 (latency violation).

Rule 7: if the parameters will cause a reserved slot violation the device should propose new parameters that shall not cause a reserved slot violation. In this case the negotiation_state shall be set to 2. Otherwise the device shall send LMP_not_accepted_ext.

Rule 8: If the requested parameters are not supported the device should propose a setting that is supported, and set the negotiation_state to 4. If it is not possible to find such a parameter set, the device shall send LMP_not_accepted_ext.

Rule 9: when proposing new parameters for reasons other than a latency violation, reserved slot violation, or configuration not supported, the negotiation_state shall be set to 1.

4.6.2.6 Negotiation state definitions

Reserved Slot Violation: a reserved slot violation is when the receiving LM cannot setup the requested eSCO logical transport because the eSCO reserved slots would overlap with other regularly scheduled slots (e.g. other synchronous reserved slots, sniff instants, or park beacons).

Latency Violation: a latency violation is when the receiving LM cannot setup the requested eSCO logical transport because the latency (W_{eSCO}+T_{eSCO} + reserved synchronous slots) is greater than the maximum allowed latency.

Configuration not supported: The combination of parameters requested is not inside the supported range for the device.

4.7 TEST MODE

LMP has PDUs to support different test modes used for certification and compliance testing of the Bluetooth radio and baseband. See "Test Methodology" on page 231[vol. 4] for a detailed description of these test modes.

4.7.1 Activation and deactivation of test mode

The activation may be carried out locally (via a HW or SW interface), or using the air interface.

- For activation over the air interface, entering the test mode shall be locally enabled for security and type approval reasons. The implementation of this local enabling is not subject to standardization.

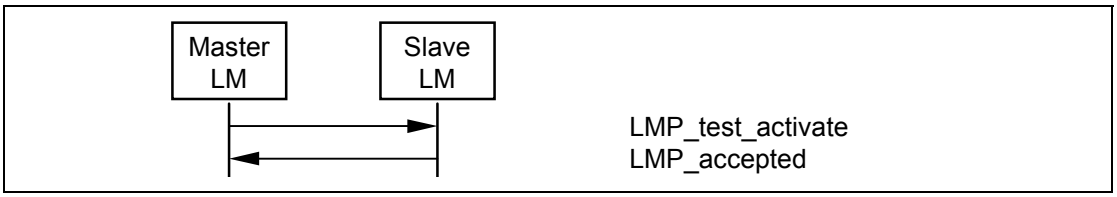
  The tester sends an LMP command that shall force the DUT to enter test mode. The DUT shall terminate all normal operation before entering the test mode.

  The DUT shall return an LMP_Accepted on reception of an activation command. LMP_Not_Accepted shall be returned if the DUT is not locally enabled.

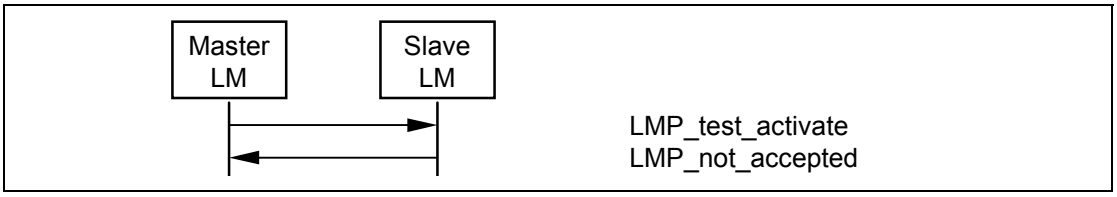
- If the activation is performed locally using a HW or SW interface, the DUT shall terminate all normal operation before entering the test mode.

  Until a connection to the tester exists, the device shall perform page scan and inquiry scan. Extended scan activity is recommended.

The test mode is activated by sending an LMP_test_activate PDU to the device under test (DUT). The DUT is always the slave. The lm shall be able to receive this message anytime. If entering test mode is locally enabled in the DUT it shall respond with an LMP_accepted PDU and test mode is entered. Otherwise the DUT responds with an LMP_not_accepted PDU and the DUT remains in normal operation. The error code in the LMP_not_accepted PDU shall be *PDU not allowed*.



Sequence 71: Activation of test mode successful.



Sequence 72: Activation of test mode fails. Slave is not allowed to enter test mode.

The test mode can be deactivated in two ways. Sending an LMP_test_control PDU with the test scenario set to "exit test mode" exits the test mode and the slave returns to normal operation still connected to the master. Sending an LMP_detach PDU to the DUT ends the test mode and the connection.

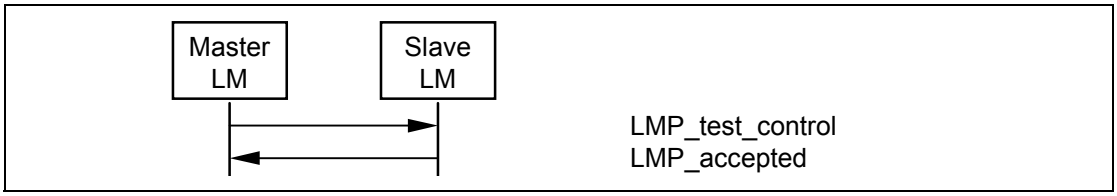
4.7.2 Control of test mode

Control and configuration is performed using special LMP commands (see Section 4.7.3 on page 291). These commands shall be rejected if the Bluetooth device is not in test mode. In this case, an LMP_not_accepted shall be returned. The DUT shall return an LMP_accepted on reception of a control command when in test mode.

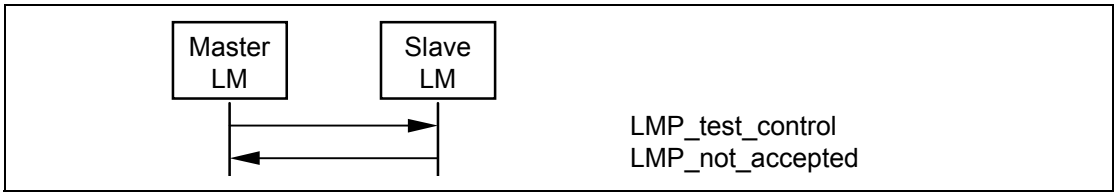
A Bluetooth device in test mode shall ignore all LMP commands not related to control of the test mode. LMP commands dealing with power control and the request for LMP features (LMP_features_req), and adaptive frequency hopping (LMP_set_AFH, LMP_channel_classification_req and LMP_channel_classification) are allowed in test mode; the normal procedures are also used to test the adaptive power control.

The DUT shall leave the test mode when an LMP_Detach command is received or an LMP_test_control command is received with test scenario set to ’exit test mode’.

When the DUT has entered test mode, the PDU LMP_test_control PDU can be sent to the DUT to start a specific test. This PDU is acknowledged with an LMP_accepted PDU. If a device that is not in test mode receives an LMP_test_control PDU it responds with an LMP_not_accepted PDU, where the error code shall be *PDU not allowed*.



Sequence 73: Control of test mode successful.



Sequence 74: Control of test mode rejected since slave is not in test mode.

4.7.3 Summary of test mode PDUs

Table 4.30 lists all LMP messages used for test mode. To ensure that the contents of LMP_test_control PDU are suitably whitened (important when sent in transmitter mode), all parameters listed in Table 4.31 on page 291 are XORed with 0x55 before being sent.

LMP PDU	PDU number	Possible Direction	Contents	Position in Payload
LMP_test_activate	56	m → s		
LMP_test_control	57	m → s	test scenario hopping mode TX frequency RX frequency power control mode poll period packet type length of test data	2 3 4 5 6 7 8 9-10
LMP_detach	7	m → s		
LMP_accepted	3	m ← s		
LMP_not_accepted	4	m ← s		

Table 4.30: LMP messages used for Test Mode

Name	Length (bytes)	Type	Unit	Detailed
Test scenario	1	u_int8		0 Pause Test Mode 1 Transmitter test – 0 pattern 2 Transmitter test – 1 pattern 3 Transmitter test – 1010 pattern 4 Pseudorandom bit sequence 5 Closed Loop Back – ACL packets 6 Closed Loop Back – Synchronous packets 7 ACL Packets without whitening 8 Synchronous Packets without whitening 9 Transmitter test – 1111 0000 pattern 10–254 reserved 255 Exit Test Mode The value is XORed with 0x55.

Table 4.31: Parameters used in LMP_Test_Control PDU

Name	Length (bytes)	Type	Unit	Detailed
Hopping mode	1	u_int8		0 RX/TX on single frequency 1 Normal hopping 2 Reserved 3 Reserved 4 Reserved 5–255 reserved The value is XORed with 0x55.
TX frequency (for DUT)	1	u_int8		f = [2402 + k] MHz The value is XORed with 0x55.
RX frequency (for DUT)	1	u_int8		f = [2402 + k] MHz The value is XORed with 0x55.
Power control mode	1	u_int8		0 fixed TX output power 1 adaptive power control The value is XORed with 0x55.
Poll period	1	u_int8	1.25 ms	The value is XORed with 0x55.
Packet type	1	u_int8		Bits 3-0 numbering as in packet header, see Baseband Specification Bits 7-4 0: ACL/SCO 1: eSCO 2: Enhanced Data Rate ACL 3: Enhanced Data Rate eSCO 4-15: reserved Other values are reserved The value is XORed with 0x55.
length of test sequence (=length of user data in Baseband Specification)	2	u_int16	1 byte	unsigned binary number The value is XORed with 0x5555.

Table 4.31: Parameters used in LMP_Test_Control PDU

The control PDU is used for both transmitter and loop back tests. The following restrictions apply for the parameter settings:

Parameter	**Restrictions Transmitter Test**	**Restrictions Loopback Test**
TX frequency	$0 \leq k \leq 93$	$0 \leq k \leq 78$
RX frequency	same as TX frequency	$0 \leq k \leq 78$
Poll period		not applicable (set to 0)
Length of test sequence	depends on packet type: [see table 6.9 and 6.10 in the Baseband specification]	For ACL and SCO packets: not applicable (set to 0) For eSCO packets: [see table 6.10 in the Baseband specification]

Table 4.32: Restrictions for Parameters used in LMP_Test_Control PDU

5 SUMMARY

5.1 PDU SUMMARY

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
Escape 1	variable	124	DM1	m ↔ s	extended op code	2
					variable	3-?
Escape 2	variable	125	DM1	m ↔ s	extended op code	2
					variable	3-?
Escape 3	variable	126	DM1	m ↔ s	extended op code	2
					variable	3-?
Escape 4	variable	127	DM1	m ↔ s	extended op code	2
					variable	3-?
LMP_accepted	2	3	DM1/ DV	m ↔ s	op code	2
LMP_accepted_ext	4	127/ 01	DM1	m ↔ s	escape op code	3
					extended op code	4
LMP_au_rand	17	11	DM1	m ↔ s	random number	2-17
LMP_auto_rate	1	35	DM1/ DV	m ↔ s	-	
LMP_channel _classification_req	7	127/ 16	DM1	m → s	AFH_reporting_mode	3
					AFH_min_interval	4-5
					AFH_max_interval	6-7
LMP_channel _classification	12	127/ 17	DM1	m ← s	AFH_channel_ classification	3 – 12
LMP_clkoffset_req	1	5	DM1/ DV	m → s	-	
LMP_clkoffset_res	3	6	DM1/ DV	m ← s	clock offset	2–3
LMP_comb_key	17	9	DM1	m ↔ s	random number	2-17
LMP_decr_power_req	2	32	DM1/ DV	m ↔ s	for future use	2
LMP_detach	2	7	DM1/ DV	m ↔ s	error code	2

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_encryption_key_size_mask_req	1	58	DM1	m → s		
LMP_encryption_key_size_mask_res	3	59	DM1	m ← s	key size mask	2-3
LMP_encryption_key_size_req	2	16	DM1/ DV	m ↔ s	key size	2
LMP_encryption_mode_req	2	15	DM1/ DV	m ↔ s	encryption mode	2
LMP_eSCO_link_req	16	127/ 12	DM1	m↔s	eSCO handle	3
					eSCO LT_ADDR	4
					timing control flags	5
					D_{eSCO}	6
					T_{eSCO}	7
					W_{eSCO}	8
					eSCO packet type M->S	9
					eSCO packet type S->M	10
					packet length M->S	11-12
					packet length S->M	13-14
					air mode	15
					negotiation state	16
LMP_features_req	9	39	DM1/ DV	m ↔ s	features	2-9
LMP_features_req_ext	12	127/ 03	DM1	m ↔ s	features page	3
					max supported page	4
					extended features	5-12
LMP_features_res	9	40	DM1/ DV	m ↔ s	features	2-9
LMP_features_res_ext	12	127/ 04	DM1	m ↔ s	features page	3
					max supported page	4
					extended features	5-12
LMP_host_connection_req	1	51	DM1/ DV	m ↔ s	-	

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_hold	7	20	DM1/ DV	m ↔ s	hold time	2-3
					hold instant	4-7
LMP_hold_req	7	21	DM1/ DV	m ↔ s	hold time	2-3
					hold instant	4-7
LMP_incr_power_req	2	31	DM1/ DV	m ↔ s	for future use	2
LMP_in_rand	17	8	DM1	m ↔ s	random number	2-17
LMP_max_power	1	33	DM1/ DV	m ↔ s	-	
LMP_max_slot	2	45	DM1/ DV	m ↔ s	max slots	2
LMP_max_slot_req	2	46	DM1/ DV	m ↔ s	max slots	2
LMP_min_power	1	34	DM1/ DV	m ↔ s	-	
LMP_modify_beacon	11 or 13	28	DM1	m → s	timing control flags	2
					D_B	3-4
					T_B	5-6
					N_B	7
					Δ_B	8
					D_{access}	9
					T_{access}	10
					$N_{acc\text{-}slots}$	11
					N_{poll}	12
					M_{access}	13:0-3
					access scheme	13:4-7
LMP_name_req	2	1	DM1/ DV	m ↔ s	name offset	2
LMP_name_res	17	2	DM1	m ↔ s	name offset	2
					name length	3
					name fragment	4-17

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_not_accepted	3	4	DM1/ DV	m ↔ s	op code	2
					error code	3
LMP_not_accepted_ext	5	127/ 02	DM1	m ↔ s	escape op code	3
					extended op code	4
					error code	5
LMP_packet_type_table_ req	3	127/ 11	DM1	m ↔ s	packet type table	3
LMP_page_mode_req	3	53	DM1/ DV	m ↔ s	paging scheme	2
					paging scheme settings	3
LMP_page_scan_mode _req	3	54	DM1/ DV	m ↔ s	paging scheme	2
					paging scheme settings	3
LMP_park_req	17	25	DM1	m ↔ s	timing control flags	2
					D_B	3-4
					T_B	5-6
					N_B	7
					Δ_B	8
					PM_ADDR	9
					AR_ADDR	10
					N_{Bsleep}	11
					D_{Bsleep}	12
					D_{access}	13
					T_{access}	14
					$N_{acc-slots}$	15
					N_{poll}	16
					M_{access}	17:0-3
					access scheme	17:4-7
LMP_preferred_rate	2	36	DM1/ DV	m ↔ s	data rate	2
LMP_quality_of_service	4	41	DM1/ DV	m → s	poll interval	2-3
					N_{BC}	4

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_quality_of_service_req	4	42	DM1/ DV	m ↔ s	poll interval	2-3
					N_{BC}	4
LMP_remove_eSCO_link_req [] see Note4 on page 302	4	127/ 13	DM1	m↔s	eSCO handle	3
					error code	4
LMP_remove_SCO_link_req	3	44	DM1/ DV	m ↔ s	SCO handle	2
					error code	3
LMP_SCO_link_req	7	43	DM1/ DV	m ↔ s	SCO handle	2
					timing control flags	3
					D_{sco}	4
					T_{sco}	5
					SCO packet	6
					air mode	7
LMP_set_AFH	16	60	DM1	m → s	AFH_instant	2-5
					AFH_mode	6
					AFH_channel_map	7-16
LMP_set_broadcast_scan_window	4 or 6	27	DM1	m → s	timing control flags	2
					D_B	3-4
					broadcast scan window	5-6
LMP_setup_complete	1	49	DM1	m ↔ s	-	
LMP_slot_offset	9	52	DM1/ DV	m ↔ s	slot offset	2-3
					BD_ADDR	4-9
LMP_sniff_req	10	23	DM1	m ↔ s	timing control flags	2
					D_{sniff}	3-4
					T_{sniff}	5-6
					sniff attempt	7-8
					sniff timeout	9-10
LMP_sres	5	12	DM1/ DV	m ↔ s	authentication response	2-5
LMP_start_encryption_req	17	17	DM1	m → s	random number	2-17
LMP_stop_encryption_req	1	18	DM1/ DV	m → s	-	

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_supervision_timeout	3	55	DM1/ DV	m →s	supervision timeout	2-3
LMP_switch_req	5	19	DM1/ DV	m ↔ s	switch instant	2-5
LMP_temp_rand	17	13	DM1	m → s	random number	2-17
LMP_temp_key	17	14	DM1	m → s	key	2-17
LMP_test_activate	1	56	DM1/ DV	m → s	-	
LMP_test_control	10	57	DM1	m → s	test scenario	2
					hopping mode	3
					TX frequency	4
					RX frequency	5
					power control mode	6
					poll period	7
					packet type	8
					length of test data	9-10
LMP_timing_accuracy_req	1	47	DM1/ DV	m ↔ s	-	
LMP_timing_accuracy_res	3	48	DM1/ DV	m ↔ s	drift	2
					jitter	3
LMP_unit_key	17	10	DM1	m ↔ s	key	2-17
LMP_unpark_BD_ADDR _req	variable	29	DM1	m → s	timing control flags	2
					D_B	3-4
					LT_ADDR 1st unpark	5:0-2
					LT_ADDR 2nd unpark	5:4-6
					BD_ADDR 1st unpark	6-11
					BD_ADDR 2nd unpark	12-17

Table 5.1: Coding of the different LM PDUs.

LMP PDU	Length (bytes)	op code	Packet type	Possible direction	Contents	Position in payload
LMP_unpark_PM_ADDR_req	variable	30	DM1	m → s	timing control flags	2
					D_B	3-4
					LT_ADDR 1st unpark	5:0-3
					LT_ADDR 2nd unpark	5:4-7
					PM_ADDR 1st unpark	6
					PM_ADDR 2nd unpark	7
					LT_ADDR 3rd unpark	8:0-3
					LT_ADDR 4th unpark	8:4-7
					PM_ADDR 3rd unpark	9
					PM_ADDR 4th unpark	10
					LT_ADDR 5th unpark	11:0-3
					LT_ADDR 6th unpark	11:4-7
					PM_ADDR 6th unpark	12
					PM_ADDR 6th unpark	13
					LT_ADDR 7th unpark	14:0-3
					PM_ADDR 7th unpark	15
LMP_unsniff_req	1	24	DM1/ DV	m ↔ s	-	
LMP_use_semi_ permanent_key	1	50	DM1/ DV	m → s	-	
LMP_version_req	6	37	DM1/ DV	m ↔ s	VersNr	2
					CompId	3-4
					SubVersNr	5-6
LMP_version_res	6	38	DM1/ DV	m ↔ s	VersNr	2
					CompId	3-4
					SubVersNr	5-6

Table 5.1: Coding of the different LM PDUs.

Note1: For LMP_set_broadcast_scan_window, LMP_modify_beacon, LMP_unpark_BD_ADDR_req and LMP_unpark_PM_ADDR_req the parameter D_B is optional. This parameter is only present if bit0 of *timing control flags* is 1.

If the parameter is not included, the position in payload for all parameters following D_B are decreased by 2.

Note2: For LMP_unpark_BD_ADDR the LT_ADDR and the BD_ADDR of the 2^{nd} unparked slave are optional. If only one slave is unparked *LT_ADDR 2^{nd} unpark* shall be zero and *BD_ADDR 2^{nd} unpark* is left out.

Note3: For LMP_unpark_PM_ADDR the LT_ADDR and the PM_ADDR of the 2^{nd} – 7^{th} unparked slaves are optional. If N slaves are unparked, the fields up to and including the N^{th} unparked slave are present. If N is odd, the *LT_ADDR $(N+1)^{th}$ unpark* shall be zero. The length of the message is x + 3N/2 if N is even and x + 3(N+1)/2 -1 if N is odd, where x = 2 or 4 depending on if the D_B is incluDed Or Not (See Note1).

Note4: Parameters coincide with their namesakes in LMP_<remove>_SCO_link_req apart from the following:

1. eSCO_LT_ADDR - the eSCO connection will be active on an additional LT_ADDR that needs to be defined. The master is allowed to re-assign an active eSCO link to a different LT_ADDR.
2. D_{eSCO}, T_{eSCO} - as per LMP_SCO_link_req but with a greater flexibility in values (e.g. no longer fixed with respect to HV1, HV2, and HV3 packet choice).
3. W_{eSCO} - the eSCO retransmission window size (in slots)
4. packet type and packet length may be prescribed differently in Master-to-Slave or Slave-to-Master directions for asynchronous eSCO links
5. packet length (in bytes) - eSCO packet types no longer have fixed length
6. negotiation state – this is used to better enable the negotiation of the negotiable parameters: D_{eSCO}, T_{eSCO}, W_{eSCO}, eSCO packet type M->S, eSCO packet type S->M, packet length M->S, packet length S->M. When responding to an eSCO link request with a new suggestion for these parameters, this flag may be set to 1 to indicate that the last received negotiable parameters are possible, but the new parameters specified in the response eSCO link request would be preferable, to 2 to indicate that the last received negotiable parameters are not possible as they cause a reserved slot violation or to 3 to indicate that the last received negotiable parameters would cause a latency violation. The flag shall be set to zero in the initiating LMP_eSCO_link_req.

5.2 PARAMETER DEFINITIONS

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
access scheme	1	u_int4		0: polling technique 1-15: Reserved	
AFH_channel_classification	10	multiple bytes	-	This parameter contains 40 2-bit fields. The n^{th} (numbering from 0) such field defines the classification of channels *2n* and *2n+1*, other than the 39^{th} field which just contains the classification of channel 78. Each field interpreted as an integer whose values indicate: 0 = unknown 1 = good 2 = reserved 3 = bad	
AFH_channel_map	10	multiple bytes	-	If *AFH_mode* is *AFH_enabled*, this parameter contains 79 1-bit fields, otherwise the contents are reserved. The n^{th} (numbering from 0) such field (in the range 0 to 78) contains the value for channel *n*. Bit 79 is reserved (set to 0 when transmitted and ignored when received) The 1-bit field is interpreted as follows: 0: channel *n* is unused 1: channel *n* is used	
AFH_instant	4	u_int32	slots	Bits 27:1 of the Bluetooth master clock value at the time of switching hop sequences. Must be even.	
AFH_max_interval	2	u_int16	slots	Range is 0x0640 to 0xBB80 slots (1 to 30s)	
AFH_min_interval	2	u_int16	slots	Range is 0x0640 to 0xBB80 slots (1 to 30s)	

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
AFH_mode	1	u_int8	-	0: AFH_disabled 1: AFH_enabled 2-255: Reserved	
AFH_reporting_mode	1	u_int8	-	0: AFH_reporting_disabled 1: AFH_reporting_enabled 2-255: reserved	
air mode	1	u_int8		0: μ-law log 1: A-law log 2: CVSD 3: transparent data 4-255: Reserved	See Table 5.3 on page 310
AR_ADDR	1	u_int8			
authentication response	4	multiple bytes			
BD_ADDR	6	multiple bytes		BD_ADDR of the sending device	
broadcast scan window	2	u_int16	slots		
clock offset	2	u_int16	1.25ms	$(CLKN_{16-2}$ slave - $CLKN_{16-2}$ master) mod 2^{15} MSbit of second byte not used.	
CompId	2	u_int16		see Bluetooth Assigned Numbers (https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers)	
D_{access}	1	u_int8	slots		

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
data rate	1	u_int8		When in Basic Rate mode: bit0 = 0: use FEC bit0 = 1: do not use FEC bit1-2=0: No packet-size preference available bit1-2=1: use 1-slot packets bit1-2=2: use 3-slot packets bit1-2=3: use 5-slot packets When in Enhanced Data Rate mode: bit3-4=0: use DM1 packets bit3-4=1: use 2 Mbps packets bit3-4=2: use 3 Mbps packets bit3-4=3: reserved bit5-6=0: No packet-size preference available bit5-6=1: use 1-slot packets bit5-6=2: use 3-slot packets bit5-6=3: use 5-slot packets bit7: Reserved - shall be zero	
D_B	2	u_int16	slots		
Δ_B	1	u_int8	slots		
D_{Bsleep}	1	u_int8			
D_{eSCO}	1	u_int8	slots	Valid range is 0 - 254 slots	See Table 5.3 on page 310
drift	1	u_int8	ppm		
D_{sco}	1	u_int8	slots	Only even values are valid[1]	0 to T_{sco} -2
D_{sniff}	2	u_int16	slots	Only even values are valid[1]	0 to (T_{sniff} - 2)

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
encryption mode	1	u_int8		0: no encryption 1: encryption 2: encryption 3 -255: Reserved	
error code	1	u_int8		See Part D on page 319	
escape op code	1	u_int8		Identifies which escape op code is being acknowl-edged: range 124-127	
eSCO handle	1	u_int8			
eSCO LT_ADDR	1	u_int8		Logical transport address for the eSCO logical trans-port. The range is extended to 8 bits com-pared with the normal LT_ADDR field: range 0-7.	0 - 7
eSCO packet type	1	u_int8		0x00: NULL/POLL 0x07: EV3 0x0C: EV4 0x0D: EV5 0x26: 2-EV3 0x2C: 2-EV5 0x37: 3-EV3 0x3D: 3-EV5 Other values are reserved	If the value is 0x00 the POLL packet shall be used by the master, the NULL packet shall be used by the slave. See Table 5.3 on page 310
extended features	8	multiple bytes		One page of extended fea-tures	
extended op code	1	u_int8		Which extended op code is being acknowledged	
features	8	multiple bytes		See Table 3.2 on page 230	
features page	1	u_int8		Identifies which page of extended features is being requested. 0 means standard features 1-255 other feature pages	
hold instant	4	u_int32	slots	Bits 27:1 of the master Bluetooth clock value	

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
hold time	2	u_int16	slots	Only even values are valid[1]	0x0014 to 0x8000; shall not exceed (supervisionTO * 0.999)
jitter	1	u_int8	µs		
key	16	multiple bytes			
key size	1	u_int8	byte		
key size mask	2	u_int16		Bit mask of supported broadcast encryption key sizes: least significant bit is support for length 1, and so on. The bit shall be one if the key size is supported.	
LT_ADDR	1	u_int4			
M_{access}	1	u_int4		number of access windows	
max slots	1	u_int8	slots		
max supported page	1	u_int8		Highest page of extended features which contains a non-zero bit for the originating device. Range 0-255	
$N_{acc\text{-}slots}$	1	u_int8	slots		
name fragment	14	multiple bytes		UTF-8 characters.	
name length	1	u_int8	bytes		
name offset	1	u_int8	bytes		
N_B	1	u_int8			
N_{BC}	1	u_int8			
N_{Bsleep}	1	u_int8			

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
negotiation state	1	u_int8		0: Initiate negotiation 1: the latest received set of negotiable parameters were possible but these parameters are preferred. 2: the latest received set of negotiable parameters would cause a reserved slot violation. 3: the latest received set of negotiable parameters would cause a latency violation. 4: the latest received set of of negotiable parameters are not supported. Other values are reserved	
N_{poll}	1	u_int8			
op code	1	u_int8			
packet length	2	uint16	bytes	Length of the eSCO payload 0 for POLL/NULL 1-30 for EV3 1-120 for EV4 1-180 for EV5 1-60 for 2-EV3 1-360 for 2-EV5 1-90 for 3-EV3 1-540 for 3-EV5 Other values are invalid	See Table 5.3 on page 310
packet type table	1	u_int8		0: 1Mbps only 1: 2/3 Mbps 2-255: reserved	0-1
paging scheme	1	u_int8		0: mandatory scheme 1-255: Reserved	
paging scheme settings	1	u_int8		For mandatory scheme: 0: R0 1: R1 2: R2 3-255: Reserved	
PM_ADDR	1	u_int8			

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
poll interval	2	u_int16	slots	Only even values are valid[1]	0x0006 to 0x1000
random number	16	multiple bytes			
reserved(n)	n	u_int8		Reserved for future use – must be 0 when transmitted, ignore value when received	
SCO handle	1	u_int8			
SCO packet	1	u_int8		0: HV1 1: HV2 2: HV3 3-255: Reserved	
slot offset	2	u_int16	μs	0 ≤ slot offset < 1250	
sniff attempt	2	u_int16	received slots	Number of receive slots	1 to $T_{sniff}/2$
sniff timeout	2	u_int16	received slots	Number of receive slots	0 to 0x0028
SubVersNr	2	u_int16		Defined by each company	
supervision timeout	2	u_int16	slots	0 means an infinite time-out	0 and 0x0190 to 0xFFFF
switch instant	4	u_int32	slots	Bits 27:1 of the master Bluetooth clock value	
T_{access}	1	u_int8	slots		
T_B	2	u_int16	slots		
T_{eSCO}	1	u_int8	slots	Valid range is 4 – 254 slots	See Table 5.3 on page 310
timing control flags	1	u_int8		bit0 = 0: no timing change bit0 = 1: timing change bit1 = 0: use initialization 1 bit1 = 1: use initialization 2 bit2 = 0: access window bit2 = 1: no access window bit3-7: Reserved	
T_{sco}	1	u_int8	slots	Only even values are valid[1]	2 to 6

Table 5.2: Parameters in LM PDUs.

Name	Length (bytes)	Type	Unit	Detailed	Mandatory range
T_{sniff}	2	u_int16	slots	Only even values are valid[1]	0x0006 to 0x0540; shall not exceed (supervisionTO * 0.999)
VersNr	1	u_int8		See Bluetooth Assigned Numbers, (http://www.bluetooth.org/assigned-numbers.htm)	
W_{eSCO}	1	u_int8	slots	Number of slots in the retransmission window Valid range is 0 – 254 slots	See Table 5.3 on page 310

Table 5.2: Parameters in LM PDUs.

1. If a device receives an LMP PDU with an odd value in this parameter field the PDU should be rejected with an error code of *invalid LMP parameters*.

	Single Slot Packets	3-Slot Packets
D_{eSCO}	0 to **T_{eSCO}**-2 (even)	0 to **T_{eSCO}**-2 (even)
T_{eSCO}	EV3: 6 2-EV3: 6-12 (even) 3-EV3: 6-18 (even)	EV4: 16 EV5: 16 2-EV5: 16 3-EV5: 16
W_{eSCO}	0, 2, and 4	0 and 6
packet length M->S	10*T_{eSCO} /2	10*T_{eSCO} /2
packet length S->M	10*T_{eSCO} /2	10*T_{eSCO} /2
air mode	At least one of A-law, mu-law, CVSD, transparent	transparent

Table 5.3: Mandatory parameter ranges for eSCO packet types

5.3 DEFAULT VALUES

Devices shall use these values before anything else has been negotiated:

Parameter	Value
AFH_mode	AFH_disabled
AFH_reporting_mode	AFH_reporting_disabled
drift	250
jitter	10
max slots	1
poll interval	40

Table 5.4: Default values.

6 LIST OF FIGURES

7 LIST OF TABLES

Core System Package [Controller volume]
Part D

ERROR CODES

CONTENTS

1 OVERVIEW OF ERROR CODES

This document lists the various possible error codes. When a command fails, or an LMP message needs to indicate a failure, error codes are used to indicate the reason for the error. Error codes have a size of one octet.

1.1 USAGE DESCRIPTIONS

The purpose of this section is to give descriptions of how the error codes should be used. It is beyond the scope of this document to give detailed descriptions of all situations where error codes can be used, especially as this may be implementation dependent.

1.2 HCI COMMAND ERRORS

If an HCI Command that should generate an HCI_Command_Complete event generates an error then this error shall be reported in the HCI_Command_Complete event.

If an HCI Command that sent an HCI_Command_Status with the error code ‘Success’ to the host before processing may find an error during execution then the error may be reported in the normal completion command for the original command or in an HCI_Command_Status event.

Some HCI Commands may generate errors that need to be reported to be host, but there is insufficient information to determine how the command would normally be processed. In this case, two events can be used to indicate this to the host, the HCI_Command_Complete event and HCI_Command_Status events. Which of the two events is used is implementation-dependent.

1.3 LIST OF ERROR CODES

The error code of 0x00 means Success. The possible range of failure error codes is 0x01-0xFF. Section 2 provides an error code usage description for each failure error code.

Values marked as "Reserved for Future Use", can be used in future versions of the specification. A host shall consider any error code that it does not explicitly understand equivalent to the "Unspecified Error (0x1F)."

Error Code	Name
0x00	Success
0x01	Unknown HCI Command
0x02	Unknown Connection Identifier
0x03	Hardware Failure
0x04	Page Timeout
0x05	Authentication Failure
0x06	PIN or Key Missing
0x07	Memory Capacity Exceeded
0x08	Connection Timeout
0x09	Connection Limit Exceeded
0x0A	Synchronous Connection Limit To A Device Exceeded
0x0B	ACL Connection Already Exists
0x0C	Command Disallowed
0x0D	Connection Rejected due to Limited Resources
0x0E	Connection Rejected Due To Security Reasons
0x0F	Connection Rejected due to Unacceptable BD_ADDR
0x10	Connection Accept Timeout Exceeded
0x11	Unsupported Feature or Parameter Value
0x12	Invalid HCI Command Parameters
0x13	Remote User Terminated Connection
0x14	Remote Device Terminated Connection due to Low Resources
0x15	Remote Device Terminated Connection due to Power Off
0x16	Connection Terminated By Local Host
0x17	Repeated Attempts
0x18	Pairing Not Allowed

Table 1.1: List of Possible Error Codes

Error Code	Name
0x19	Unknown LMP PDU
0x1A	Unsupported Remote Feature / Unsupported LMP Feature
0x1B	SCO Offset Rejected
0x1C	SCO Interval Rejected
0x1D	SCO Air Mode Rejected
0x1E	Invalid LMP Parameters
0x1F	Unspecified Error
0x20	Unsupported LMP Parameter Value
0x21	Role Change Not Allowed
0x22	LMP Response Timeout
0x23	LMP Error Transaction Collision
0x24	LMP PDU Not Allowed
0x25	Encryption Mode Not Acceptable
0x26	Link Key Can Not be Changed
0x27	Requested QoS Not Supported
0x28	Instant Passed
0x29	Pairing With Unit Key Not Supported
0x2A	Different Transaction Collision
0x2B	Reserved
0x2C	QoS Unacceptable Parameter
0x2D	QoS Rejected
0x2E	Channel Classification Not Supported
0x2F	Insufficient Security
0x30	Parameter Out Of Mandatory Range
0x31	Reserved
0x32	Role Switch Pending
0x33	Reserved
0x34	Reserved Slot Violation
0x35	Role Switch Failed

Table 1.1: List of Possible Error Codes

2 ERROR CODE DESCRIPTIONS

2.1 UNKNOWN HCI COMMAND (0X01)

The Unknown HCI Command error code indicates that the controller does not understand the HCI Command Packet OpCode that the host sent. The OpCode given might not correspond to any of the OpCodes specified in this document, or any vendor-specific OpCodes, or the command may not have been implemented.

2.2 UNKNOWN CONNECTION IDENTIFIER (0X02)

The Unknown Connection Identifier error code indicates that a command was sent from the host that should identify a connection, but that connection does not exist.

2.3 HARDWARE FAILURE (0X03)

The Hardware Failure error code indicates to the host that something in the controller has failed in a manner that cannot be described with any other error code. The meaning implied with this error code is implementation dependent.

2.4 PAGE TIMEOUT (0X04)

The Page Timeout error code indicates that a page timed out because of the Page Timeout configuration parameter. This error code may occur only with the HCI_Remote_Name_Request and HCI_Create_Connection commands.

2.5 AUTHENTICATION FAILURE (0X05)

The Authentication Failure error code indicates that pairing or authentication failed due to incorrect results in the pairing or authentication procedure. This could be due to an incorrect PIN or Link Key.

2.6 PIN OR KEY MISSING (0X06)

The PIN or Key Missing error code is used when pairing failed because of a missing PIN, or authentication failed because of a missing Key.

2.7 MEMORY CAPACITY EXCEEDED (0X07)

The Memory Capacity Exceeded error code indicates to the host that the controller has run out of memory to store new parameters.

2.8 CONNECTION TIMEOUT (0X08)

The Connection Timeout error code indicates that the link supervision timeout has expired for a given connection.

2.9 CONNECTION LIMIT EXCEEDED (0X09)

The Connection Limit Exceeded error code indicates that an attempt to create another connection failed because the controller is already at its limit of the number of connections it can support. The number of connections a device can support is implementation dependent.

2.10 SYNCHRONOUS CONNECTION LIMIT TO A DEVICE EXCEEDED (0X0A)

The Synchronous Connection Limit to a Device Exceeded error code indicates that the controller has reached the limit to the number of synchronous connections that can be achieved to a device. The number of synchronous connections a device can support is implementation dependent.

2.11 ACL CONNECTION ALREADY EXISTS (0X0B)

The ACL Connection Already Exists error code indicates that an attempt to create a new ACL Connection to a device when there is already a connection to this device.

2.12 COMMAND DISALLOWED (0X0C)

The Command Disallowed error code indicates that the command requested cannot be executed because the controller is in a state where it cannot process this command at this time. This error shall not be used for command OpCodes where the error code Unknown HCI Command is valid.

2.13 CONNECTION REJECTED DUE TO LIMITED RESOURCES (0X0D)

The Connection Rejected Due To Limited Resources error code indicates that an incoming connection was rejected due to limited resources.

2.14 CONNECTION REJECTED DUE TO SECURITY REASONS (0X0E)

The Connection Rejected Due To Security Reasons error code indicates that a connection was rejected due to security requirements not being fulfilled, like authentication or pairing.

2.15 CONNECTION REJECTED DUE TO UNACCEPTABLE BD_ADDR (0X0F)

The Connection Rejected due to Unacceptable BD_ADDR error code indicates that a connection was rejected because this device does not accept the BD_ADDR. This may be because the device will only accept connections from specific BD_ADDRs.

2.16 CONNECTION ACCEPT TIMEOUT EXCEEDED (0X10)

The Connection Accept Timeout Exceeded error code indicates that the Connection Accept Timeout has been exceeded for this connection attempt.

2.17 UNSUPPORTED FEATURE OR PARAMETER VALUE (0X11)

The Unsupported Feature Or Parameter Value error code indicates that a feature or parameter value in the HCI Command is not supported. This error code shall not be used in an LMP PDU.

2.18 INVALID HCI COMMAND PARAMETERS (0X12)

The Invalid HCI Command Parameters error code indicates that at least one of the HCI command parameters is invalid.

This shall be used when:

- the parameter total length is invalid.
- a command parameter is an invalid type.
- a connection identifier does not match the corresponding event.
- a parameter value must be even.
- a parameter is outside of the specified range.
- two or more parameter values have inconsistent values.

Note: An invalid type can be, for example, when an SCO connection handle is used where an ACL connection handle is required.

2.19 REMOTE USER TERMINATED CONNECTION (0X13)

The Remote User Terminated Connection error code indicates that the user on the remote device terminated the connection.

2.20 REMOTE DEVICE TERMINATED CONNECTION DUE TO LOW RESOURCES (0X14)

The Remote Device Terminated Connection due to Low Resources error code indicates that the remote device terminated the connection because of low resources.

2.21 REMOTE DEVICE TERMINATED CONNECTION DUE TO POWER OFF (0X15)

The Remote Device Terminated Connection due to Power Off error code indicates that the remote device terminated the connection because the device is about to power off.

2.22 CONNECTION TERMINATED BY LOCAL HOST (0X16)

The Connection Terminated By Local Host error code indicates that the local device terminated the connection.

2.23 REPEATED ATTEMPTS (0X17)

The Repeated Attempts error code indicates that the controller is disallowing an authentication or pairing procedure because too little time has elapsed since the last authentication or pairing attempt failed.

2.24 PAIRING NOT ALLOWED (0X18)

The Pairing Not Allowed error code indicates that the device does not allow pairing. For example, when a device only allows pairing during a certain time window after some user input allows pairing.

2.25 UNKNOWN LMP PDU (0X19)

The Unknown LMP PDU error code indicates that the controller has received an unknown LMP opcode.

2.26 Unsupported Remote Feature / Unsupported LMP Feature (0X1A)

The Unsupported Remote Feature error code indicates that the remote device does not support the feature associated with the issued command or LMP PDU.

2.27 SCO OFFSET REJECTED (0X1B)

The SCO Offset Rejected error code indicates that the offset requested in the LMP_SCO_link_req message has been rejected.

2.28 SCO INTERVAL REJECTED (0X1C)

The SCO Interval Rejected error code indicates that the interval requested in the LMP_SCO_link_req message has been rejected.

2.29 SCO AIR MODE REJECTED (0X1D)

The SCO Air Mode Rejected error code indicates that the air mode requested in the LMP_SCO_link_req message has been rejected.

2.30 INVALID LMP PARAMETERS (0X1E)

The Invalid LMP Parameters error code indicates that some LMP message parameters were invalid. This shall be used when:

- the PDU length is invalid.
- a parameter value must be even.
- a parameter is outside of the specified range.
- two or more parameters have inconsistent values.

2.31 UNSPECIFIED ERROR (0X1F)

The Unspecified Error error code indicates that no other error code specified is appropriate to use.

2.32 UNSUPPORTED LMP PARAMETER VALUE (0X20)

The Unsupported LMP Parameter Value error code indicates that an LMP message contains at least one parameter value that is not supported by the controller at this time. This is normally used after a long negotiation procedure, for example during an LMP_hold_req, LMP_sniff_req and LMP_encryption_key_size_req message exchanges.

2.33 ROLE CHANGE NOT ALLOWED (0X21)

The Role Change Not Allowed error code indicates that a controller will not allow a role change at this time.

2.34 LMP RESPONSE TIMEOUT (0X22)

The LMP Response Timeout error code indicates that an LMP transaction failed to respond within the LMP response timeout.

2.35 LMP ERROR TRANSACTION COLLISION (0X23)

The LMP Error Transaction Collision error code indicates that an LMP transaction has collided with the same transaction that is already in progress.

2.36 LMP PDU NOT ALLOWED (0X24)

The LMP PDU Not Allowed error code indicates that a controller sent an LMP message with an opcode that was not allowed.

2.37 ENCRYPTION MODE NOT ACCEPTABLE (0X25)

The Encryption Mode Not Acceptable error code indicates that the requested encryption mode is not acceptable at this time.

2.38 LINK KEY CAN NOT BE CHANGED (0X26)

The Link Key Can Not be Changed error code indicates that a link key can not be changed because a fixed unit key is being used.

2.39 REQUESTED QoS NOT SUPPORTED (0X27)

The Requested QoS Not Supported error code indicates that the requested Quality of Service is not supported.

2.40 INSTANT PASSED (0X28)

The Instant Passed error code indicates that an LMP PDU that includes an instant can not be performed because the instant when this would have occurred has passed.

2.41 PAIRING WITH UNIT KEY NOT SUPPORTED (0X29)

The Pairing With Unit Key Not Supported error code indicates that it was not possible to pair as a unit key was requested and it is not supported.

2.42 DIFFERENT TRANSACTION COLLISION (0X2A)

The Different Transaction Collision error code indicates that an LMP transaction was started that collides with an ongoing transaction.

2.43 QoS UNACCEPTABLE PARAMETER (0X2C)

The QoS Unacceptable Parameter error code indicates that the specified quality of service parameters could not be accepted at this time, but other parameters may be acceptable.

2.44 QoS REJECTED (0X2D)

The QoS Rejected error code indicates that the specified quality of service parameters can not be accepted and QoS negotiation should be terminated.

2.45 CHANNEL CLASSIFICATION NOT SUPPORTED (0X2E)

The Channel Classification Not Supported error code indicates that the controller can not perform channel classification because it is not supported.

2.46 INSUFFICIENT SECURITY (0X2F)

The Insufficient Security error code indicates that the HCI command or LMP message sent is only possible on an encrypted link.

2.47 PARAMETER OUT OF MANDATORY RANGE (0X30)

The Parameter Out Of Mandatory Range error code indicates that a parameter value requested is outside the mandatory range of parameters for the given HCI command or LMP message.

2.48 ROLE SWITCH PENDING (0X32)

The Role Switch Pending error code indicates that a Role Switch is pending. This can be used when an HCI command or LMP message can not be accepted because of a pending role switch. This can also be used to notify a peer device about a pending role switch.

2.49 RESERVED SLOT VIOLATION (0X34)

The Reserved Slot Violation error code indicates that the current Synchronous negotiation was terminated with the negotiation state set to Reserved Slot Violation.

2.50 ROLE SWITCH FAILED (0X35)

The Role Switch Failed error code indicates that a role switch was attempted but it failed and the original piconet structure is restored. The switch may have failed because the TDD switch or piconet switch failed.

Core System Package [Controller volume]
Part E

HOST CONTROLLER INTERFACE FUNCTIONAL SPECIFICATION

This document describes the functional specification for the Host Controller Interface (HCI).
The HCI provides a command interface to the baseband controller and link manager, and access to configuration parameters. This interface provides a uniform method of accessing the Bluetooth baseband capabilities.

CONTENTS

1 INTRODUCTION

This document describes the functional specifications for the Host Controller Interface (HCI). The HCI provides a uniform interface method of accessing the Bluetooth controller capabilities. The next two sections provide a brief overview of the lower layers of the Bluetooth software stack and of the Bluetooth hardware. Section 2, provides an overview of the Lower HCI Device Driver Interface on the host device. Section 4, describes the flow control used between the Host and the Controller. Section 7, describes each of the HCI Commands in details, identifies parameters for each of the commands, and lists events associated with each command.

1.1 LOWER LAYERS OF THE BLUETOOTH SOFTWARE STACK

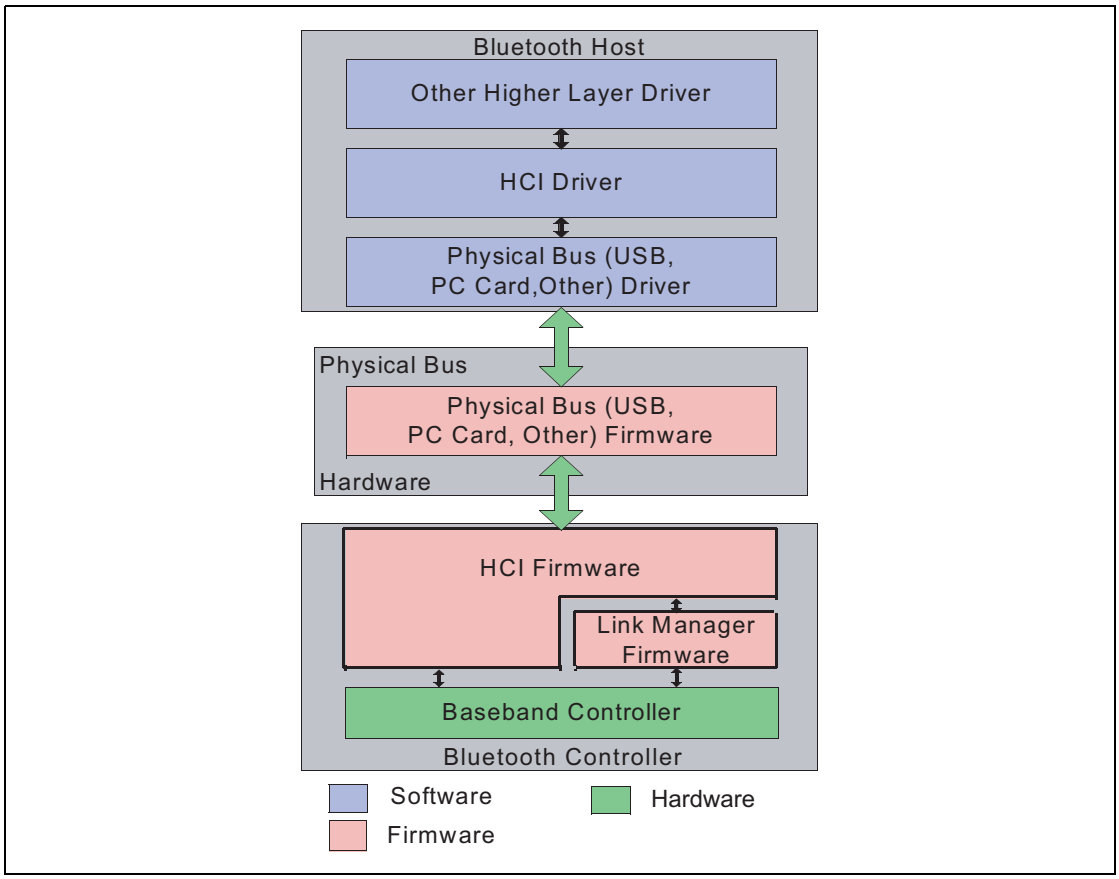


Figure 1.1: Overview of the Lower Software Layers

Figure 1.1, provides an overview of the lower software layers. The HCI firmware implements the HCI Commands for the Bluetooth hardware by accessing baseband commands link manager commands, hardware status registers, control registers, and event registers.

Several layers may exist between the HCI driver on the host system and the HCI firmware in the Bluetooth hardware. These intermediate layers, the Host

Controller Transport Layer, provide the ability to transfer data without intimate knowledge of the data.

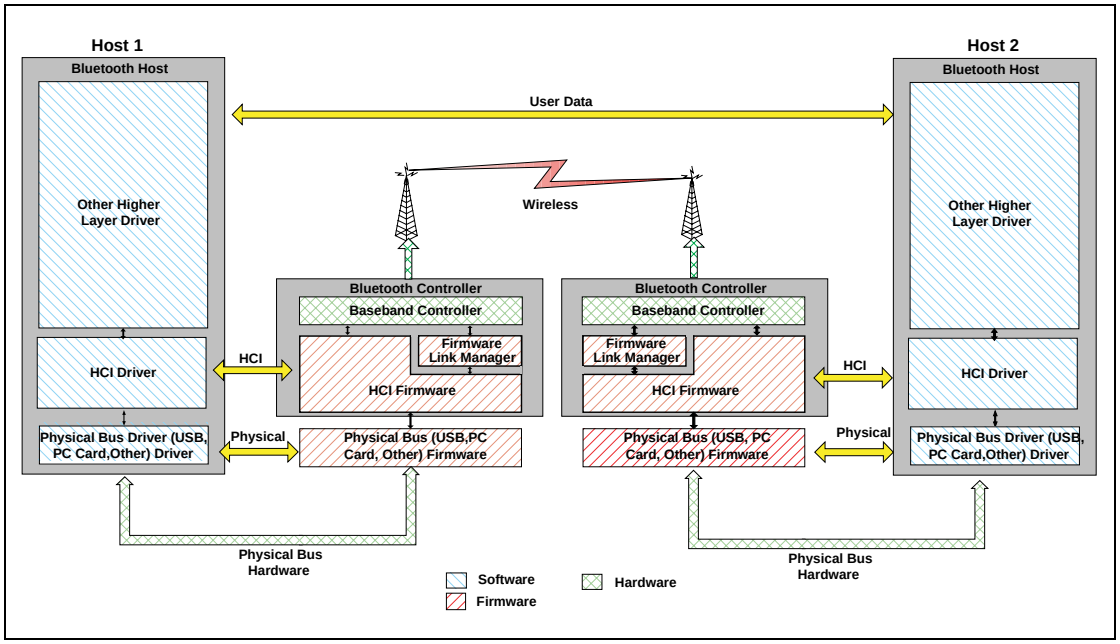


Figure 1.2: End to End Overview of Lower Software Layers to Transfer Data

Figure 1.2, illustrates the path of a data transfer from one device to another. The HCI driver on the Host exchanges data and commands with the HCI firmware on the Bluetooth hardware. The Host Control Transport Layer (i.e. physical bus) driver provides both HCI layers with the ability to exchange information with each other.

The Host will receive asynchronous notifications of HCI events independent of which Host Controller Transport Layer is used. HCI events are used for notifying the Host when something occurs. When the Host discovers that an event has occurred it will then parse the received event packet to determine which event occurred.

2 OVERVIEW OF HOST CONTROLLER TRANSPORT LAYER

The host driver stack has a transport layer between the Host Controller driver and the Host.

The main goal of this transport layer is transparency. The Host Controller driver (which interfaces to the Controller) should be independent of the underlying transport technology. Nor should the transport require any visibility into the data that the Host Controller driver passes to the Controller. This allows the interface (HCI) or the Controller to be upgraded without affecting the transport layer.

The specified Host Controller Transport Layers are described in a separate volume. (See specification volume 4)

3 OVERVIEW OF COMMANDS AND EVENTS

The commands and events are sent between the Host and the Controller. These are grouped into logical groups by function.

Generic Events	The generic events can occur due to multiple commands, or events that can occur at any time.
Device Setup	The device setup commands are used to place the Controller into a known state.
Controller Flow Control	The controller flow control commands and events are used to control data flow from the Host to the controller.
Controller Information	The controller information commands allow the Host to discover local information about the device.
Controller Configuration	The controller configuration commands and events allow the global configuration parameters to be configured.
Device Discovery	The device discovery commands and events allow a device to discover other devices in the surrounding area.
Connection Setup	The connection setup commands and events allow a device to make a connection to another device.
Remote Information	The remote information commands and events allow information about a remote device's configuration to be discovered.
Synchronous Connections	The synchronous connection commands and events allow synchronous connections to be created
Connection State	The connection state commands and events allow the configuration of a link, especially for low power operation.
Piconet Structure	The piconet structure commands and events allow the discovery and reconfiguration of piconet.
Quality of Service	The quality of service commands and events allow quality of service parameters to be specified.
Physical Links	The physical link commands and events allow the configuration of a physical link.
Host Flow Control	The Host flow control commands and events allow flow control to be used towards the Host.
Link Information	The link information commands and events allow information about a link to be read.
Authentication and Encryption	The authentication and encryption commands and events allow authentication of a remote device and then encryption of the link.
Testing	The testing commands and events allow a device to be placed into test mode.

Table 3.1: Overview of commands and events

The version information in this section denotes the version number of the specification that this command or event was first specified

3.1 GENERIC EVENTS

The generic events occur due to multiple commands, or events that can occur at any time.

Name	Vers.	Summary description
Command Complete Event	1.1	The Command Complete event is used by the Controller to pass the return status of a command and the other event parameters for each HCI Command.
Command Status Event	1.1	The Command Status event is used to indicate that the command described by the Command_Opcode parameter has been received and the Controller is currently performing the task for this command.
Hardware Error Event	1.1	The Hardware Error event is used to indicate some type of hardware failure for the Controller.

Table 3.2: Generic events

3.2 DEVICE SETUP

The device setup group of commands are used to place the Controller into a known state.

Name	Vers.	Summary description
Reset Command	1.1	The Reset command will reset the Controller, Link Manager, and the Bluetooth radio.

Device setup

3.3 CONTROLLER FLOW CONTROL

The controller flow control group of commands and events are used to control data flow from the Host to the Controller.

Name	Vers.	Summary description
Read Buffer Size Command	1.1	The Read_Buffer_Size command returns the size of the HCI buffers. These buffers are used by the Controller to buffer data that is to be transmitted.
Number Of Completed Packets Event	1.1	The Number Of Completed Packets event is used by the Controller to indicate to the Host how many HCI Data Packets have been completed for each Connection Handle since the previous Number Of Completed Packets event was sent.

Table 3.3: Controller flow control

3.4 CONTROLLER INFORMATION

The controller information group of commands allows the Host to discover local information about the device.

Name	Vers.	Summary description
Read Local Version Information Command	1.1	The Read Local Version Information command will read the version information for the local Bluetooth device.
Read Local Supported Commands Command	1.2	The Read Local Supported Commands command requests a list of the supported HCI commands for the local device.
Read Local Supported Features Command	1.1	The Read Local Supported Features command requests a list of the supported features for the local device.
Read Local Extended Features Command	1.2	The Read Local Extended Features command requests a list of the supported extended features for the local device
Read BD_ADDR Command	1.1	The Read BD_ADDR command will read the value for the BD_ADDR parameter.

Table 3.4: Controller information

3.5 CONTROLLER CONFIGURATION

The controller configuration group of commands and events allows the global configuration parameters to be configured.

Name	Vers.	Summary description
Read Local Name Command	1.1	The Read Local Name command provides the ability to read the stored user-friendly name for the Bluetooth device.
Write Local Name Command	1.1	The Write Local Name command provides the ability to modify the user-friendly name for the Bluetooth device.
Read Class of Device Command	1.1	The Read Class of Device command will read the value for the Class of Device configuration parameter, which is used to indicate its capabilities to other devices.
Write Class of Device Command	1.1	The Write Class of Device command will write the value for the Class_of_Device configuration parameter, which is used to indicate its capabilities to other devices.
Read Number Of Supported IAC Command	1.1	The Read Number of Supported IAC command will read the value for the number of Inquiry Access Codes (IAC) that the local Bluetooth device can simultaneously listen for during an Inquiry Scan.
Read Current IAC LAP Command	1.1	The Read Current IAC LAP command will read the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans.
Write Current IAC LAP Command	1.1	The Write Current IAC LAP command will write the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans.
Read Scan Enable Command	1.1	The Read Scan Enable command will read the value for the Scan Enable configuration parameter, which controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth devices.
Write Scan Enable Command	1.1	The Write Scan Enable command will write the value for the Scan Enable configuration parameter, which controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth devices.

Table 3.5: Controller configuration

3.6 DEVICE DISCOVERY

The device discovery group of commands and events allow a device to discovery other devices in the surrounding area.

Name	Vers.	Summary description
Inquiry Command	1.1	The Inquiry command will cause the Bluetooth device to enter Inquiry Mode. Inquiry Mode is used to discovery other nearby Bluetooth devices.
Inquiry Result Event	1.1	The Inquiry Result event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process.
Inquiry Result with RSSI Event	1.2	The Inquiry Result with RSSI event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process.
Inquiry Cancel Command	1.1	The Inquiry Cancel command will cause the Bluetooth device to stop the current Inquiry if the Bluetooth device is in Inquiry Mode.
Inquiry Complete Event	1.1	The Inquiry Complete event indicates that the Inquiry is finished.
Periodic Inquiry Mode Command	1.1	The Periodic Inquiry Mode command is used to configure the Bluetooth device to perform an automatic Inquiry based on a specified period range.
Exit Periodic Inquiry Mode Command	1.1	The Exit Periodic Inquiry Mode command is used to end the Periodic Inquiry mode when the local device is in Periodic Inquiry Mode.
Read Inquiry Scan Activity Command	1.1	The Read Inquiry Scan Activity command will read the value for Inquiry Scan Interval and Inquiry Scan Window configuration parameters. Inquiry Scan Interval defines the amount of time between consecutive inquiry scans. Inquiry Scan Window defines the amount of time for the duration of the inquiry scan.
Write Inquiry Scan Activity Command	1.1	The Write Inquiry Scan Activity command will write the value for Inquiry Scan Interval and Inquiry Scan Window configuration parameters. Inquiry Scan Interval defines the amount of time between consecutive inquiry scans. Inquiry Scan Window defines the amount of time for the duration of the inquiry scan.
Read Inquiry Scan Type Command	1.2	The Read Inquiry Scan Type command is used to read the Inquiry Scan Type configuration parameter of the local Bluetooth device. The Inquiry Scan Type configuration parameter can set the inquiry scan to either normal or interlaced scan.

Table 3.6: Device discovery

Name	Vers.	Summary description
Write Inquiry Scan Type Command	1.2	The Write Inquiry Scan Type command is used to write the Inquiry Scan Type configuration parameter of the local Bluetooth device. The Inquiry Scan Type configuration parameter can set the inquiry scan to either normal or interlaced scan.
Read Inquiry Mode Command	1.2	The Read Inquiry Mode command is used to read the Inquiry Mode configuration parameter of the local Bluetooth device.
Write Inquiry Mode Command	1.2	The Write Inquiry Mode command is used to write the Inquiry Mode configuration parameter of the local Bluetooth device.

Table 3.6: Device discovery

3.7 CONNECTION SETUP

The connection setup group of commands and events are used to allow a device to make a connection to another device.

Name	Vers.	Summary description
Create Connection Command	1.1	The Create Connection command will cause the link manager to create an ACL connection to the Bluetooth device with the BD_ADDR specified by the command parameters.
Connection Request Event	1.1	The Connection Request event is used to indicate that a new incoming connection is trying to be established.
Accept Connection Request Command	1.1	The Accept Connection Request command is used to accept a new incoming connection request.
Reject Connection Request Command	1.1	The Reject Connection Request command is used to decline a new incoming connection request.
Create Connection Cancel Command	1.2	The Create Connection Cancel Command is used to cancel an ongoing Create Connection.
Connection Complete Event	1.1	The Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been established.
Disconnect Command	1.1	The Disconnect command is used to terminate an existing connection.
Disconnection Complete Event	1.1	The Disconnection Complete event occurs when a connection has been terminated.
Read Page Timeout Command	1.1	The Read Page Timeout command will read the value for the Page Reply Timeout configuration parameter, which determines the time the Bluetooth controller will wait for the remote device to respond to a connection request before the local device returns a connection failure.
Write Page Timeout Command	1.1	The Write Page Timeout command will write the value for the Page Reply Timeout configuration parameter, which allows the Bluetooth hardware to define the amount of time a connection request will wait for the remote device to respond before the local device returns a connection failure.
Read Page Scan Activity Command	1.1	The Read Page Scan Activity command will read the values for the Page Scan Interval and Page Scan Window configuration parameters. Page Scan Interval defines the amount of time between consecutive page scans. Page Scan Window defines the duration of the page scan.

Table 3.7: Connection setup

Name	Vers.	Summary description
Write Page Scan Activity Command	1.1	The Write Page Scan Activity command will write the value for Page Scan Interval and Page Scan Window configuration parameters. Page Scan Interval defines the amount of time between consecutive page scans. Page Scan Window defines the duration of the page scan.
Page Scan Repetition Mode Change Event	1.1	The Page Scan Repetition Mode Change event indicates that the connected remote Bluetooth device with the specified Connection_Handle has successfully changed the Page_Scan_Repetition_Mode (SR)."
Read Page Scan Type Command	1.2	The Read Page Scan Type command is used to read the page scan type of the local Bluetooth device. The Page Scan Type configuration parameter can set the page scan to either normal or interlaced scan.
Write Page Scan Type Command	1.2	The Write Page Scan Type command is used to write the page scan type of the local Bluetooth device. The Page Scan Type configuration parameter can set the page scan to either normal or interlaced scan.
Read Connection Accept Timeout Command	1.1	The Read Connection Accept Timeout command will read the value for the Connection Accept Timeout configuration parameter, which allows the Bluetooth hardware to automatically deny a connection request after a specified period has occurred, and to refuse a new connection.
Write Connection Accept Timeout Command	1.1	The Write Connection Accept Timeout command will write the value for the Connection Accept Timeout configuration parameter, which allows the Bluetooth hardware to automatically deny a connection request after a specified period has occurred, and to refuse a new connection.

Table 3.7: Connection setup

3.8 REMOTE INFORMATION

The remote information group of commands and events allows information about a remote devices configuration to be discovered.

Name	Vers.	Summary description
Remote Name Request Command	1.1	The Remote Name Request command is used to obtain the user-friendly name of another Bluetooth device.
Remote Name Request Cancel Command	1.2	The Remote Name Request Cancel Command is used to cancel an ongoing Remote Name Request.
Remote Name Request Complete Event	1.1	The Remote Name Request Complete event is used to indicate a remote name request has been completed.
Read Remote Supported Features Command	1.1	The Read Remote Supported Features command requests a list of the supported features of a remote device.
Read Remote Supported Features Complete Event	1.1	The Read Remote Supported Features Complete event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote Bluetooth device specified by the Connection Handle event parameter.
Read Remote Extended Features Command	1.2	The Read Remote Extended Features command requests a list of the supported extended features of a remote device
Read Remote Extended Features Complete Event	1.2	The Read Remote Extended Features Complete Event is used to indicate the completion of the process of the Link Manager obtaining the supported Extended features of the remote Bluetooth device specified by the connection handle event parameter.
Read Remote Version Information Command	1.1	The Read Remote Version Information command will read the values for the version information for the remote Bluetooth device.
Read Remote Version Information Complete Event	1.1	The Read Remote Version Information Complete event is used to indicate the completion of the process of the Link Manager obtaining the version information of the remote Bluetooth device specified by the Connection Handle event parameter.

Table 3.8: Remote information

3.9 SYNCHRONOUS CONNECTIONS

The synchronous connections group of commands and events allows synchronous connections to be created.

Name	Vers.	Summary description
Setup Synchronous Connection Command	1.2	The Setup Synchronous Connection command adds a new or modifies an existing synchronous logical transport (SCO or eSCO) on a physical link depending on the Connection Handle parameter specified.
Synchronous Connection Complete Event	1.2	The Synchronous Connection Complete event indicates to both the Hosts that a new Synchronous connection has been established.
Synchronous Connection Changed event	1.2	The Synchronous Connection Changed event indicates to the Host that an existing Synchronous connection has been reconfigured.
Accept Synchronous Connection Request Command	1.2	The Accept_Synchronous_Connection_Request command is used to accept an incoming request for a synchronous connection and to inform the local Link Manager about the acceptable parameter values for the synchronous connection.
Reject Synchronous Connection Request Command	1.2	The Reject_Synchronous_Connection_Request is used to decline an incoming request for a synchronous link.
Read Voice Setting Command	1.1	The Read Voice Setting command will read the values for the Voice Setting configuration parameter, which controls all the various settings for the voice connections.
Write Voice Setting Command	1.1	The Write Voice Setting command will write the values for the Voice Setting configuration parameter, which controls all the various settings for the voice connections.

Table 3.9: Synchronous connections

3.10 CONNECTION STATE

The connection state group of commands and events allows the configuration of a link, especially for low power operation.

Name	Vers.	Summary description
Mode Change Event	1.1	The Mode Change event is used to indicate that the current mode has changed.
Max Slots Change Event	1.1	The Max Slots Change event it used to indicate a change in the max slots by the LM.
Hold Mode Command	1.1	The Hold Mode command is used to initiate Hold Mode.
Sniff Mode Command	1.1	The Sniff Mode command is used to alter the behavior of the LM and have the LM place the local or remote device into the sniff mode.
Exit Sniff Mode Command	1.1	The Exit Sniff Mode command is used to end the sniff mode for a connection handle which is currently in sniff mode.
Park State Command	1.1	The Park State command is used to alter the behavior of the LM and have the LM place the local or remote device into the park state.
Exit Park State Command	1.1	The Exit Park State command is used to switch the Bluetooth device from park state back to active mode.
Read Link Policy Settings Command	1.1	The Read Link Policy Settings command will read the Link Policy configuration parameter for the specified Connection Handle. The Link Policy settings allow the Host to specify which Link Modes the LM can use for the specified Connection Handle.
Write Link Policy Settings Command	1.1	The Write Link Policy Settings command will write the Link Policy configuration parameter for the specified Connection Handle. The Link Policy settings allow the Host to specify which Link Modes the LM can use for the specified Connection Handle.
Read Default Link Policy Settings Command	1.2	The Read Default Link Policy Settings command will read the Default Link Policy configuration parameter for all new connections.
Write Default Link Policy Settings Command	1.2	The Write Default Link Policy Settings command will write the Default Link Policy configuration parameter for all new connections.

Table 3.10: Connection state

3.11 PICONET STRUCTURE

The piconet structure group of commands and events allows the discovery and reconfiguration a piconet.

Name	Vers.	Summary description
Role Discovery Command	1.1	The Role Discovery command is used for a Bluetooth device to determine which role the device is performing for a particular Connection Handle.
Switch Role Command	1.1	The Switch Role command is used to switch master and slave roles of the devices on either side of a connection.
Role Change Event	1.1	The Role Change event is used to indicate that the current Bluetooth role related to the particular connection has been changed.

Table 3.11: Piconet structure

3.12 QUALITY OF SERVICE

The quality of service group of commands and events allows the configuration of links to allow for quality of service parameters to be specified.

Name	Vers.	Summary description
Flow Specification Command	1.2	The Flow Specification command is used to specify the flow parameters for the traffic carried over the ACL connection identified by the Connection Handle.
Flow Specification Complete Event	1.2	The Flow Specification Complete event is used to inform the Host about the Quality of Service for the ACL connection the Controller is able to support.
QoS Setup Command	1.1	The QoS Setup command is used to specify Quality of Service parameters for a connection handle.
QoS Setup Complete Event	1.1	The QoS Setup Complete event is used to indicate that QoS is setup.
QoS Violation Event	1.1	The QoS Violation event is used to indicate the Link Manager is unable to provide the current QoS requirement for the Connection Handle.
Flush Command	1.1	The Flush command is used to discard all data that is currently pending for transmission in the Controller for the specified connection handle.
Flush Occurred Event	1.1	The Flush Occurred event is used to indicate that, for the specified Connection Handle, the data to be transmitted has been discarded.
Read Automatic Flush Timeout Command	1.1	The Read Automatic Flush Timeout will read the value for the Flush Timeout configuration parameter for the specified connection handle. The Flush Timeout parameter is only used for ACL connections.
Write Automatic Flush Timeout Command	1.1	The Write Automatic Flush Timeout will write the value for the Flush Timeout configuration parameter for the specified connection handle. The Flush Timeout parameter is only used for ACL connections.
Read Failed Contact Counter Command	1.1	The Read Failed Contact Counter will read the value for the Failed Contact Counter configuration parameter for a particular connection to another device.
Reset Failed Contact Counter Command	1.1	The Reset Failed Contact Counter will reset the value for the Failed Contact Counter configuration parameter for a particular connection to another device.
Read Num Broadcast Retransmissions Command	1.1	The Read Num Broadcast Retransmissions command will read the parameter value for the Number of Broadcast Retransmissions for the device.
Write Num Broadcast Retransmissions Command	1.1	The Write Num Broadcast Retransmissions command will write the parameter value for the Number of Broadcast Retransmissions for the device.

Table 3.12: Quality of service

3.13 PHYSICAL LINKS

The physical links commands and events allows configuration of the physical link.

Name	Vers.	Summary description
Read Link Supervision Timeout Command	1.1	The Read Link Supervision Timeout command will read the value for the Link Supervision Timeout configuration parameter for the device. This parameter is used by the device to determine link loss.
Write Link Supervision Timeout Command	1.1	The Write Link Supervision Timeout command will write the value for the Link Supervision Timeout configuration parameter for the device. This parameter is used by the device to determine link loss.
Read AFH Channel Assessment Mode Command	1.2	The Read AFH Channel Assessment Mode command will read the value for the AFH Channel Classification Mode parameter. This value is used to enable or disable the Controller's channel assessment scheme.
Write AFH Channel Assessment Mode Command	1.2	The Write AFH Channel Assessment Mode command will write the value for the Channel Classification Mode configuration parameter. This value is used to enable or disable the Controller's channel assessment scheme.
Set AFH Host Channel Classification Command	1.2	The Set AFH Host Channel Classification command allows the Host to specify a channel classification based on its "local information".
Change Connection Packet Type Command	1.1	The Change Connection Packet Type command is used to change which packet types can be used for a connection that is currently established.
Connection Packet Type Changed Event	1.1	The Connection Packet Type Changed event is used to indicate the completion of the process of the Link Manager changing the packet type mask used for the specified Connection Handle.

Table 3.13: Physical links

3.14 HOST FLOW CONTROL

The Host flow control group of commands and events allows flow control to be used towards the Host.

Name	Vers.	Summary description
Host Buffer Size Command	1.1	The Host Buffer Size command is used by the Host to notify the Controller about its buffer sizes for ACL and synchronous data. The Controller will segment the data to be transmitted from the Controller to the Host, so that data contained in HCI Data Packets will not exceed these sizes.
Set Event Mask Command	1.1	The Set Event Mask command is used to control which events are generated by the HCI for the Host.
Set Event Filter Command	1.1	The Set Event Filter command is used by the Host to specify different event filters. The Host may issue this command multiple times to request various conditions for the same type of event filter and for different types of event filters.
Set Controller To Host Flow Control Command	1.1	The Set Controller To Host Flow Control command is used by the Host to turn flow control on or off in the direction from the Controller to the Host.
Host Number Of Completed Packets Command	1.1	The Host Number Of Completed Packets command is used by the Host to indicate to the Controller when the Host is ready to receive more HCI packets for any connection handle.
Data Buffer Overflow Event	1.1	The Data Buffer Overflow event is used to indicate that the Controller's data buffers have overflowed, because the Host has sent more packets than allowed.
Read Synchronous Flow Control Enable Command	1.1	The Read Synchronous Flow Control Enable command provides the ability to read the Synchronous Flow Control Enable setting. By using this setting, the Host can decide if the Controller will send Number Of Completed Packets events for Synchronous Connection Handles.
Write Synchronous Flow Control Enable Command	1.1	The Write Synchronous Flow Control Enable command provides the ability to write the Synchronous Flow Control Enable setting. By using this setting, the Host can decide if the Controller will send Number Of Completed Packets events for Synchronous Connection Handles.

Table 3.14: Controller flow control.

3.15 LINK INFORMATION

The link information group of commands and events allows information about a link to be read.

Name	Vers.	Summary description
Read LMP Handle Command	1.2	The Read LMP Handle command will read the current LMP Handle associated with the Connection Handle.
Read Transmit Power Level Command	1.1	The Read Transmit Power Level command will read the values for the Transmit Power Level parameter for the specified Connection Handle.
Read Link Quality Command	1.1	The Read Link Quality command will read the value for the Link Quality for the specified Connection Handle.
Read RSSI Command	1.1	The Read RSSI command will read the value for the Received Signal Strength Indication (RSSI) for a connection handle to another Bluetooth device.
Read Clock Offset Command	1.1	The Read Clock Offset command allows the Host to read the clock offset of remote devices.
Read Clock Offset Complete Event	1.1	The Read Clock Offset Complete event is used to indicate the completion of the process of the LM obtaining the Clock offset information.
Read Clock Command	1.2	The Read Clock command will read an estimate of a piconet or the local Bluetooth Clock.
Read AFH Channel Map Command	1.2	The Read AFH Channel Map command will read the current state of the channel map for a connection.

Table 3.15: Link information

3.16 AUTHENTICATION AND ENCRYPTION

The authentication and encryption group of commands and events allows authentication of a remote device and then encryption of the link to one or more remote devices.

Name	Vers.	Summary description
Read Authentication Enable Command	1.1	The Read Authentication Enable command will read the value for the Authentication Enable parameter, which controls whether the Bluetooth device will require authentication for each connection with other Bluetooth devices.
Write Authentication Enable Command	1,1	The Write Authentication Enable command will write the value for the Authentication Enable parameter, which controls whether the Bluetooth device will require authentication for each connection with other Bluetooth devices.
Read Encryption Mode Command	1.1	The Read Encryption Mode command will read the value for the Encryption Mode parameter, which controls whether the Bluetooth device will require encryption for each connection with other Bluetooth devices.
Write Encryption Mode Command	1.1	The Write Encryption Mode command will write the value for the Encryption Mode parameter, which controls whether the Bluetooth device will require encryption for each connection with other Bluetooth devices.
Link Key Request Event	1.1	The Link Key Request event is used to indicate that a Link Key is required for the connection with the device specified in BD_ADDR.
Link Key Request Reply Command	1.1	The Link Key Request Reply command is used to reply to a Link Key Request event from the Controller, and specifies the Link Key stored on the Host to be used as the link key for the connection with the other Bluetooth device specified by BD_ADDR.
Link Key Request Negative Reply Command	1.1	The Link Key Request Negative Reply command is used to reply to a Link Key Request event from the Controller if the Host does not have a stored Link Key for the connection with the other Bluetooth Device specified by BD_ADDR.
PIN Code Request Event	1.1	The PIN Code Request event is used to indicate that a PIN code is required to create a new link key for a connection.
PIN Code Request Reply Command	1.1	The PIN Code Request Reply command is used to reply to a PIN Code Request event from the Controller and specifies the PIN code to use for a connection.

Table 3.16: Authentication and encryption

Name	Vers.	Summary description
PIN Code Request Negative Reply Command	1.1	The PIN Code Request Negative Reply command is used to reply to a PIN Code Request event from the Controller when the Host cannot specify a PIN code to use for a connection.
Link Key Notification Event	1.1	The Link Key Notification event is used to indicate to the Host that a new Link Key has been created for the connection with the device specified in BD_ADDR.
Authentication Requested Command	1.1	The Authentication Requested command is used to establish authentication between the two devices associated with the specified Connection Handle.
Authentication Complete Event	1.1	The Authentication Complete event occurs when authentication has been completed for the specified connection.
Set Connection Encryption Command	1.1	The Set Connection Encryption command is used to enable and disable the link level encryption.
Encryption Change Event	1.1	The Encryption Change event is used to indicate that the change in the encryption has been completed for the Connection Handle specified by the Connection Handle event parameter.
Change Connection Link Key Command	1.1	The Change Connection Link Key command is used to force both devices of a connection associated to the connection handle, to generate a new link key.
Change Connection Link Key Complete Event	1.1	The Change Connection Link Key Complete event is used to indicate that the change in the Link Key for the Connection Handle specified by the Connection Handle event parameter had been completed.
Master Link Key Command	1.1	The Master Link Key command is used to force both devices of a connection associated to the connection handle to use the temporary link key of the Master device or the regular link keys.
Master Link Key Complete Event	1.1	The Master Link Key Complete event is used to indicate that the change in the temporary Link Key or in the semi-permanent link keys on the Bluetooth master side has been completed.
Read PIN Type Command	1.1	The Read PIN Type command is used for the Host to read the value that is specified to indicate whether the Host supports variable PIN or only fixed PINs.
Write PIN Type Command	1.1	The Write PIN Type command is used for the Host to specify whether the Host supports variable PIN or only fixed PINs.
Read Stored Link Key Command	1.1	The Read Stored Link Key command provides the ability to read one or more link keys stored in the Controller.

Table 3.16: Authentication and encryption

Name	Vers.	Summary description
Return Link Keys Event	1.1	The Return Link Keys event is used to return stored link keys after a Read Stored Link Key command is used.
Write Stored Link Key Command	1.1	The Write Stored Link Key command provides the ability to write one or more link keys to be stored in the Controller.
Delete Stored Link Key Command	1.1	The Delete Stored Link Key command provides the ability to remove one or more of the link keys stored in the Controller.
Create New Unit Key Command	1.1	The Create New Unit Key command is used to create a new unit key.

Table 3.16: Authentication and encryption

3.17 TESTING

The testing group of commands and events allows a device to be placed into a special testing mode to allow for testing to be performed.

Name	Vers.	Summary description
Read Loopback Mode Command	1.1	The Read Loopback Mode will read the value for the setting of the Controllers Loopback Mode. The setting of the Loopback Mode will determine the path of information.
Write Loopback Mode Command	1.1	The Write Loopback Mode will write the value for the setting of the Controllers Loopback Mode. The setting of the Loopback Mode will determine the path of information.
Loopback Command Event	1.1	The Loopback Command event is used to loop back all commands that the Host sends to the Controller with some exceptions.
Enable Device Under Test Mode Command	1.1	The Enable Device Under Test Mode command will allow the local Bluetooth module to enter test mode via LMP test commands. The Host issues this command when it wants the local device to be the DUT for the Testing scenarios as described in the Bluetooth Test Mode document.

Table 3.17: Testing

3.18 ALPHABETICAL LIST OF COMMANDS AND EVENTS

Commands/Events	Group
Accept Connection Request Command	Connection Setup
Authentication Complete Event	Authentication and Encryption
Authentication Requested Command	Authentication and Encryption
Change Connection Link Key Command	Authentication and Encryption
Change Connection Link Key Complete Event	Authentication and Encryption
Change Connection Packet Type Command	Physical Links
Command Complete Event	Generic Events
Command Status Event	Generic Events
Connection Complete Event	Connection Setup
Connection Packet Type Changed Event	Physical Links
Connection Request Event	Connection Setup
Create Connection Cancel Command	Connection Setup
Create Connection Command	Connection Setup
Create New Unit Key Command	Authentication and Encryption
Data Buffer Overflow Event	Host Flow Control
Delete Stored Link Key Command	Authentication and Encryption
Disconnect Command	Connection Setup
Disconnection Complete Event	Connection Setup
Enable Device Under Test Mode Command	Testing
Encryption Change Event	Authentication and Encryption
Exit Park State Command	Connection State
Exit Periodic Inquiry Mode Command	Device Discovery
Exit Sniff Mode Command	Connection State
Flow Specification Command	Quality of Service
Flow Specification Complete Event	Quality of Service
Flush Command	Quality of Service
Flush Occurred Event	Quality of Service
Hardware Error Eventt	Generic Events
Hold Mode Command	Connection State
Host Buffer Size Command	Host Flow Control

Table 3.18: Alphabetical list of commands and events.

Commands/Events	Group
Host Number Of Completed Packets Command	Host Flow Control
Inquiry Cancel Command	Device Discovery
Inquiry Command	Device Discovery
Inquiry Complete Event	Device Discovery
Inquiry Result Event	Device Discovery
Inquiry Result with RSSI Event	Device Discovery
Link Key Notification Event	Authentication and Encryption
Link Key Request Event	Authentication and Encryption
Link Key Request Negative Reply Command	Authentication and Encryption
Link Key Request Reply Command	Authentication and Encryption
Loopback Command Event	Testing
Master Link Key Command	Authentication and Encryption
Master Link Key Complete Event	Authentication and Encryption
Max Slots Change Event	Connection State
Mode Change Event	Connection State
Number Of Completed Packets Event	Controller Flow Control
Page Scan Repetition Mode Change Event	Connection Setup
Park State Command	Connection State
Periodic Inquiry Mode Command	Device Discovery
PIN Code Request Event	Authentication and Encryption
PIN Code Request Negative Reply Command	Authentication and Encryption
PIN Code Request Reply Command	Authentication and Encryption
QoS Setup Command	Quality of Service
QoS Setup Complete Event	Quality of Service
QoS Violation Event	Quality of Service
Read AFH Channel Assessment Mode Command	Physical Links
Read AFH Channel Map Command	Link Information
Read Authentication Enable Command	Authentication and Encryption
Read Automatic Flush Timeout Command	Quality of Service
Read BD_ADDR Command	Controller Information
Read Buffer Size Command	Controller Flow Control
Read Class of Device Command	Controller Information

Table 3.18: Alphabetical list of commands and events.

Commands/Events	Group
Read Clock Command	Link Information
Read Clock Offset Command	Link Information
Read Clock Offset Complete Event	Link Information
Read Connection Accept Timeout Command	Connection Setup
Read Current IAC LAP Command	Controller Information
Read Default Link Policy Settings Command	Connection State
Read Encryption Mode Command	Authentication and Encryption
Read Failed Contact Counter Command	Quality of Service
Read Hold Mode Activity Command	Connection State
Read Inquiry Mode Command	Device Discovery
Read Inquiry Scan Activity Command	Device Discovery
Read Inquiry Scan Type Command	Device Discovery
Read Link Policy Settings Command	Connection State
Read Link Quality Command	Link Information
Read Link Supervision Timeout Command	Physical Links
Read LMP Handle Command	Link Information
Read Local Extended Features Command	Controller Information
Read Local Name Command	Controller Configuration
Read Local Supported Commands Command	Controller Information
Read Local Supported Features Command	Controller Information
Read Local Version Information Command	Controller Information
Read Loopback Mode Command	Testing
Read Num Broadcast Retransmissions Command	Quality of Service
Read Number Of Supported IAC Command	Controller Information
Read Page Scan Activity Command	Connection Setup
Read Page Scan Type Command	Connection Setup
Read Page Timeout Command	Connection Setup
Read PIN Type Command	Authentication and Encryption
Read Remote Extended Features Command	Remote Information
Read Remote Extended Features Complete Event	Remote Information
Read Remote Supported Features Command	Remote Information
Read Remote Supported Features Complete Event	Remote Information

Table 3.18: Alphabetical list of commands and events.

Commands/Events	Group
Read Remote Version Information Command	Remote Information
Read Remote Version Information Complete Event	Remote Information
Read RSSI Command	Link Information
Read Scan Enable Command	Controller Information
Read Stored Link Key Command	Authentication and Encryption
Read Synchronous Flow Control Enable Command	Host Flow Control
Read Transmit Power Level Command	Link Information
Read Voice Setting Command	Synchronous Connections
Reject Connection Request Command	Connection Setup
Remote Name Request Cancel Command	Remote Information
Remote Name Request Command	Remote Information
Remote Name Request Complete Event	Remote Information
Reset Command	Device Setup
Reset Failed Contact Counter Command	Quality of Service
Return Link Keys Event	Authentication and Encryption
Role Change Event	Piconet Structure
Role Discovery Command	Piconet Structure
Set AFH Host Channel Classification Command	Physical Links
Set Connection Encryption Command	Authentication and Encryption
Set Controller To Host Flow Control Command	Host Flow Control
Set Event Filter Command	Host Flow Control
Set Event Mask Command	Host Flow Control
Setup Synchronous Connection Command	Synchronous Connections
Sniff Mode Command	Connection State
Switch Role Command	Piconet Structure
Synchronous Connection Changed event	Synchronous Connections
Synchronous Connection Complete Event	Synchronous Connections
Write AFH Channel Assessment Mode Command	Physical Links
Write Authentication Enable Command	Authentication and Encryption
Write Automatic Flush Timeout Command	Quality of Service
Write Class of Device Command	Controller Information
Write Connection Accept Timeout Command	Connection Setup

Table 3.18: Alphabetical list of commands and events.

Commands/Events	Group
Write Current IAC LAP Command	Controller Information
Write Default Link Policy Settings Command	Connection State
Write Encryption Mode Command	Authentication and Encryption
Write Hold Mode Activity Command	Connection State
Write Inquiry Mode Command	Device Discovery
Write Inquiry Scan Activity Command	Device Discovery
Write Inquiry Scan Type Command	Device Discovery
Write Link Policy Settings Command	Connection State
Write Link Supervision Timeout Command	Physical Links
Write Local Name Command	Controller Information
Write Loopback Mode Command	Testing
Write Num Broadcast Retransmissions Command	Quality of Service
Write Page Scan Activity Command	Connection Setup
Write Page Scan Type Command	Connection Setup
Write Page Timeout Command	Connection Setup
Write PIN Type Command	Authentication and Encryption
Write Scan Enable Command	Controller Information
Write Stored Link Key Command	Authentication and Encryption
Write Synchronous Flow Control Enable Command	Host Flow Control
Write Voice Setting Command	Synchronous Connections

Table 3.18: Alphabetical list of commands and events.

4 HCI FLOW CONTROL

Flow control shall be used in the direction from the Host to the Controller to avoid overflowing the Controller data buffers with ACL data destined for a remote device (using a connection handle) that is not responding. The Host manages the data buffers of the Controller.

4.1 HOST TO CONTROLLER DATA FLOW CONTROL

On initialization, the Host shall issue the Read Buffer Size command. Two of the return parameters of this command determine the maximum size of HCI ACL and synchronous Data Packets (excluding header) sent from the Host to the Controller. There are also two additional return parameters that specify the total number of HCI ACL and synchronous Data Packets that the Controller may have waiting for transmission in its buffers.

When there is at least one connection to another device, or when in local loop-back mode, the Controller shall use the Number Of Completed Packets event to control the flow of data from the Host. This event contains a list of connection handles and a corresponding number of HCI Data Packets that have been completed (transmitted, flushed, or looped back to the Host) since the previous time the event was returned (or since the connection was established, if the event has not been returned before for a particular connection handle).

Based on the information returned in this event, and the return parameters of the Read Buffer Size command that specify the total number of HCI ACL and synchronous Data Packets that can be stored in the Controller, the Host decides for which Connection Handles the following HCI Data Packets should be sent.

Every time it has sent an HCI Data Packet, the Host shall assume that the free buffer space for the corresponding link type (ACL, SCO or eSCO) in the Controller has decreased by one HCI Data Packet.

Each Number Of Completed Packets event received by the Host provides information about how many HCI Data Packets have been completed (transmitted or flushed) for each Connection Handle since the previous Number Of Completed Packets event was sent to the Host. It can then calculate the actual current buffer usage.

When the Controller has completed one or more HCI Data Packet(s) it shall send a Number Of Completed Packets event to the Host, until it finally reports that all the pending HCI Data Packets have been completed. The frequency at which this event is sent is manufacturer specific.

Note: The Number Of Completed Packets events will not report on synchronous connection handles if Synchronous Flow Control is disabled. (See Read/

Write Synchronous Flow Control Enable, Section 7.3.38 on page 513 and Section 7.3.39 on page 514)

For each individual Connection Handle, the data must be sent to the Controller in HCI Data Packets in the order in which it was created in the Host. The Controller shall also transmit data on the air that is received from the Host for a given Connection Handle in the same order as it is received from the Host.

Data that is received on the air from another device shall, for the corresponding Connection Handle, be sent in HCI Data Packets to the Host in the same order as it is received. This means the scheduling shall be decided separately for each Connection Handle basis. For each individual Connection Handle, the order of the data shall not be changed from the order in which the data has been created.

4.2 CONTROLLER TO HOST DATA FLOW CONTROL

In some implementations, flow control may also be necessary in the direction from the Controller to the Host. The Set Host Controller To Host Flow Control command can be used to turn flow control on or off in that direction.

On initialization, the Host uses the Host Buffer Size command to notify the Controller about the maximum size of HCI ACL and synchronous Data Packets sent from the Controller to the Host. The command also contains two additional command parameters to notify the Controller about the total number of ACL and synchronous Data Packets that can be stored in the data buffers of the Host.

The Host uses the Host Number Of Completed Packets command in exactly the same way as the Controller uses the Number Of Completed Packets event as was previously described in this section.

The Host Number Of Completed Packets command is a special command for which no command flow control is used, and which can be sent anytime there is a connection or when in local loopback mode. The command also has no event after the command has completed. This makes it possible for the flow control to work in exactly the same way in both directions, and the flow of normal commands will not be disturbed.

4.3 DISCONNECTION BEHAVIOR

When the Host receives a Disconnection Complete event, the Host shall assume that all unacknowledged HCI Data Packets that have been sent to the Controller for the returned Connection Handle have been flushed, and that the corresponding data buffers have been freed. The Controller does not have to notify the Host about this in a Number Of Completed Packets event.

If flow control is also enabled in the direction from the Controller to the Host, the Controller may, after it has sent a Disconnection Complete event, assume that the Host will flush its data buffers for the sent Connection Handle when it receives the Disconnection Complete event. The Host does not have to notify the Controller about this in a Host Number Of Completed Packets command.

4.4 COMMAND FLOW CONTROL

On initial power-on, and after a reset, the Host shall send a maximum of one outstanding HCI Command Packet until a Command Complete or Command Status event has been received.

The Command Complete and Command Status events contain a parameter called Num HCI Command Packets, which indicates the number of HCI Command Packets the Host is currently allowed to send to the Controller. The Controller may buffer one or more HCI command packets, but the Controller must start performing the commands in the order in which they are received. The Controller can start performing a command before it completes previous commands. Therefore, the commands do not always complete in the order they are started.

To indicate to the Host that the Controller is ready to receive HCI command packets, the Controller may generate a Command Complete event with the Command Opcode 0x0000, and the Num HCI Command Packets event parameter is set to 1 or more. Command Opcode 0x0000 is a NOP (No OPeration), and can be used to change the number of outstanding HCI command packets that the Host can send. The Controller may generate a Command Complete event with the Num HCI Command Packets event parameter set to zero to inform Host it must stop sending commands.

For most commands, a Command Complete event shall be sent to the Host when the Controller completes the command. Some commands are executed in the background and do not return a Command Complete event when they have been completed. Instead, the Controller shall send a Command Status event back to the Host when it has begun to execute the command. When the actions associated with the command have finished, an event that is associated with the command shall be sent by the Controller to the Host.

If the command does not begin to execute (for example, if there was a parameter error or the command is currently not allowed), the Command Status event shall be returned with the appropriate error code in the Status parameter, and the event associated with the sent command shall not be returned.

4.5 COMMAND ERROR HANDLING

If an error occurs for a command for which a Command Complete event is returned, the Return Parameters field may not contain all the return parameters specified for the command. The Status parameter, which explains the error reason and which is the first return parameter, shall always be returned. If there is a Connection Handle parameter or a BD_ADDR parameter right after the Status parameter, this parameter shall also be returned so that the Host can identify to which instance of a command the Command Complete event belongs. In this case, the Connection Handle or BD_ADDR parameter shall have exactly the same value as that in the corresponding command parameter. It is implementation specific whether more parameters will be returned in case of an error.

The above also applies to commands that have associated command specific completion events with a status parameter in their completion event, with two exceptions. The exceptions are the Connection Complete and the Synchronous Connection Complete events. On failure, for these two events only, the second parameter, Connection_Handle, is not valid and the third parameter, BD_ADDR, is valid for identification purposes. The validity of other parameters is likewise implementation specific for failed commands in this group.

Note: The BD_ADDR return parameter of the command Read BD_ADDR is not used to identify to which instance of the Read BD ADDR command the Command Complete event belongs. It is optional for the Controller to return this parameter in case of an error.

5 HCI DATA FORMATS

5.1 INTRODUCTION

The HCI provides a uniform command method of accessing the Bluetooth capabilities. The HCI Link commands provide the Host with the ability to control the link layer connections to other Bluetooth devices. These commands typically involve the Link Manager (LM) to exchange LMP commands with remote Bluetooth devices. For details see "Link Manager Protocol" on page 211 [Part C].

The HCI Policy commands are used to affect the behavior of the local and remote LM. These Policy commands provide the Host with methods of influencing how the LM manages the piconet. The Controller & Baseband, Informational, and Status commands provide the Host access to various registers in the Controller.

HCI commands may take different amounts of time to be completed. Therefore, the results of commands will be reported back to the Host in the form of an event. For example, for most HCI commands the Controller will generate the Command Complete event when a command is completed. This event contains the return parameters for the completed HCI command. For enabling the Host to detect errors on the HCI-Transport Layer, there needs to be a timeout between the transmission of the Host's command and the reception of the Controller's response (e.g. a Command Complete or Command Status event). Since the maximum response timeout is strongly dependent on the HCI-Transport Layer used, it is recommended to use a default value of one second for this timer. This amount of time is also dependent on the number of commands unprocessed in the command queue.

5.2 DATA AND PARAMETER FORMATS

- All values are in Binary and Hexadecimal Little Endian formats unless otherwise noted
- In addition, all parameters which can have negative values must use 2's complement when specifying values
- Arrayed parameters are specified using the following notation: ParameterA[i]. If more than one set of arrayed parameters are specified (e.g. ParameterA[i], ParameterB[i]), then the order of the parameters are as follows: ParameterA[0], ParameterB[0], ParameterA[1], ParameterB[1], ParameterA[2], ParameterB[2], … ParameterA[n], ParameterB[n]
- Unless noted otherwise, all parameter values are sent and received in Little Endian format (i.e. for multi-octet parameters the rightmost (Least Signification Octet) is transmitted first)
- All command and event parameters that are not-arrayed and all elements in an arrayed parameter have fixed sizes (an integer number of octets). The parameters and the size of each not arrayed parameter (or of each element in an arrayed parameter) contained in a command or an event is specified

for each command or event. The number of elements in an arrayed parameter is not fixed.

- Where bit strings are specified, the low order bit is the right hand bit, e.g. 0 is the low order bit in ‘10’.
- Values or parameters marked as Reserved for Future Use, shall be set to 0 unless explicitly stated otherwise on transmission, and shall be ignored on reception. Parameter values or opcodes that an implementation does not know how to interpret shall be ignored, and the operation that is being attempted shall be completed with the correct signaling. The host or controller shall not stop functioning because of receiving a reserved value.

5.3 CONNECTION HANDLES

Connection Handles are used to identify logical channels between the Host and Controller. Connection Handles are assigned by the Controller when a new logical link is created, using the Connection Complete or Synchronous Connection Complete events. Broadcast Connection Handles are handled differently, and are described below.

The first time the Host sends an HCI Data Packet with Broadcast_Flag set to 01b (active slave broadcast) or 10b (parked slave broadcast) after a power-on or a reset, the value of the Connection Handle parameter must be a value which is not currently assigned by the Host Controller. The Host must use different connection handles for active broadcast and piconet broadcast.

The Controller must then continue to use the same connection handles for each type of broadcast until a reset is made. Note: The Host Controller must not send a Connection Complete event containing a new Connection_Handle that it knows is used for broadcast.

Note: In some situations, it may happen that the Host Controller sends a Connection Complete event before having interpreted a Broadcast packet received from the Host, and that the Connection_Handles of both Connection Complete event and HCI Data packet are the same. This conflict has to be avoided as follows:

If a Connection Complete event is received containing one of the connection handles used for broadcast, the Host has to wait before sending any packets for the new connection until it receives a Number Of Completed Packets event indicating that there are no pending broadcast packets belonging to the connection handle. In addition, the Host must change the Connection_Handle used for the corresponding type of broadcast to a Connection_Handle which is currently not assigned by the Host Controller.

This Connection_Handle must then be used for all the following broadcasts of that type until a reset is performed or the same conflict situation happens again. However, this will occur very rarely.

The Host Controller must, in the above conflict case, be able to distinguish between the Broadcast message sent by the Host and the new connection made (this could be even a new synchronous link) even though the connection handles are the same.

For an HCI Data Packet sent from the Host Controller to the Host where the Broadcast_Flag is 01 or 10, the Connection_Handle parameter should contain the connection handle for the ACL connection to the master that sent the broadcast.

Note: Connection handles used for Broadcast do not identify an ACL point-to-point connection, so they must not be used in any command having a Connection_Handle parameter and they will not be returned in any event having a Connection_Handle parameter except the Number Of Completed Packets event.

5.4 EXCHANGE OF HCI-SPECIFIC INFORMATION

The Host Controller Transport Layer provides transparent exchange of HCI specific information. These transporting mechanisms provide the ability for the Host to send HCI commands, ACL data and synchronous data to the Controller. These transport mechanisms also provide the ability for the Host to receive HCI events, ACL data and synchronous data from the Controller. Since the Host Controller Transport Layer provides transparent exchange of HCI-specific information, the HCI specification specifies the format of the commands, events, and data exchange between the Host and the Controller. The next sections specify the HCI packet formats.

5.4.1 HCI Command Packet

The HCI Command Packet is used to send commands to the Controller from the Host. The format of the HCI Command Packet is shown in Figure 5.1, and the definition of each field is explained below.

The Controller must be able to accept HCI Command Packets with up to 255 bytes of data excluding the HCI Command Packet header.

Each command is assigned a 2 byte Opcode used to uniquely identify different types of commands. The Opcode parameter is divided into two fields, called the OpCode Group Field (OGF) and OpCode Command Field (OCF). The OGF occupies the upper 6 bits of the Opcode, while the OCF occupies the remaining 10 bits. The OGF of 0x3F is reserved for vendor-specific debug commands. The OGF of 0x3E is reserved for Bluetooth Logo Testing. The organization of the Opcodes allows additional information to be inferred without fully decoding the entire Opcode.

Note: the OGF composed of all ‘ones’ has been reserved for vendor-specific debug commands. These commands are vendor-specific and are used during manufacturing, for a possible method for updating firmware, and for debugging.

Note: the OGF composed of all ‘zeros’ and an OCF or all ‘zeros’ is the NOP command. This command Opcode may be used in Command Flow Control. (See Section 4.4 on page 373)

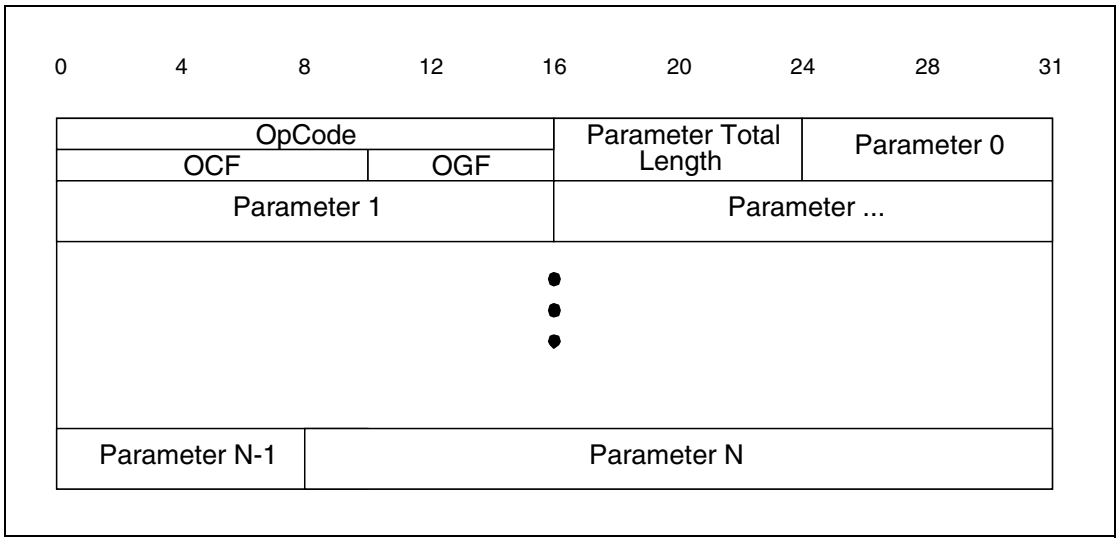


Figure 5.1: HCI Command Packet

Op_Code: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	OGFRange (6 bits): 0x00-0x3F (0x3E reserved for Bluetooth logo testing and 0x3F reserved for vendor-specific debug commands) OCF Range (10 bits): 0x0000-0x03FF

Parameter_Total_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	Lengths of all of the parameters contained in this packet measured in octets. (N.B.: total length of parameters, not number of parameters)

Parameter 0 - N: *Size: Parameter Total Length*

Value	Parameter Description
0xXX	Each command has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each command. Each parameter is an integer number of octets in size.

5.4.2 HCI ACL Data Packets

HCI ACL Data Packets are used to exchange data between the Host and Controller. The format of the HCI ACL Data Packet is shown in Figure 5.2. The definition for each of the fields in the data packets is explained below.

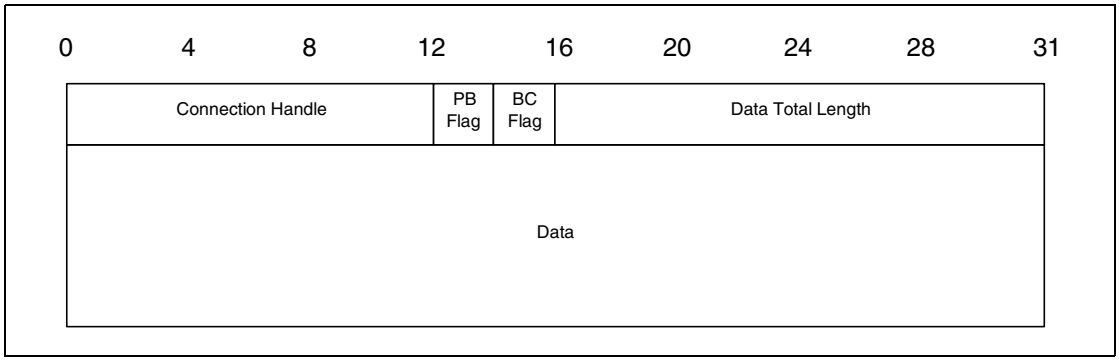


Figure 5.2: HCI ACL Data Packet

Connection_Handle: *Size: 12 Bits*

Value	Parameter Description
0xXXX	Connection Handle to be used for transmitting a data packet or segment. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Flags: *Size: 2 Bits*

The Flag Bits consist of the Packet_Boundary_Flag and Broadcast_Flag. The Packet_Boundary_Flag is located in bit 4 and bit 5, and the Broadcast_Flag is located in bit 6 and 7 in the second octet of the HCI ACL Data packet.

Packet_Boundary_Flag: *Size: 2 Bits*

Value	Parameter Description
00	Reserved for future use
01	Continuing fragment packet of Higher Layer Message
10	First packet of Higher Layer Message (i.e. start of an L2CAP packet)
11	Reserved for future use

Broadcast_Flag (in packet from Host to Controller): *Size: 2 Bits*

Value	Parameter Description
00	No broadcast. Only point-to-point.
01	Active Slave Broadcast: packet is sent to all active slaves (i.e. packet is usually not sent during park beacon slots), and it may be received by slaves in sniff mode or park state.
10	Parked Slave Broadcast: packet is sent to all slaves and all slaves in park state (i.e. packet is sent during park beacon slots if there are parked slaves), and it may be received by slaves in sniff mode.
11	Reserved for future use.

Broadcast_Flag (in packet from Controller to Host): *Size: 2 Bits*

Value	Parameter Description
00	Point-to-point
01	Packet received as a slave not in park state (either Active Slave Broadcast or Parked Slave Broadcast)
10	Packet received as a slave in park state (Parked Slave Broadcast)
11	Reserved for future use.

Note: active slave broadcast packets may be sent in park beacon slots.

Note: slaves in sniff mode may or may not receive a broadcast packet depending on whether they happen to be listening at sniff slots, when the packet is sent.

Data_Total_Length: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Length of data measured in octets.

5.4.3 HCI Synchronous Data Packets

HCI synchronous (SCO and eSCO) Data Packets are used to exchange synchronous data between the Host and Controller. The format of the synchronous Data Packet is shown in Figure 5.3. The definition for each of the fields in the data packets is explained below.

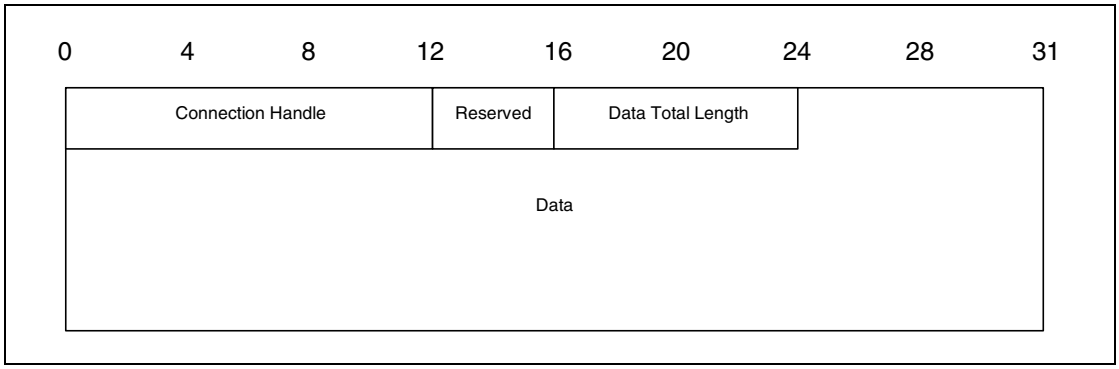


Figure 5.3: HCI Synchronous Data Packet

Connection_Handle: *Size: 12 Bits*

Value	Parameter Description
0xXXX	Connection handle to be used to for transmitting a synchronous data packet or segment. Range: 0x0000-0x0EFF (0x0F00- 0x0FFF Reserved for future use)

The Reserved Bits consist of four bits which are located from bit 4 to bit 7 in the second octet of the HCI Synchronous Data packet.

Reserved: *Size: 4 Bits*

Value	Parameter Description
XXXX	Reserved for future use.

Data_Total_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	Length of synchronous data measured in octets

5.4.4 HCI Event Packet

The HCI Event Packet is used by the Controller to notify the Host when events occur. The Host must be able to accept HCI Event Packets with up to 255 octets of data excluding the HCI Event Packet header. The format of the HCI Event Packet is shown in Figure 5.4, and the definition of each field is explained below.

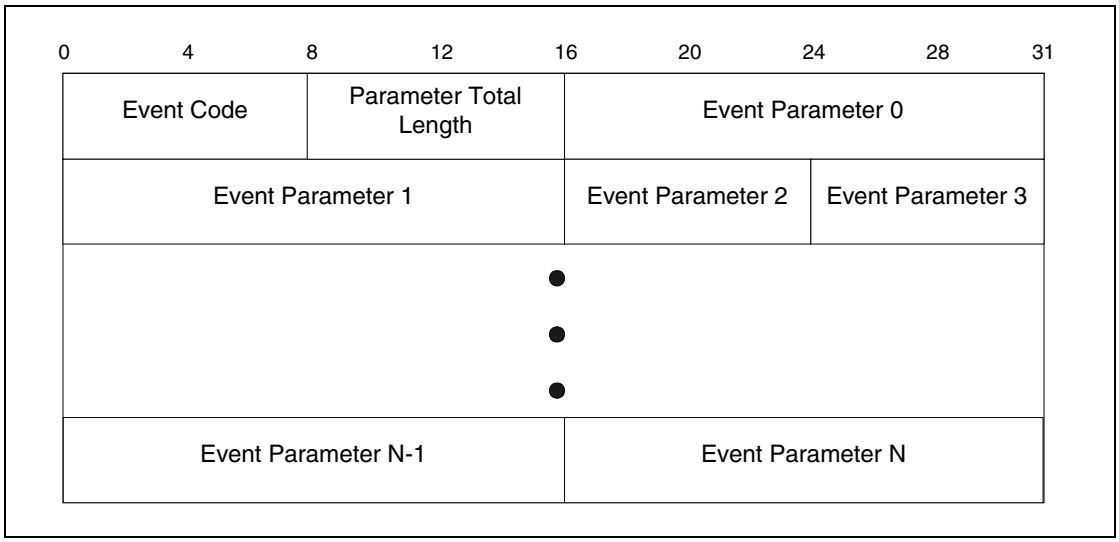


Figure 5.4: HCI Event Packet

Event_Code: *Size: 1 Octet*

Value	Parameter Description
0xXX	Each event is assigned a 1-Octet event code used to uniquely identify different types of events. Range: 0x00-0xFF (The event code 0xFF is reserved for the event code used for vendor-specific debug events. In addition, the event code 0xFE is also reserved for Bluetooth Logo Testing)

Parameter_Total_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	Length of all of the parameters contained in this packet, measured in octets

Event_Parameter 0 - N: *Size: Parameter Total Length*

Value	Parameter Description
0xXX	Each event has a specific number of parameters associated with it. These parameters and the size of each of the parameters are defined for each event. Each parameter is an integer number of octets in size.

6 HCI CONFIGURATION PARAMETERS

6.1 SCAN ENABLE

The Scan_Enable parameter controls whether or not the Bluetooth device will periodically scan for page attempts and/or inquiry requests from other Bluetooth devices. If Page_Scan is enabled, then the device will enter page scan mode based on the value of the Page_Scan_Interval and Page_Scan_Window parameters. If Inquiry_Scan is enabled, then the device will enter Inquiry Scan mode based on the value of the Inquiry_Scan_Interval and Inquiry_Scan_Window parameters.

Value	Parameter Description
0x00	No Scans enabled.
0x01	Inquiry Scan enabled. Page Scan always disabled.
0x02	Inquiry Scan disabled. Page Scan enabled.
0x03	Inquiry Scan enabled. Page Scan enabled.
0x04-0xFF	Reserved

6.2 INQUIRY SCAN INTERVAL

The Inquiry_Scan_Interval configuration parameter defines the amount of time between consecutive inquiry scans. This is defined as the time interval from when the Controller started its last inquiry scan until it begins the next inquiry scan.

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0012 to 0x1000; only even values are valid Default: 0x1000 Mandatory Range: 0x0012 to 0x1000 Time = N * 0.625 msec Time Range: 11.25 to 2560 msec Time Default: 2.56 sec

6.3 INQUIRY SCAN WINDOW

The Inquiry_Scan_Window configuration parameter defines the amount of time for the duration of the inquiry scan. The Inquiry_Scan_Window can only be less than or equal to the Inquiry_Scan_Interval.

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0011 to 0x1000 Default: 0x0012 Mandatory Range: 0x0011 to Inquiry Scan Interval Time = N * 0.625 msec Time Range: 10.625 msec to 2560 msec Time Default: 11.25 msec

6.4 INQUIRY SCAN TYPE

The Inquiry_Scan_Type configuration parameter indicates whether inquiry scanning will be done using non-interlaced scan or interlaced scan. Currently, one mandatory inquiry scan type and one optional inquiry scan type are defined. For details, see the Baseband Specification, “Inquiry scan substate” on page 164 [Part B].

Value	Parameter Description
0x00	Mandatory: Standard Scan (default)
0x01	Optional: Interlaced Scan
0x02-0xFF	Reserved

6.5 INQUIRY MODE

The Inquiry_Mode configuration parameter indicates whether inquiry returns Inquiry Result events in the standard format or with RSSI.

Value	Parameter Description
0x00	Standard Inquiry Result event format
0x01	Inquiry Result format with RSSI
0x02-0xFF	Reserved

6.6 PAGE TIMEOUT

The Page_Timeout configuration parameter defines the maximum time the local Link Manager will wait for a baseband page response from the remote device at a locally initiated connection attempt. If this time expires and the remote device has not responded to the page at baseband level, the connection attempt will be considered to have failed.

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0001 to 0xFFFF Default: 0x2000 Mandatory Range: 0x0016 to 0xFFFF Time = N * 0.625 msec Time Range: 0.625 msec to 40.9 sec Time Default: 5.12 sec

6.7 CONNECTION ACCEPT TIMEOUT

The Connection_Accept_Timeout configuration parameter allows the Bluetooth hardware to automatically deny a connection request after a specified time period has occurred and the new connection is not accepted. The parameter defines the time duration from when the Controller sends a Connection Request event until the Controller will automatically reject an incoming connection.

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0001 to 0xB540 Default: 0x1F40 Mandatory Range: 0x00A0 to 0xB540 Time = N * 0.625 msec Time Range: 0.625 msec to 29 sec Time Default: 5 sec

6.8 PAGE SCAN INTERVAL

The Page_Scan_Interval configuration parameter defines the amount of time between consecutive page scans. This time interval is defined from when the Controller started its last page scan until it begins the next page scan.

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0012 to 0x1000; only even values are valid Default: 0x0800 Mandatory Range: 0x0012 to 0x1000 Time = N * 0.625 msec Time Range: 11.25 msec to 2560 msec Time Default: 1.28 sec

6.9 PAGE SCAN WINDOW

The Page_Scan_Window configuration parameter defines the amount of time for the duration of the page scan. The Page_Scan_Window can only be less than or equal to the Page_Scan_Interval.

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0011 to 0x1000 Default: 0x0012 Mandatory Range: 0x0011 to Page Scan Interval Time = N * 0.625 msec Time Range: 10.625 msec to Page Scan Interval Time Default: 11.25 msec

6.10 PAGE SCAN PERIOD MODE (DEPRECATED)

Every time an inquiry response message is sent, the Bluetooth device will start a timer (T_mandatory_pscan), the value of which is dependent on the Page_Scan_Period_Mode. As long as this timer has not expired, the Bluetooth device will use the mandatory page scan mode for all following page scans. Note: the timer T_mandatory_pscan will be reset at each new inquiry response. For details see the “Baseband Specification” on page 55 [Part B]. (Keyword: SP-Mode, FHS-Packet, T_mandatory_pscan, Inquiry-Response).

Value	Parameter Description
0x00	P0
0x01	P1
0x02	P2
0x03-0xFF	Reserved.

6.11 PAGE SCAN TYPE

The Page_Scan_Type parameter indicates whether inquiry scanning will be done using non-interlaced scan or interlaced scan. For details, see the Baseband Specification, “Page scan substate” on page 154 [Part B].

Value	Parameter Description
0x00	Mandatory: Standard Scan (default)
0x01	Optional: Interlaced Scan
0x02-0xFF	Reserved

6.12 VOICE SETTING

The Voice_Setting parameter controls all the various settings for voice connections. The input settings apply to all voice connections, and **cannot** be set for individual voice connections. The Voice_Setting parameter controls the configuration for voice connections: Input Coding, Air coding format, input data format, Input sample size, and linear PCM parameter.The air coding format bits in the Voice_Setting command parameter specify which air coding format the local device requests. The air coding format bits do not specify which air coding format(s) the local device accepts when a remote device requests an air coding format. This is determined by the hardware capabilities of the local device.

Value	Parameter Description
00XXXXXXXX	Input Coding: Linear
01XXXXXXXX	Input Coding: μ-law Input Coding
10XXXXXXXX	Input Coding: A-law Input Coding
11XXXXXXXX	Reserved for Future Use
XX00XXXXXX	Input Data Format: 1’s complement
XX01XXXXXX	Input Data Format: 2’s complement
XX10XXXXXX	Input Data Format: Sign-Magnitude
XX11XXXXXX	Input Data Format: Unsigned
XXXX0XXXXX	Input Sample Size: 8-bit (only for linear PCM)
XXXX1XXXXX	Input Sample Size: 16-bit (only for linear PCM)
XXXXXnnnXX	Linear_PCM_Bit_Pos: # bit positions that MSB of sample is away from starting at MSB (only for Linear PCM).
XXXXXXXX00	Air Coding Format: CVSD
XXXXXXXX01	Air Coding Format: μ-law
XXXXXXXX10	Air Coding Format: A-law
XXXXXXXX11	Air Coding Format: Transparent Data

6.13 PIN TYPE

The PIN Type configuration parameter determines whether the Link Manager assumes that the Host supports variable PIN codes or a fixed PIN code. The host controller uses the PIN-type information during pairing.

Value	Parameter Description
0x00	Variable PIN.
0x01	Fixed PIN.

6.14 LINK KEY

The Controller can store a limited number of link keys for other Bluetooth devices. Link keys are shared between two Bluetooth devices, and are used for all security transactions between the two devices. A Host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the Bluetooth Controller when needed. A Link Key is associated with a BD_ADDR.

Value	Parameter Description
0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX	Link Key for an associated BD_ADDR.

6.15 AUTHENTICATION ENABLE

The Authentication_Enable parameter controls if the local device requires to authenticate the remote device at connection setup (between the Create_Connection command or acceptance of an incoming ACL connection and the corresponding Connection Complete event). At connection setup, only the device(s) with the Authentication_Enable parameter enabled will try to authenticate the other device.

Note: Changing this parameter does not affect existing connections.

Value	Parameter Description
0x00	Authentication not required.
0x01	Authentication required for all connections.
0x02-0xFF	Reserved

6.16 ENCRYPTION MODE

The Encryption_Mode parameter controls if the local device requires encryption to the remote device at connection setup (between the Create_Connection command or acceptance of an incoming ACL connection and the corresponding Connection Complete event). At connection setup, only devices with the Authentication_Enabled configuration parameter set to required and the Encryption_Mode configuration parameter set to required will try to encrypt the physical link to the other device.

Note: Changing this parameter does not affect existing connections.

A temporary link key is used when both broadcast and point-to-point traffic are encrypted.

The Host must not specify the Encryption_Mode parameter with more encryption capability than its local device currently supports, although the parameter is used to request the encryption capability to the remote device. Note that the Host must not request the command with the Encryption_Mode parameter set to 0x01, when the local device does not support encryption.

Note: for encryption to be used, both devices must support and enable encryption.

Value	Parameter Description
0x00	Encryption not required.
0x01	Encryption required for all connections.
0x02-0xFF	Reserved.

Note: in the Connection Complete event the Encryption_Mode parameter will show whether encryption was successfully turned on. The remote device may not support encryption or may have set Encryption_Mode to 0x01 when the local device has not, so the encryption mode returned in the Connection Complete event may not equal the encryption mode set in the HCI_Write_Encryption_Mode Command.

6.17 FAILED CONTACT COUNTER

The Failed_Contact_Counter records the number of consecutive incidents in which either the slave or master didn't respond after the flush timeout had expired, and the L2CAP packet that was currently being transmitted was automatically 'flushed'. When this occurs, the Failed_Contact_Counter is incremented by 1. The Failed_Contact_
Counter for a connection is reset to zero on the following conditions:

1. When a new connection is established
2. When the Failed_Contact_Counter is > zero and an L2CAP packet is acknowledged for that connection
3. When the Reset_Failed_Contact_Counter command has been issued

Value	Parameter Description
0xXXXX	Number of consecutive failed contacts for a connection corresponding to the connection handle.

6.18 HOLD MODE ACTIVITY

The Hold_Mode_Activity value is used to determine what activities should be suspended when the device is in hold mode. After the hold period has expired, the device will return to the previous mode of operation. Multiple hold mode activities may be specified for the Hold_Mode_Activity parameter by performing a bitwise OR operation of the different activity types. If no activities are suspended, then all of the current Periodic Inquiry, Inquiry Scan, and Page Scan settings remain valid during the Hold Mode. If the Hold_Mode_Activity parameter is set to Suspend Page Scan, Suspend Inquiry Scan, and Suspend Periodic Inquiries, then the device can enter a low-power state during the Hold Mode period, and all activities are suspended. Suspending multiple activities can be specified for the Hold_Mode_Activity parameter by performing a bitwise OR operation of the different activity types.The Hold Mode Activity is only valid if all connections are in Hold Mode.

Value	Parameter Description
0x00	Maintain current Power State.
0x01	Suspend Page Scan.
0x02	Suspend Inquiry Scan.
0x04	Suspend Periodic Inquiries.
0x08-0xFF	Reserved for Future Use.

6.19 LINK POLICY SETTINGS

The Link_Policy_Settings parameter determines the behavior of the local Link Manager when it receives a request from a remote device or it determines itself to change the master-slave role or to enter park state, hold, or sniff mode. The local Link Manager will automatically accept or reject such a request from the remote device, and may even autonomously request itself, depending on the value of the Link_Policy_Settings parameter for the corresponding Connection_Handle. When the value of the Link_Policy_Settings parameter is changed for a certain Connection_Handle, the new value will only be used for requests from a remote device or from the local Link Manager itself made after this command has been completed. By enabling each mode individually, the Host can choose any combination needed to support various modes of operation. Multiple LM policies may be specified for the Link_Policy_Settings parameter by performing a bitwise OR operation of the different activity types.

Note: The local device may be forced into hold mode (regardless of whether the local device is master or slave) by the remote device regardless of the value of the Link_Policy_Settings parameter. The forcing of hold mode can however only be done once the connection has already been placed into hold mode through an LMP request (the Link_Policy_Settings determine if requests from a remote device should be accepted or rejected). The forcing of hold mode can after that be done as long as the connection lasts regardless of the setting for hold mode in the Link_Policy_Settings parameter.

Note that the previous description implies that if the implementation in the remote device is a "polite" implementation that does not force another device into hold mode via LMP PDUs, then the Link_Policy_Settings will never be overruled.

Value	Parameter Description
0x0000	Disable All LM Modes Default
0x0001	Enable Role switch.
0x0002	Enable Hold Mode.
0x0004	Enable Sniff Mode.
0x0008	Enable Park State.
0x0010 – 0x8000	Reserved for Future Use.

6.20 FLUSH TIMEOUT

The Flush_Timeout configuration parameter is used for ACL connections only. The Flush Timeout is defined in the Baseband specification Section 7.6.3, “Flushing payloads,” on page 150. This parameter allows ACL packets to be automatically flushed without the Host device issuing a Flush command. This provides support for isochronous data, such as audio. When the L2CAP packet that is currently being transmitted is automatically ‘flushed’, the Failed Contact Counter is incremented by one.

Value	Parameter Description
0	Timeout = ∞; No Automatic Flush
N = 0xXXXX	Size: 2 Octets Range: 0x0001 to 0x07FF Mandatory Range: 0x0002 to 0x07FF Time = N * 0.625 msec Time Range: 0.625 msec to 1279.375 msec

6.21 NUM BROADCAST RETRANSMISSIONS

Broadcast packets are not acknowledged and are unreliable. The Number of Broadcast Retransmissions parameter, N, is used to increase the reliability of a broadcast message by retransmitting the broadcast message multiple times. This sets the value N_{BC} in the baseband to one greater than the Num Broadcast Retransmissions value. (See Baseband Specification, Section 7.6.5, on page 150) This parameter should be adjusted as the link quality measurement changes.

Value	Parameter Description
N = 0xXX	N_{BC} = N + 1 Range 0x00-0xFE

6.22 LINK SUPERVISION TIMEOUT

The Link_Supervision_Timeout parameter is used by the master or slave Bluetooth device to monitor link loss. If, for any reason, no Baseband packets are received from that Connection Handle for a duration longer than the Link_Supervision_Timeout, the connection is disconnected. The same timeout value is used for both synchronous and ACL connections for the device specified by the Connection Handle.

Note: Setting the Link_Supervision_Timeout to No Link_Supervision_Timeout (0x0000) will disable the Link_Supervision_Timeout check for the specified Connection Handle. This makes it unnecessary for the master of the piconet to unpark and then park each Bluetooth Device every ~40 seconds. By using the No Link_Supervision_Timeout setting, the scalability of the Park state is not limited.

Value	Parameter Description
0x0000	No Link_Supervision_Timeout.
N = 0xXXXX	Size: 2 Octets Range: 0x0001 to 0xFFFF Default: 0x7D00 Mandatory Range: 0x0190 to 0xFFFF Time = N * 0.625 msec Time Range: 0.625 msec to 40.9 sec Time Default: 20 sec

6.23 SYNCHRONOUS FLOW CONTROL ENABLE

The Synchronous Flow Control Enable configuration parameter allows the Host to decide if the Controller will send Number Of Completed Packets events for synchronous Connection Handles. This setting allows the Host to enable and disable synchronous flow control.

Value	Parameter Description
0x00	Synchronous Flow Control is disabled. No Number of Completed Packets events will be sent from the Controller for synchronous Connection Handles.
0x01	Synchronous Flow Control is enabled. Number of Completed Packets events will be sent from the Controller for synchronous Connection Handles.

6.24 LOCAL NAME

The user-friendly Local Name provides the user the ability to distinguish one Bluetooth device from another. The Local Name configuration parameter is a UTF-8 encoded string with up to 248 octets in length. The Local Name configuration parameter will be null terminated (0x00) if the UTF-8 encoded string is less than 248 octets.

Note: the Local Name configuration parameter is a string parameter. Endianess does therefore not apply to the Local Name configuration parameter. The first octet of the name is received first.

Value	Parameter Description
	A UTF-8 encoded User Friendly Descriptive Name for the device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.

6.25 CLASS OF DEVICE

The Class_of_Device parameter is used to indicate the capabilities of the local device to other devices.

Value	Parameter Description
0xXXXXXX	Class of Device for the device.

6.26 SUPPORTED COMMANDS

The Supported Commands configuration parameter lists which HCI commands the local controller supports. It is implied that if a command is listed as supported, the feature underlying that command is also supported.

The Supported Commands is a 64 octet bit field. If a bit is set to 1, then this command is supported.

Octet	Bit	Command Supported
0	0	Inquiry
	1	Inquiry Cancel
	2	Periodic Inquiry Mode
	3	Exit Periodic Inquiry Mode
	4	Create Connection
	5	Disconnect
	6	Add SCO Connection
	7	Cancel Create Connection
1	0	Accept Connection Request
	1	Reject Connection Request
	2	Link Key Request Reply
	3	Link Key Request Negative Reply
	4	PIN Code Request Reply
	5	PIN Code Request Negative Reply
	6	Change Connection Packet Type
	7	Authentication Request
2	0	Set Connection Encryption
	1	Change Connection Link Key
	2	Master Link Key
	3	Remote Name Request
	4	Cancel Remote Name Request
	5	Read Remote Supported Features
	6	Read Remote Extended Features
	7	Read Remote Version Information
3	0	Read Clock Offset
	1	Read LMP Handle
	2	Reserved
	3	Reserved
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

Octet	Bit	Command Supported
4	0	Reserved
	1	Hold Mode
	2	Sniff Mode
	3	Exit Sniff Mode
	4	Park State
	5	Exit Park State
	6	QoS Setup
	7	Role Discovery
5	0	Switch Role
	1	Read Link Policy Settings
	2	Write Link Policy Settings
	3	Read Default Link Policy Settings
	4	Write Default Link Policy Settings
	5	Flow Specification
	6	Set Event Mark
	7	Reset
6	0	Set Event Filter
	1	Flush
	2	Read PIN Type
	3	Write PIN Type
	4	Create New Unit Key
	5	Read Stored Link Key
	6	Write Stored Link Key
	7	Delete Stored Link Key
7	0	Write Local Name
	1	Read Local Name
	2	Read Connection Accept Timeout
	3	Write Connection Accept Timeout
	4	Read Page Timeout
	5	Write Page Timeout
	6	Read Scan Enable
	7	Write Scan Enable
8	0	Read Page Scan Activity
	1	Write Page Scan Activity
	2	Read Inquiry Scan Activity
	3	Write Inquiry Scan Activity
	4	Read Authentication Enable
	5	Write Authentication Enable
	6	Read Encryption Mode
	7	Write Encryption Mode

Octet	Bit	Command Supported
9	0	Read Class Of Device
	1	Write Class Of Device
	2	Read Voice Setting
	3	Write Voice Setting
	4	Read Automatic Flush Timeout
	5	Write Automatic Flush Timeout
	6	Read Num Broadcast Retransmissions
	7	Write Num Broadcast Retransmissions
10	0	Read Hold Mode Activity
	1	Write Hold Mode Activity
	2	Read Transmit Power Level
	3	Read Synchronous Flow Control Enable
	4	Write Synchronous Flow Control Enable
	5	Set Host Controller To Host Flow Control
	6	Host Buffer Size
	7	Host Number Of Completed Packets
11	0	Read Link Supervision Timeout
	1	Write Link Supervision Timeout
	2	Read Number of Supported IAC
	3	Read Current IAC LAP
	4	Write Current IAC LAP
	5	Reserved
	6	Reserved
	7	Read Page Scan Mode
12	0	Write Page Scan Mode
	1	Set AFH Channel Classification
	2	reserved
	3	reserved
	4	Read Inquiry Scan Type
	5	Write Inquiry Scan Type
	6	Read Inquiry Mode
	7	Write Inquiry Mode
13	0	Read Page Scan Type
	1	Write Page Scan Type
	2	Read AFH Channel Assessment Mode
	3	Write AFH Channel Assessment Mode
	4	Reserved
	5	Reserved
	6	Reserved
	7	Reserved

Octet	Bit	Command Supported
14	0	Reserved
	1	Reserved
	2	Reserved
	3	Read Local Version Information
	4	Reserved
	5	Read Local Supported Features
	6	Read Local Extended Features
	7	Read Buffer Size
15	0	Read Country Code
	1	Read BD ADDR
	2	Read Failed Contact Count
	3	Reset Failed Contact Count
	4	Get Link Quality
	5	Read RSSI
	6	Read AFH Channel Map
	7	Read BD Clock
16	0	Read Loopback Mode
	1	Write Loopback Mode
	2	Enable Device Under Test Mode
	3	Setup Synchronous Connection
	4	Accept Synchronous Connection
	5	Reject Synchronous Connection[
	6	Reserved
	7	Reserved

7 HCI COMMANDS AND EVENTS

7.1 LINK CONTROL COMMANDS

The Link Control commands allow the Controller to control connections to other Bluetooth devices. When the Link Control commands are used, the Link Manager (LM) controls how the Bluetooth piconets and scatternets are established and maintained. These commands instruct the LM to create and modify link layer connections with Bluetooth remote devices, perform Inquiries of other Bluetooth devices in range, and other LMP commands. For the Link Control commands, the OGF is defined as 0x01.

7.1.1 Inquiry Command

Command	OCF	Command Parameters	Return Parameters
HCI_Inquiry	0x0001	LAP, Inquiry_Length, Num_Responses	

Description:

This command will cause the Bluetooth device to enter Inquiry Mode. Inquiry Mode is used to discover other nearby Bluetooth devices. The LAP input parameter contains the LAP from which the inquiry access code shall be derived when the inquiry procedure is made. The Inquiry_Length parameter specifies the total duration of the Inquiry Mode and, when this time expires, Inquiry will be halted. The Num_Responses parameter specifies the number of responses that can be received before the Inquiry is halted. A Command Status event is sent from the Controller to the Host when the Inquiry command has been started by the Bluetooth device. When the Inquiry process is completed, the Controller will send an Inquiry Complete event to the Host indicating that the Inquiry has finished. The event parameters of Inquiry Complete event will have a summary of the result from the Inquiry process, which reports the number of nearby Bluetooth devices that responded. When a Bluetooth device responds to the Inquiry message, an Inquiry Result event will occur to notify the Host of the discovery.

A device which responds during an inquiry or inquiry period should always be reported to the Host in an Inquiry Result event if the device has not been reported earlier during the current inquiry or inquiry period and the device has not been filtered out using the command Set_Event_Filter. If the device has been reported earlier during the current inquiry or inquiry period, it may or may not be reported depending on the implementation (depending on if earlier results have been saved in the Controller and in that case how many responses that have been saved). It is recommended that the Controller tries to report a particular device only once during an inquiry or inquiry period.

Command Parameters:

LAP: *Size: 3 Octets*

Value	Parameter Description
0x9E8B00– 0X9E8B3F	This is the LAP from which the inquiry access code should be derived when the inquiry procedure is made; see “Bluetooth Assigned Numbers” (https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers).

Inquiry_Length: *Size: 1 Octet*

Value	Parameter Description
N = 0xXX	Maximum amount of time specified before the Inquiry is halted. Size: 1 octet Range: 0x01 – 0x30 Time = N * 1.28 sec Range: 1.28 – 61.44 Sec

Num_Responses: *Size: 1 Octet*

Value	Parameter Description
0x00	Unlimited number of responses.
0xXX	Maximum number of responses from the Inquiry before the Inquiry is halted. Range: 0x01 – 0xFF

Return Parameters:

None.

Event(s) generated (unless masked away):

A Command Status event is sent from the Controller to the Host when the Controller has started the Inquiry process. An Inquiry Result event will be created for each Bluetooth device which responds to the Inquiry message. In addition, multiple Bluetooth devices which respond to the Inquire message may be combined into the same event. An Inquiry Complete event is generated when the Inquiry process has completed.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Inquiry Complete event will indicate that this command has been completed. No Inquiry Complete event will be generated for the canceled Inquiry process.

7.1.2 Inquiry Cancel Command

Command	OCF	Command Parameters	Return Parameters
HCI_Inquiry_Cancel	0x0002		Status

Description:

This command will cause the Bluetooth device to stop the current Inquiry if the Bluetooth device is in Inquiry Mode. This command allows the Host to interrupt the Bluetooth device and request the Bluetooth device to perform a different task. The command should only be issued after the Inquiry command has been issued, a Command Status event has been received for the Inquiry command, and before the Inquiry Complete event occurs.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Inquiry_Cancel command succeeded.
0x01-0xFF	Inquiry_Cancel command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Inquiry Cancel command has completed, a Command Complete event will be generated. No Inquiry Complete event will be generated for the canceled Inquiry process.

7.1.3 Periodic Inquiry Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Periodic_ Inquiry_Mode	0x0003	Max_Period_Length, Min_Period_Length, LAP, Inquiry_Length, Num_Responses	Status

Description:

The Periodic_Inquiry_Mode command is used to configure the Bluetooth device to enter the Periodic Inquiry Mode that performs an automatic Inquiry. Max_Period_Length and Min_Period_Length define the time range between two consecutive inquiries, from the beginning of an inquiry until the start of the next inquiry. The Controller will use this range to determine a new random time between two consecutive inquiries for each Inquiry. The LAP input parameter contains the LAP from which the inquiry access code shall be derived when the inquiry procedure is made. The Inquiry_Length parameter specifies the total duration of the InquiryMode and, when time expires, Inquiry will be halted. The Num_Responses parameter specifies the number of responses that can be received before the Inquiry is halted. This command is completed when the Inquiry process has been started by the Bluetooth device, and a Command Complete event is sent from the Controller to the Host. When each of the periodic Inquiry processes are completed, the Controller will send an Inquiry Complete event to the Host indicating that the latest periodic Inquiry process has finished. When a Bluetooth device responds to the Inquiry message an Inquiry Result event will occur to notify the Host of the discovery.
Note: Max_Period_Length > Min_Period_Length > Inquiry_Length

A device which responds during an inquiry or inquiry period should always be reported to the Host in an Inquiry Result event if the device has not been reported earlier during the current inquiry or inquiry period and the device has not been filtered out using the command Set_Event_Filter. If the device has been reported earlier during the current inquiry or inquiry period, it may or may not be reported depending on the implementation (depending on if earlier results have been saved in the Controller and in that case how many responses that have been saved). It is recommended that the Controller tries to report a particular device only once during an inquiry or inquiry period.

Command Parameters:

Max_Period_Length: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Maximum amount of time specified between consecutive inquiries. Size: 2 octets Range: 0x03 – 0xFFFF Time = N * 1.28 sec Range: 3.84 – 83884.8 Sec 0.0 – 23.3 hours

Min_Period_Length: *Size: 2 Octets*

Value	**Parameter Description**
N = 0xXXXX	Minimum amount of time specified between consecutive inquiries. Size: 2 octets Range: 0x02 – 0xFFFE Time = N * 1.28 sec Range: 2.56 – 83883.52 Sec 0.0 – 23.3 hours

LAP: *Size: 3 Octets*

Value	**Parameter Description**
0x9E8B00– 0X9E8B3F	This is the LAP from which the inquiry access code should be derived when the inquiry procedure is made, see “Bluetooth Assigned Numbers” (https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers).

Inquiry_Length: *Size: 1 Octet*

Value	**Parameter Description**
N = 0xXX	Maximum amount of time specified before the Inquiry is halted. Size: 1 octet Range: 0x01 – 0x30 Time = N * 1.28 sec Range: 1.28 – 61.44 Sec

Num_Responses: *Size: 1 Octet*

Value	**Parameter Description**
0x00	Unlimited number of responses.
0xXX	Maximum number of responses from the Inquiry before the Inquiry is halted. Range: 0x01 – 0xFF

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Periodic Inquiry Mode command succeeded.
0x01-0xFF	Periodic Inquiry Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

The Periodic Inquiry Mode begins when the Controller sends the Command Complete event for this command to the Host. An Inquiry Result event will be created for each Bluetooth device which responds to the Inquiry message. In addition, multiple Bluetooth devices which response to the Inquiry message may be combined into the same event. An Inquiry Complete event is generated when each of the periodic Inquiry processes has completed. No Inquiry Complete event will be generated for the canceled Inquiry process.

7.1.4 Exit Periodic Inquiry Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Exit_Periodic_Inquiry_Mode	0x0004		Status

Description:

The Exit Periodic Inquiry Mode command is used to end the Periodic Inquiry mode when the local device is in Periodic Inquiry Mode. If the local device is currently in an Inquiry process, the Inquiry process will be stopped directly and the Controller will no longer perform periodic inquiries until the Periodic Inquiry Mode command is reissued.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Exit Periodic Inquiry Mode command succeeded.
0x01-0xFF	Exit Periodic Inquiry Mode command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

A Command Complete event for this command will occur when the local device is no longer in Periodic Inquiry Mode. No Inquiry Complete event will be generated for the canceled Inquiry process.

7.1.5 Create Connection Command

Command	OCF	Command Parameters	Return Parameters
HCI_Create_Connection	0x0005	BD_ADDR, Packet_Type, Page_Scan_Repetition_Mode, Reserved, Clock_Offset, Allow_Role_Switch	

Description:

This command will cause the Link Manager to create a connection to the Bluetooth device with the BD_ADDR specified by the command parameters. This command causes the local Bluetooth device to begin the Page process to create a link level connection. The Link Manager will determine how the new ACL connection is established. This ACL connection is determined by the current state of the device, its piconet, and the state of the device to be connected. The Packet_Type command parameter specifies which packet types the Link Manager shall use for the ACL connection. When sending HCI ACL Data Packets the Link Manager shall only use the packet type(s) specified by the Packet_Type command parameter or the always-allowed DM1 packet type. Multiple packet types may be specified for the Packet Type parameter by performing a bit-wise OR operation of the different packet types. The Link Manager may choose which packet type to be used from the list of acceptable packet types. The Page_Scan_Repetition_Mode parameter specifies the page scan repetition mode supported by the remote device with the BD_ADDR. This is the information that was acquired during the inquiry process. The Clock_Offset parameter is the difference between its own clock and the clock of the remote device with BD_ADDR. Only bits 2 through 16 of the difference are used, and they are mapped to this parameter as bits 0 through 14 respectively. A Clock_Offset_Valid_Flag, located in bit 15 of the Clock_Offset parameter, is used to indicate if the Clock Offset is valid or not. A Connection handle for this connection is returned in the Connection Complete event (see below). The Allow_Role_Switch parameter specifies if the local device accepts or rejects the request of a master-slave role switch when the remote device requests it at the connection setup (in the Role parameter of the Accept_Connection_Request command) (before the local Controller returns a Connection Complete event). For a definition of the different packet types see the “Baseband Specification” on page 55 [Part B].

Note: The Host should enable as many packet types as possible for the Link Manager to perform efficiently. However, the Host must not enable packet types that the local device does not support.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device to be connected.

Packet_Type: *Size: 2 Octets*

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	2-DH1 may not be used.
0x0004	3-DH1 may not be used.
0x0008[1]	DM1 may be used.
0x0010	DH1 may be used.
0x0020	Reserved for future use.
0x0040	Reserved for future use.
0x0080	Reserved for future use.
0x0100	2-DH3 may not be used.
0x0200	3-DH3 may not be used.
0x0400	DM3 may be used.
0x0800	DH3 may be used.
0x1000	2-DH5 may not be used.
0x2000	3-DH5 may not be used.
0x4000	DM5 may be used.
0x8000	DH5 may be used.

1. This bit will be interpreted as set to 1 by Bluetooth V1.2 or later controllers.

Page_Scan_Repetition_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03 – 0xFF	Reserved.

Reserved: *Size: 1 Octet*

Value	Parameter Description
0x00	Reserved, must be set to 0x00. See “Page Scan Mode” on page 612.

Clock_Offset: *Size: 2 Octets*

Bit format	Parameter Description
Bit 14-0	Bit 16-2 of CLKslave-CLKmaster.
Bit 15	Clock_Offset_Valid_Flag Invalid Clock Offset = 0 Valid Clock Offset = 1

Allow_Role_Switch: *Size: 1 Octet*

Value	Parameter Description
0x00	The local device will be a master, and will not accept a role switch requested by the remote device at the connection setup.
0x01	The local device may be a master, or may become a slave after accepting a role switch requested by the remote device at the connection setup.
0x02-0xFF	Reserved for future use.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Create Connection command, the Controller sends the Command Status event to the Host. In addition, when the LM determines the connection is established, the Controller, on both Bluetooth devices that form the connection, will send a Connection Complete event to each Host. The Connection Complete event contains the Connection Handle if this command is successful.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

7.1.6 Disconnect Command

Command	OCF	Command Parameters	Return Parameters
HCI_Disconnect	0x0006	Connection_Handle, Reason	

Description:

The Disconnection command is used to terminate an existing connection. The Connection_Handle command parameter indicates which connection is to be disconnected. The Reason command parameter indicates the reason for ending the connection. The remote Bluetooth device will receive the Reason command parameter in the Disconnection Complete event. All synchronous connections on a physical link should be disconnected before the ACL connection on the same physical connection is disconnected.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle for the connection being disconnected. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Reason: *Size: 1 Octet*

Value	Parameter Description
0x05, 0x13-0x15, 0x1A, 0x29	Authentication Failure error code (0x05), Other End Terminated Connection error codes (0x13-0x15), Unsupported Remote Feature error code (0x1A) and Pairing with Unit Key Not Supported error code (0x29), see "Error Codes" on page 319 [Part D] for list of Error Codes.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Disconnect command, it sends the Command Status event to the Host. The Disconnection Complete event will occur at each Host when the termination of the connection has completed, and indicates that this command has been completed.
Note: No Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Disconnection Complete event will indicate that this command has been completed.

7.1.7 Create Connection Cancel Command

Command	OCF	Command Parameters	Return Parameters
HCI_Create_Connection _Cancel	0x0008	BD_ADDR	Status, BD_ADDR

Description:

This command is used to request cancellation of the ongoing connection creation process, which was started by a Create_Connection command of the local device.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Create Connection command request that was issued before and is subject of this cancellation request

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Create Connection Cancel command succeeded
0x01-0xff	Create Connection Cancel command failed. See “Error Codes” on page 319 [Part D] for list of error codes

BD_ADDR: *Size: 6 Octet*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Create Connection command that was issued before and is the subject of this cancellation request.

Event(s) generated (unless masked away):

When the Create Connection Cancel command has completed, a Command Complete event shall be generated.

If the connection is already established by the baseband, but the Controller has not yet sent the Connection Complete event, then the local device shall detach the link and return a Command Complete event with the status “Success”.

If the connection is already established, and the Connection Complete event has been sent, then the Controller shall return a Command Complete event with the error code *ACL Connection already exists* (0x0B).

If the Create Connection Cancel command is sent to the Controller without a preceding Create Connection command to the same device, the Controller shall return a Command Complete event with the error code *Unknown Connection Identifier* (0x02).

The Connection Complete event for the corresponding Create Connection Command shall always be sent. The Connection Complete event shall be sent after the Command Complete event for the Create Connection Cancel command. If the cancellation was successful, the Connection Complete event will be generated with the error code *Unknown Connection Identifier* (0x02).

7.1.8 Accept Connection Request Command

Command	OCF	Command Parameters	Return Parameters
HCI_Accept_Connection Request	0x0009	BD_ADDR, Role	

Description:

The Accept_Connection_Request command is used to accept a new incoming connection request. The Accept_Connection_Request command shall only be issued after a Connection Request event has occurred. The Connection Request event will return the BD_ADDR of the device which is requesting the connection. This command will cause the Link Manager to create a connection to the Bluetooth device, with the BD_ADDR specified by the command parameters. The Link Manager will determine how the new connection will be established. This will be determined by the current state of the device, its piconet, and the state of the device to be connected. The Role command parameter allows the Host to specify if the Link Manager shall request a role switch and become the Master for this connection. This is a preference and not a requirement. If the Role Switch fails then the connection will still be accepted, and the Role Discovery Command will reflect the current role.
Note: The Link Manager may terminate the connection if it would be low on resources if the role switch fails. The decision to accept a connection must be completed before the connection accept timeout expires on the local Bluetooth Module.
Note: when accepting synchronous connection request, the Role parameter is not used and will be ignored by the Controller.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device to be connected

Role: *Size: 1 Octet*

Value	Parameter Description
0x00	Become the Master for this connection. The LM will perform the role switch.
0x01	Remain the Slave for this connection. The LM will **NOT** perform the role switch.

Return Parameters:

None.

Event(s) generated (unless masked away):

The Accept_Connection_Request command will cause the Command Status event to be sent from the Controller when the Controller begins setting up the connection. In addition, when the Link Manager determines the connection is established, the local Controller will send a Connection Complete event to its Host, and the remote Controller will send a Connection Complete event or a Synchronous Connection Complete event to the Host. The Connection Complete event contains the Connection Handle if this command is successful. Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

7.1.9 Reject Connection Request Command

Command	OCF	Command Parameters	Return Parameters
HCI_Reject_Connection_Request	0x000A	BD_ADDR, Reason	

Description:

The Reject_Connection_Request command is used to decline a new incoming connection request. The Reject_Connection_Request command shall only be called after a Connection Request event has occurred. The Connection Request event will return the BD_ADDR of the device that is requesting the connection. The Reason command parameter will be returned to the connecting device in the Status parameter of the Connection Complete event returned to the Host of the connection device, to indicate why the connection was declined.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device to reject the connection from.

Reason: *Size: 1 Octet*

Value	Parameter Description
0x0D-0x0F	Host Reject Error Code. See “Error Codes” on page 319 [Part D] for list of Error Codes and descriptions.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Reject_Connection_Request command, the Controller sends the Command Status event to the Host. Then, the local Controller will send a Connection Complete event to its host, and the remote Controller will send a Connection Complete event or a Synchronous Connection Complete event to the host. The Status parameter of the Connection Complete event, which is sent to the Host of the device attempting to make the connection, will contain the Reason command parameter from this command.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

7.1.10 Link Key Request Reply Command

Command	OCF	Command Parameters	Return Parameters
HCI_Link_Key_Request_Reply	0x000B	BD_ADDR, Link_Key	Status, BD_ADDR

Description:

The Link_Key_Request_Reply command is used to reply to a Link Key Request event from the Controller, and specifies the Link Key stored on the Host to be used as the link key for the connection with the other Bluetooth Device specified by BD_ADDR. The Link Key Request event will be generated when the Controller needs a Link Key for a connection.

When the Controller generates a Link Key Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a Link_Key_Request_Reply or Link_Key_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 211 [Part C].)

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device of which the Link Key is for.

Link_Key: *Size: 16 Octets*

Value	Parameter Description
0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX	Link Key for the associated BD_ADDR.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Link_Key_Request_Reply command succeeded.
0x01-0xFF	Link_Key_Request_Reply command failed. See "Error Codes" on page 319 [Part D] for error codes and description.

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device of which the Link Key request reply has completed.

Event(s) generated (unless masked away):

The Link_Key_Request_Reply command will cause a Command Complete event to be generated.

7.1.11 Link Key Request Negative Reply Command

Command	OCF	Command Parameters	Return Parameters
HCI_Link_Key_Request_Negative_Reply	0x000C	BD_ADDR	Status, BD_ADDR

Description:

The Link_Key_Request_Negative_Reply command is used to reply to a Link Key Request event from the Controller if the Host does not have a stored Link Key for the connection with the other Bluetooth Device specified by BD_ADDR. The Link Key Request event will be generated when the Controller needs a Link Key for a connection.

When the Controller generates a Link Key Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a Link_Key_Request_Reply or Link_Key_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 211 [Part C].)

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which the Link Key is for.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Link_Key_Request_Negative_Reply command succeeded.
0x01-0xFF	Link_Key_Request_Negative_Reply command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which the Link Key request negative reply has completed.

Event(s) generated (unless masked away):

The Link_Key_Request_Negative_Reply command will cause a Command Complete event to be generated.

7.1.12 PIN Code Request Reply Command

Command	OCF	Command Parameters	Return Parameters
HCI_PIN_Code_Request_Reply	0x000D	BD_ADDR, PIN_Code_Length, PIN_Code	Status, BD_ADDR

Description:

The PIN_Code_Request_Reply command is used to reply to a PIN Code request event from the Controller, and specifies the PIN code to use for a connection. The PIN Code Request event will be generated when a connection with remote initiating device has requested pairing.

When the Controller generates a PIN Code Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a PIN_Code_Request_Reply or PIN_Code_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See “Link Manager Protocol” on page 211 [Part C].)

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which the PIN code is for.

PIN_Code_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	The PIN code length specifics the length, in octets, of the PIN code to be used. Range: 0x01-0x10

PIN_Code: *Size: 16 Octets*

Value	Parameter Description
0xXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX	PIN code for the device that is to be connected. The Host should insure that strong PIN Codes are used. PIN Codes can be up to a maximum of 128 bits. Note: the PIN_Code Parameter is a string parameter. Endianess does therefore not apply to the PIN_Code Parameter. The first octet of the PIN code should be transmitted first.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	PIN_Code_Request_Reply command succeeded.
0x01-0xFF	PIN_Code_Request_Reply command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which the PIN Code request reply has completed.

Event(s) generated (unless masked away):

The PIN_Code_Request_Reply command will cause a Command Complete event to be generated.

7.1.13 PIN Code Request Negative Reply Command

Command	OCF	Command Parameters	Return Parameters
HCI_PIN_Code_ Request_Negative Reply	0x000E	BD_ADDR	Status, BD_ADDR

Description:

The PIN_Code_Request_Negative_Reply command is used to reply to a PIN Code request event from the Controller when the Host cannot specify a PIN code to use for a connection. This command will cause the pair request with remote device to fail.

When the Controller generates a PIN Code Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a PIN_Code_Request_Reply or PIN_Code_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See "Link Manager Protocol" on page 211 [Part C].)

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which this command is responding to.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	PIN_Code_Request_Negative_Reply command succeeded.
0x01-0xFF	PIN_Code_Request_Negative_Reply command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which the PIN Code request negative reply has completed.

Event(s) generated (unless masked away):

The PIN_Code_Request_Negative_Reply command will cause a Command Complete event to be generated.

7.1.14 Change Connection Packet Type Command

Command	OCF	Command Parameters	Return Parameters
HCI_Change_Connection_ Packet_Type	0x000F	Connection_Handle, Packet_Type	

Description:

The Change_Connection_Packet_Type command is used to change which packet types can be used for a connection that is currently established. This allows current connections to be dynamically modified to support different types of user data. The Packet_Type command parameter specifies which packet types the Link Manager can use for the connection. When sending HCI ACL Data Packets the Link Manager shall only use the packet type(s) specified by the Packet_Type command parameter or the always-allowed DM1 packet type. The interpretation of the value for the Packet_Type command parameter will depend on the Link_Type command parameter returned in the Connection Complete event at the connection setup. Multiple packet types may be specified for the Packet_Type command parameter by bitwise OR operation of the different packet types. For a definition of the different packet types see the “Baseband Specification” on page 55 [Part B].

Note: The Host should enable as many packet types as possible for the Link Manager to perform efficiently. However, the Host must not enable packet types that the local device does not support.

Note: to change an eSCO connection, use the Setup Synchronous Connection command.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to for transmitting and receiving voice or data. Returned from creating a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Packet_Type: *Size: 2 Octets*

For ACL Link_Type

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	2-DH1 may not be used.
0x0004	3-DH1 may not be used.
0x0008[1]	DM1 may be used.
0x0010	DH1 may be used.
0x0020	Reserved for future use.
0x0040	Reserved for future use.
0x0080	Reserved for future use.
0x0100	2-DH3 may not be used.
0x0200	3-DH3 may not be used.
0x0400	DM3 may be used.
0x0800	DH3 may be used.
0x1000	2-DH5 may not be used.
0x2000	3-DH5 may not be used.
0x4000	DM5 may be used.
0x8000	DH5 may be used.

1. This bit will be interpreted as set to 1 by Bluetooth V1.2 or later controllers.

For SCO Link Type

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	Reserved for future use.
0x0004	Reserved for future use.
0x0008	Reserved for future use.
0x0010	Reserved for future use.
0x0020	HV1
0x0040	HV2
0x0080	HV3
0x0100	Reserved for future use.
0x0200	Reserved for future use.
0x0400	Reserved for future use.
0x0800	Reserved for future use.
0x1000	Reserved for future use.
0x2000	Reserved for future use.
0x4000	Reserved for future use.
0x8000	Reserved for future use.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Change Connection Packet Type command, the Controller sends the Command Status event to the Host. In addition, when the Link Manager determines the packet type has been changed for the connection, the Controller on the local device will send a Connection Packet Type Changed event to the Host. This will be done at the local side only.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Connection Packet Type Changed event will indicate that this command has been completed.

7.1.15 Authentication Requested Command

Command	OCF	Command Parameters	Return Parameters
HCI_Authentication_ Requested	0x0011	Connection_Handle	

Description:

The Authentication_Requested command is used to try to authenticate the remote device associated with the specified Connection Handle. On an authentication failure, the Controller or Link Manager shall not automatically detach the link. The Host is responsible for issuing a Disconnect command to terminate the link if the action is appropriate.
Note**:** the Connection_Handle command parameter is used to identify the other Bluetooth device, which forms the connection. The Connection Handle should be a Connection Handle for an ACL connection.

Command Parameters:

Connection_Handle: *Size 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to set up authentication for all Connection Handles with the same Bluetooth device end-point as the specified Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Authentication_Requested command, it sends the Command Status event to the Host. The Authentication Complete event will occur when the authentication has been completed for the connection and is indication that this command has been completed.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Authentication Complete event will indicate that this command has been completed.

Note: When the local or remote Controller does not have a link key for the specified Connection_Handle, it will request the link key from its Host, before the local Host finally receives the Authentication Complete event.

7.1.16 Set Connection Encryption Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_Connection_ Encryption	0x0013	Connection_Handle, Encryption_Enable	

Description:

The Set_Connection_Encryption command is used to enable and disable the link level encryption. Note**:** the Connection_Handle command parameter is used to identify the other Bluetooth device which forms the connection. The Connection Handle should be a Connection Handle for an ACL connection. While the encryption is being changed, all ACL traffic must be turned off for all Connection Handles associated with the remote device.

Command Parameters:

Connection_Handle: *Size 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to enable/disable the link layer encryption for all Connection Handles with the same Bluetooth device end-point as the specified Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Encryption_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	Turn Link Level Encryption OFF.
0x01	Turn Link Level Encryption ON.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Set_Connection_Encryption command, the Controller sends the Command Status event to the Host. When the Link Manager has completed enabling/disabling encryption for the connection, the Controller on the local Bluetooth device will send an Encryption Change event to the Host, and the Controller on the remote device will also generate an Encryption Change event.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Encryption Change event will indicate that this command has been completed.

7.1.17 Change Connection Link Key Command

Command	OCF	Command Parameters	Return Parameters
HCI_Change_Connection_ Link_Key	0x0015	Connection_Handle	

Description:

The Change_Connection_Link_Key command is used to force both devices of a connection associated with the connection handle to generate a new link key. The link key is used for authentication and encryption of connections.
Note: the Connection_Handle command parameter is used to identify the other Bluetooth device forming the connection. The Connection Handle should be a Connection Handle for an ACL connection.

Command Parameters:

Connection_Handle: *Size 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Change_Connection_Link_Key command, the Controller sends the Command Status event to the Host. When the Link Manager has changed the Link Key for the connection, the Controller on the local Bluetooth device will send a Link Key Notification event and a Change Connection Link Key Complete event to the Host, and the Controller on the remote device will also generate a Link Key Notification event. The Link Key Notification event indicates that a new connection link key is valid for the connection.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Change Connection Link Key Complete event will indicate that this command has been completed.

7.1.18 Master Link Key Command

Command	OCF	Command Parameters	Return Parameters
HCI_Master_Link_Key	0x0017	Key_Flag	

Description:

The Master Link Key command is used to force the device that is master of the piconet to use the temporary link key of the master device, or the semi-permanent link keys. The temporary link key is used for encryption of broadcast messages within a piconet, and the semi-permanent link keys are used for private encrypted point-to-point communication. The Key_Flag command parameter is used to indicate which Link Key (temporary link key of the Master, or the semi-permanent link keys) shall be used.

Command Parameters:

Key_Flag: *Size: 1 Octet*

Value	Parameter Description
0x00	Use semi-permanent Link Keys.
0x01	Use Temporary Link Key.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Master_Link_Key command, the Controller sends the Command Status event to the Host. When the Link Manager has changed link key, the Controller on both the local and the remote device will send a Master Link Key Complete event to the Host. The Connection Handle on the master side will be a Connection Handle for one of the existing connections to a slave. On the slave side, the Connection Handle will be a Connection Handle to the initiating master.

The Master Link Key Complete event contains the status of this command. Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Master Link Key Complete event will indicate that this command has been completed.

7.1.19 Remote Name Request Command

Command	OCF	Command Parameters	Return Parameters
HCI_Remote_Name_Request	0x0019	BD_ADDR, Page_Scan_Repetition_Mode, Reserved, Clock_Offset	

Description:

The Remote_Name_Request command is used to obtain the user-friendly name of another Bluetooth device. The user-friendly name is used to enable the user to distinguish one Bluetooth device from another. The BD_ADDR command parameter is used to identify the device for which the user-friendly name is to be obtained. The Page_Scan_Repetition_Mode parameter specifies the page scan repetition mode supported by the remote device with the BD_ADDR. This is the information that was acquired during the inquiry process. The Clock_Offset parameter is the difference between its own clock and the clock of the remote device with BD_ADDR. Only bits 2 through 16 of the difference are used and they are mapped to this parameter as bits 0 through 14 respectively. A Clock_Offset_Valid_Flag, located in bit 15 of the Clock_Offset command parameter, is used to indicate if the Clock Offset is valid or not. Note: if no connection exists between the local device and the device corresponding to the BD_ADDR, a temporary link layer connection will be established to obtain the name of the remote device.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the device whose name is requested.

Page_Scan_Repetition_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03 – 0xFF	Reserved.

Reserved: *Size: 1 Octet*

Value	Parameter Description
0x00	Reserved, must be set to 0x00. See “Page Scan Mode” on page 612.

Clock_Offset: *Size: 2 Octets*

Bit format	Parameter Description
Bit 14.0	Bit 16.2 of CLKslave-CLKmaster.
Bit 15	Clock_Offset_Valid_Flag Invalid Clock Offset = 0 Valid Clock Offset = 1

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Remote_Name_Request command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to obtain the remote name, the Controller on the local Bluetooth device will send a Remote Name Request Complete event to the Host. Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, only the Remote Name Request Complete event will indicate that this command has been completed.

7.1.20 Remote Name Request Cancel Command

Command	OCF	Command Parameters	Return Parameters
HCI_Remote_Name_Request_Cancel	0x001A	BD_ADDR	Status, BD_ADDR

Description:

This command is used to request cancellation of the ongoing remote name request process, which was started by the Remote Name Request command.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Remote Name Request command that was issued before and that is subject of this cancellation request

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Remote Name Request Cancel command succeeded
0x01-0xff	Remote Name Request Cancel command failed. See “Error Codes” on page 319 [Part D] for list of error codes

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Remote Name Request Cancel command that was issued before and that was subject of this cancellation request

Event(s) generated (unless masked away):

When the Remote Name Request Cancel command has completed, a Command Complete event shall be generated.

If the Remote Name Request Cancel command is sent to the Controller without a preceding Remote Name Request command to the same device, the Controller will return a Command Complete event with the error code *Invalid HCI Command Parameters* (0x12).

The Remote Name Request Complete event for the corresponding Remote Name Request Command shall always be sent. The Remote Name Request

Complete event shall be sent after the Command Complete event for the Remote Name Request Cancel command. If the cancellation was successful, the Remote Name Request Complete event will be generated with the error code *Unknown Connection Identifier (0x02)*.

7.1.21 Read Remote Supported Features Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Remote_Supported_Features	0x001B	Connection_Handle	

Description:

This command requests a list of the supported features for the remote device identified by the Connection_Handle parameter. The Connection_Handle must be a Connection_Handle for an ACL connection. The Read Remote Supported Features Complete event will return a list of the LMP features. For details see “Link Manager Protocol” on page 211 [Part C].

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle’s LMP-supported features list to get. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Read_Remote_Supported_Features command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to determine the remote features, the Controller on the local Bluetooth device will send a Read Remote Supported Features Complete event to the Host. The Read Remote Supported Features Complete event contains the status of this command, and parameters describing the supported features of the remote device. Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Read Remote Supported Features Complete event will indicate that this command has been completed.

7.1.22 Read Remote Extended Features Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Remote_Extended_Features	0x001C	Connection_Handle, Page Number	

Description:

The HCI_Read_Remote_Extended_Features command returns the requested page of the extended LMP features for the remote device identified by the specified connection handle. The connection handle must be the connection handle for an ACL connection. This command is only available if the extended features feature is implemented by the remote device. The Read Remote Extended Features Complete event will return the requested information.
For details see "Link Manager Protocol" on page 211 [Part C].

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The connection handle identifying the remote device for which extended feature information is required. Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use)

Page Number: *Size: 1 Octet*

Value	Parameter Description
0x00	Requests the normal LMP features as returned by HCI_Read_Remote_Supported_Features
0x01-0xFF	Return the corresponding page of features

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the HCI_Read_Remote_Extended_Features command the Controller sends the Command Status command to the Host. When the Link Manager has completed the LMP sequence to determine the remote extended features the controller on the local device will generate a Read Remote Extended Features Complete event to the host. The Read Remote Extended Features Complete event contains the page number and the remote features returned by the remote device.

Note: no Command Complete event will ever be sent by the Controller to indicate that this command has been completed. Instead the Read Remote Extended Features Complete event will indicate that this command has been completed.

7.1.23 Read Remote Version Information Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Remote_Version_ Information	0x001D	Connection_Handle	

Description:

This command will obtain the values for the version information for the remote Bluetooth device identified by the Connection_Handle parameter. The Connection_Handle must be a Connection_Handle for an ACL connection.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's version information to get. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Read_Remote_Version_Information command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to determine the remote version information, the Controller on the local Bluetooth device will send a Read Remote Version Information Complete event to the Host. The Read Remote Version Information Complete event contains the status of this command, and parameters describing the version and subversion of the LMP used by the remote device.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Read Remote Version Information Complete event will indicate that this command has been completed.

7.1.24 Read Clock Offset Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Clock_Offset	0x001F	Connection_Handle	

Description:

Both the System Clock and the clock offset to a remote device are used to determine what hopping frequency is used by a remote device for page scan. This command allows the Host to read clock offset to remote devices. The clock offset can be used to speed up the paging procedure when the local device tries to establish a connection to a remote device, for example, when the local Host has issued Create_Connection or Remote_Name_Request. The Connection_Handle must be a Connection_Handle for an ACL connection.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Clock Offset parameter is returned. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Read_Clock_Offset command, the Controller sends the Command Status event to the Host. If this command was requested at the master and the Link Manager has completed the LMP messages to obtain the Clock Offset information, the Controller on the local Bluetooth device will send a Read Clock Offset Complete event to the Host.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, only the Read Clock Offset Complete event will indicate that this command has been completed. If the command is requested at the slave, the LM will immediately send a Command Status event and a Read Clock Offset Complete event to the Host, without an exchange of LMP PDU.

7.1.25 Read LMP Handle Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_LMP_Handle	0x0020	Connection_Handle	Status, Connection_Handle, LMP_Handle, Reserved

Description:

This command will read the current LMP Handle associated with the Connection_Handle. The Connection_Handle must be a SCO or eSCO Handle. If the Connection_Handle is a SCO connection handle, then this command shall read the LMP SCO Handle for this connection. If the Connection_Handle is an eSCO connection handle, then this command shall read the LMP eSCO Handle for this connection.

Command Parameters:

Connection_Handle: *Size 2 Octets (12 bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify which connection to be used for reading the LMP Handle. This must be a synchronous handle. Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_LMP_Handle command succeeded.
0x01 – 0xFF	Read_LMP_Handle command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the LMP_Handle has been read. Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

LMP_Handle: *Size: 1 Octet*

Value	Parameter Description
0xXX	The LMP Handle is the LMP Handle that is associated with this connection handle. For a synchronous handle, this would be the LMP Synchronous Handle used when negotiating the synchronous connection in the link manager.

Reserved: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	This parameter is reserved, must to set to zero.

Events(s) generated (unless masked away):

When the Read_LMP_Handle command has completed, a Command Complete event will be generated.

7.1.26 Setup Synchronous Connection Command

Command	OCF	Command Parameters	Return Parameters
HCI_Setup_Synchronous_ Connection	0x0028	Connection_Handle Transmit_Bandwidth Receive_Bandwidth Max_Latency Voice_Setting Retransmission_Effort Packet Type	

Description:

The HCI Setup Synchronous Connection command adds a new or modifies an existing synchronous logical transport (SCO or eSCO) on a physical link depending on the Connection_Handle parameter specified. If the Connection_Handle refers to an ACL link a new synchronous logical transport will be added. If the Connection_Handle refers to an already existing synchronous logical transport (eSCO only) this link will be modified.The parameters are specified per connection. This synchronous connection can be used to transfer synchronous voice at 64kbps or transparent synchronous data.

When used to setup a new synchronous logical transport, the Connection_Handle parameter shall specify an ACL connection with which the new synchronous connection will be associated. The other parameters relate to the negotiation of the link, and may be reconfigured during the lifetime of the link. The transmit and receive bandwidth specify how much bandwidth shall be available for transmitting and for receiving data. While in many cases the receive and transmit bandwidth parameters may be equal, they may be different. The latency specifies an upper limit to the time in milliseconds between the eSCO (or SCO) instants, plus the size of the retransmission window, plus the length of the reserved synchronous slots for this logical transport. The content format specifies the settings for voice or transparent data on this connection. The retransmission effort specifies the extra resources that are allocated to this connection if a packet may need to be retransmitted. The Retransmission_Effort parameter shall be set to indicate the required behavior, or to don't care.

When used to modify an existing synchronous logical transport, the Transmit_Bandwidth, Receive_Bandwidth and Voice_Settings shall be set to the same values as were used during the initial setup. The Packet_Type, Retransmission_Effort and Max_Latency parameters may be modified.

The Packet_Type field is a bitmap specifying which packet types the LM shall accept in the negotiation of the link parameters. Multiple packet types are specified by bitwise OR of the packet type codes in the table. At least one packet type must be specified for each negotiation. It is recommended to enable as many packet types as possible. Note that it is allowed to enable packet types that are not supported by the local device.

A connection handle for the new synchronous connection will be returned in the synchronous connection complete event.

Note: The link manager may choose any combination of packet types, timing, and retransmission window sizes that satisfy the parameters given. This may be achieved by using more frequent transmissions of smaller packets. The link manager may choose to set up either a SCO or an eSCO connection, if the parameters allow, using the corresponding LMP sequences.

Note: To modify a SCO connection, use the Change Connection Packet Type command.

Note: If the lower layers cannot achieve the exact transmit and receive bandwidth requested subject to the other parameters, then the link shall be rejected.

A synchronous connection may only be created when an ACL connection already exists and when it is not in park state.

Command Parameters:

Connection_Handle: *2 octets (12 bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle for the ACL connection being used to create a synchronous Connection or for the existing Connection that shall be modified. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Transmit_Bandwidth: *4 octets*

Value	Parameter Description
0xXXXXXXXX	Transmit bandwidth in octets per second.

Receive_Bandwidth: *4 octets*

Value	Parameter Description
0xXXXXXXXX	Receive bandwidth in octets per second.

Max_Latency: *2 octets*

Value	Parameter Description
0x0000-0x0003	Reserved
0x0004-0xFFFE	This is a value in milliseconds representing the upper limit of the sum of the synchronous interval, the size of the eSCO window. (See Figure 8.7 in the Baseband specification)
0xFFFF	Don't care.

Voice_Setting: *2 octets (10 bits meaningful)*

Value	Parameter Description
See Section 6.12 on page 387.	

Retransmission_Effort: *1 octet*

Value	Parameter Description
0x00	No retransmissions
0x01	At least one retransmission, optimize for power consumption.
0x02	At least one retransmission, optimize for link quality
0xFF	Don’t care
0x03 – 0xFE	Reserved

Packet_Type: *2 octets*

Value	Parameter Description
0x0001	HV1 may be used.
0x0002	HV2 may be used.
0x0004	HV3 may be used.
0x0008	EV3 may be used.
0x0010	EV4 may be used.
0x0020	EV5 may be used.
0x0040	2-EV3 may not be used.
0x0080	3-EV3 may not be used.
0x0100	2-EV5 may not be used.
0x0200	3-EV5 may not be used.
0x0400	Reserved for future use
0x0800	Reserved for future use
0x1000	Reserved for future use
0x2000	Reserved for future use
0x4000	Reserved for future use
0x8000	Reserved for future use

Return Parameters:

None

Event(s) generated (unless masked away)

When the Controller receives the Setup_Synchronous_Connection command, it sends the Command Status event to the Host. In addition, when the LM determines the connection is established, the local Controller will send a Synchronous Connection Complete event to the local Host, and the remote Controller will send a Synchronous Connection Complete event or a Connection Complete event to the remote Host. The synchronous Connection Complete event contains the Connection Handle if this command is successful.

If this command is used to change the parameters of an existing eSCO link, the Synchronous Connection Changed Event is sent to both hosts. In this case no Connection Setup Complete Event or Connection Request Event will be sent to either host. This command cannot be used to change the parameters of an SCO link.

Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the synchronous Connection Complete event will indicate that this command has been completed.

7.1.27 Accept Synchronous Connection Request Command

Command	OCF	Command Parameters	Return Parameters
HCI_Accept_ Synchronous_ Connection_Request	0x0029	BD_ADDR Transmit_Bandwidth Receive_Bandwidth Max_Latency Content_Format Retransmission_Effort Packet_Type	

Description:

The Accept_Synchronous_Connection_Request command is used to accept an incoming request for a synchronous connection and to inform the local Link Manager about the acceptable parameter values for the synchronous connection. The Command shall only be issued after a Connection_Request event with link type SCO or eSCO has occurred. Connection_Request event contains the BD_ADDR of the device requesting the connection. The decision to accept a connection must be taken before the connection accept timeout expires on the local device.

The parameter set of the Accept_Synchronous_Connection_Request command is the same as for the Setup_Synchronous_Connection command. The Transmit_Bandwidth and Receive_Bandwidth values are required values for the new link and shall be met. The Max_Latency is an upper bound to the acceptable latency for the Link, as defined in Section 7.1.26 on page 437 Setup_Synchronous_Connection and shall not be exceeded. Content_Format specifies the encoding in the same way as in the Setup_Synchronous_Connection command and shall be met. The Retransmission_Effort parameter shall be set to indicate the required behavior, or to don't care. The Packet_Type parameter is a bit mask specifying the synchronous packet types that are allowed on the link and shall be met. The reserved bits in the Packet_Type field shall be set to one. If all bits are set in the packet type field then all packets types shall be allowed.

If the Link Type of the incoming request is SCO, then only the Transmit_Bandwidth, Max_Latency, Content_Format, and Packet_Type fields are valid.

If the Connection_Request event is masked away, and the Controller is not set to auto-accept this connection attempt, the Controller will automatically reject it. If the controller is set to automatically accept the connection attempt, the LM should assume default parameters. In that case the Synchronous_Connection_Complete Event shall be generated, unless masked away.

Command Parameters:

BD_ADDR: *6 octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the device requesting the connection

Transmit_Bandwidth: *4 octets*

Value	Parameter Description
0x00000000-0xFFFFFFFE	Maximum possible transmit bandwidth in octets per second.
0xFFFFFFFF	Don’t care

Default: Don’t care

Receive_Bandwidth: *4 octets*

Value	Parameter Description
0x00000000-0xFFFFFFFE	Maximum possible receive bandwidth in octets per second.
0xFFFFFFFF	Don’t care

Default: Don’t care

Max_Latency: *2 octets*

Value	Parameter Description
0x0000-0x0003	Reserved
0x0004-0xFFFE	This is a value in milliseconds representing the upper limit of the sum of the synchronous interval and the size of the eSCO window.
0xFFFF	Don't care.

Default: Don't care

Content_Format: *2 octets (10 bits meaningful)*

Value	Parameter Description
00XXXXXXXX	Input Coding: Linear
01XXXXXXXX	Input Coding: u-law
10XXXXXXXX	Input Coding: A-law
11XXXXXXXX	Reserved for future use.
XX00XXXXXX	Input Data Format: 1's complement
XX01XXXXXX	Input Data Format: 2's complement
XX10XXXXXX	Input Data Format: Sign-Magnitude
XX11XXXXXX	Input Data Format: Unsigned
XXXX0XXXXX	Input Sample Size: 8 bit (only for Linear PCM)
XXXX1XXXXX	Input Sample Size: 16 bit (only for Linear PCM)
XXXXXnnnXX	Linear PCM Bit Position: number of bit positions that MSB of sample is away from starting at MSB (only for Linear PCM)
XXXXXXXX00	Air Coding Format: CVSD
XXXXXXXX01	Air Coding Format: u-law
XXXXXXXX10	Air Coding Format: A-law
XXXXXXXX11	Air Coding Format: Transparent Data

Default: When links are auto-accepted, the values written by the HCI_Write_Voice_Settings are used.

Retransmission_Effort: *1 octet*

Value	Parameter Description
0x00	No retransmissions
0x01	At least one retransmission, optimize for power consumption.
0x02	At least one retransmission, optimize for link quality.
0x03-0xFE	Reserved
0xFF	Don't care

Default: Don't care

Packet_Type: *2 octets*

Value	Parameter Description
0x0001	HV1 may be used.
0x0002	HV2 may be used.
0x0004	HV3 may be used.
0x0008	EV3 may be used.
0x0010	EV4 may be used.
0x0020	EV5 may be used.
0x0040	2-EV3 may not be used.
0x0080	3-EV3 may not be used
0x0100	2-EV5 may not be used.
0x0200	3-EV5 may not be used.
0x0400	Reserved for future use
0x0800	Reserved for future use
0x1000	Reserved for future use
0x2000	Reserved for future use
0x4000	Reserved for future use
0x8000	Reserved for future use

Default: 0xFFFF - means all packet types may be used.

Return Parameters:

None

Event(s) generated (unless masked away):

The Accept_Synchronous_Request command will cause the Command Status event to be sent from the Host Controller when the Host Controller starts setting up the connection. When the link setup is complete, the local Controller will send a Synchronous Connection Complete event to its Host, and the remote Controller will send a Connection Complete event or a Synchronous Connection Complete event to the Host. The Synchronous Connection Complete will contain the connection handle and the link parameters if the setup is successful. No Command Complete event will be sent by the host controller as the result of this command.

7.1.28 Reject Synchronous Connection Request Command

Command	OCF	Command Parameters	Return Parameters
HCI_Reject_Synchronous_ Connection_Request	0x002A	BD_ADDR Reason	

Description:

The Reject_Synchronous_Connection_Request is used to decline an incoming request for a synchronous link. It shall only be issued after a Connection Request Event with Link Type equal to SCO or eSCO has occurred. The Connection Request Event contains the BD_ADDR of the device requesting the connection. The Reason parameter will be returned to the initiating host in the Status parameter of the Synchronous connection complete event on the remote side.

Command Parameters:

BD_ADDR: *6 octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the device requesting the connection

Reason: *1 octet*

Value	Parameter Description
0x0D-0x0F	Host Reject Error Code. See “Error Codes” on page 319 [Part D] for error codes and description.

Return Parameters:

None.

Event(s) Generated (unless masked away):

When the Host Controller receives the Reject_Synchronous_Connection_Request, it sends a Command Status Event to the Host. When the setup is terminated, the local Controller will send a Synchronous Connection Complete event to its Host, and the remote Controller will send a Connection Complete event or a Synchronous Connection Complete event to the Host with the Reason code from this command. No Command complete Event will be sent by the Host Controller to indicate that this command has been completed.

7.2 LINK POLICY COMMANDS

The Link Policy Commands provide methods for the Host to affect how the Link Manager manages the piconet. When Link Policy Commands are used, the LM still controls how Bluetooth piconets and scatternets are established and maintained, depending on adjustable policy parameters. These policy commands modify the Link Manager behavior that can result in changes to the link layer connections with Bluetooth remote devices.
Note: only one ACL connection can exist between two Bluetooth Devices, and therefore there can only be one ACL HCI Connection Handle for each physical link layer Connection. The Bluetooth Controller provides policy adjustment mechanisms to provide support for a number of different policies. This capability allows one Bluetooth module to be used to support many different usage models, and the same Bluetooth module can be incorporated in many different types of Bluetooth devices. For the Link Policy Commands, the OGF is defined as 0x02.

7.2.1 Hold Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Hold_Mode	0x0001	Connection_Handle, Hold_Mode_Max_Interval, Hold_Mode_Min_Interval	

Description:

The Hold_Mode command is used to alter the behavior of the Link Manager, and have it place the ACL baseband connection associated by the specified Connection Handle into the hold mode. The Hold_Mode_Max_Interval and Hold_Mode_Min_Interval command parameters specify the length of time the Host wants to put the connection into the hold mode. The local and remote devices will negotiate the length in the hold mode. The Hold_Mode_Max_ Interval parameter is used to specify the maximum length of the Hold interval for which the Host may actually enter into the hold mode after negotiation with the remote device. The Hold interval defines the amount of time between when the Hold Mode begins and when the Hold Mode is completed. The Hold_ Mode_Min_Interval parameter is used to specify the minimum length of the Hold interval for which the Host may actually enter into the hold mode after the negotiation with the remote device. Therefore the Hold_Mode_Min_Interval cannot be greater than the Hold_Mode_Max_Interval. The Controller will return the actual Hold interval in the Interval parameter of the Mode Change event, if the command is successful. This command enables the Host to support a low-power policy for itself or several other Bluetooth devices, and allows the devices to enter Inquiry Scan, Page Scan, and a number of other possible actions.
Note: the connection handle cannot be of the SCO or eSCO link type

If the Host sends data to the Controller with a Connection_Handle corresponding to a connection in hold mode, the Controller will keep the data in its buffers until either the data can be transmitted (the hold mode has ended) or a flush, a flush timeout or a disconnection occurs. This is valid even if the Host has not yet been notified of the hold mode through a Mode Change event when it sends the data.
Note: the above is not valid for an HCI Data Packet sent from the Host to the Controller on the master side where the Connection_Handle is a Connection_Handle used for broadcast and the Broadcast_Flag is set to Active Broadcast or Piconet Broadcast. The broadcast data will then never be received by slaves in hold mode.

The Hold_Mode_Max_Interval shall be less than the Link Supervision Timeout configuration parameter.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Hold_Mode_Max_Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Maximum acceptable number of Baseband slots to wait in Hold Mode. Time Length of the Hold = N * 0.625 msec (1 Baseband slot) Range for N: 0x0002-0xFFFE; only even values are valid. Time Range: 1.25ms - 40.9 sec Mandatory Range: 0x0014 to 0x8000

Hold_Mode_Min_Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Minimum acceptable number of Baseband slots to wait in Hold Mode. Time Length of the Hold = N * 0.625 msec (1 Baseband slot) Range for N: 0x0002-0xFF00; only even values are valid Time Range: 1.25 msec - 40.9 sec Mandatory Range: 0x0014 to 0x8000

Return Parameters:

None.

Event(s) generated (unless masked away):

The Controller sends the Command Status event for this command to the Host when it has received the Hold_Mode command. The Mode Change event will occur when the Hold Mode has started and the Mode Change event will occur again when the Hold Mode has completed for the specified connection handle. The Mode Change event signaling the end of the Hold Mode is an estimation of the hold mode ending if the event is for a remote Bluetooth device.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed. If an error occurs after the Command Status event has occurred, then the status in the Mode Change event will indicate the error.

7.2.2 Sniff Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Sniff_Mode	0x0003	Connection_Handle, Sniff_Max_Interval, Sniff_Min_Interval, Sniff_Attempt, Sniff_Timeout	

Description:

The Sniff Mode command is used to alter the behavior of the Link Manager and have it place the ACL baseband connection associated with the specified Connection Handle into the sniff mode. The Connection_Handle command parameter is used to identify which ACL link connection is to be placed in sniff mode. The Sniff_Max_Interval and Sniff_Min_Interval command parameters are used to specify the requested acceptable maximum and minimum periods in the Sniff Mode. The Sniff_Min_Interval shall not be greater than the Sniff_Max_Interval. The sniff interval defines the amount of time between each consecutive sniff period. The Controller will return the actual sniff interval in the Interval parameter of the Mode Change event, if the command is successful. For a description of the meaning of the Sniff_Attempt and Sniff_Timeout parameters, see Baseband Specification, Section 8.7, on page 183. Sniff_Attempt is there called $N_{sniff\ attempt}$ and Sniff_Timeout is called $N_{sniff\ timeout}$. This command enables the Host to support a low-power policy for itself or several other Bluetooth devices, and allows the devices to enter Inquiry Scan, Page Scan, and a number of other possible actions.
Note: in addition, the connection handle cannot be one of the synchronous link types. If the Host sends data to the Controller with a Connection_Handle corresponding to a connection in sniff mode, the Controller will keep the data in its buffers until either the data can be transmitted or a flush, a flush timeout or a disconnection occurs. This is valid even if the Host has not yet been notified of the sniff mode through a Mode Change event when it sends the data. Note that it is possible for the master to transmit data to a slave without exiting sniff mode

(see description in Baseband Specification, Section 8.7, on page 183).
Note: the above is not valid for an HCI Data Packet sent from the Host to the Controller on the master side where the Connection_Handle is a Connection_Handle used for broadcast and the Broadcast_Flag is set to Active Broadcast or Piconet Broadcast. In that case, the broadcast data will only be received by a slave in sniff mode if that slave happens to listen to the master when the broadcast is made.

The Sniff_Max_Interval shall be less than the Link Supervision Timeout configuration parameter.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Sniff_Max_Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Range: 0x0002 to 0xFFFE; only even values are valid Mandatory Range: 0x0006 to 0x0540 Time = N * 0.625 msec Time Range: 1.25 msec to 40.9 sec

Sniff_Min_Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Range: 0x0002 to 0xFFFE; only even values are valid Mandatory Range: 0x0006 to 0x0540 Time = N * 0.625 msec Time Range: 1.25 msec to 40.9 sec)

Sniff_Attempt: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Number of Baseband receive slots for sniff attempt. Length = N* 1.25 msec Range for N: 0x0001 – 0x7FFF Time Range: 0.625msec - 40.9 Seconds Mandatory Range for Controller: 1 to $T_{sniff}/2$

Sniff_Timeout: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Number of Baseband receive slots for sniff timeout. Length = N * 1.25 msec Range for N: 0x0000 – 0x7FFF Time Range: 0 msec - 40.9 Seconds Mandatory Range for Controller: 0 to 0x0028

Return Parameters:

None.

Event(s) generated (unless masked away):

The Controller sends the Command Status event for this command to the Host when it has received the Sniff_Mode command. The Mode Change event will occur when the Sniff Mode has started for the specified connection handle. Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead only the Mode Change event will indicate that this command has been completed. If an error occurs after the Command Status event has occurred, then the status in the Mode Change event will indicate the error.

7.2.3 Exit Sniff Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Exit_Sniff_Mode	0x0004	Connection_Handle	

Description:

The Exit_Sniff_Mode command is used to end the sniff mode for a connection handle, which is currently in sniff mode. The Link Manager will determine and issue the appropriate LMP commands to remove the sniff mode for the associated Connection Handle.
Note: in addition, the connection handle cannot be one of the synchronous link types.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

None.

Event(s) generated (unless masked away):

A Command Status event for this command will occur when Controller has received the Exit_Sniff_Mode command. The Mode Change event will occur when the Sniff Mode has ended for the specified connection handle.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed.

7.2.4 Park State Command

Command	OCF	Command Parameters	Return Parameters
HCI_Park_State	0x0005	Connection_Handle, Beacon_Max_Interval, Beacon_Min_Interval	

Description:

The Park State command is used to alter the behavior of the Link Manager, and have the LM place the baseband connection associated by the specified Connection Handle into Park state. The Connection_Handle command parameter is used to identify which connection is to be placed in Park state. The Connection_Handle must be a Connection_Handle for an ACL connection. The Beacon Interval command parameters specify the acceptable length of the interval between beacons. However, the remote device may request shorter interval. The Beacon_Max_Interval parameter specifies the acceptable longest length of the interval between beacons. The Beacon_Min_Interval parameter specifies the acceptable shortest length of the interval between beacons. Therefore, the Beacon Min Interval cannot be greater than the Beacon Max Interval. The Controller will return the actual Beacon interval in the Interval parameter of the Mode Change event, if the command is successful. This command enables the Host to support a low-power policy for itself or several other Bluetooth devices, allows the devices to enter Inquiry Scan, Page Scan, provides support for large number of Bluetooth Devices in a single piconet, and a number of other possible activities.
Note: when the Host issues the Park State command, no Connection Handles for synchronous connections are allowed to exist to the remote device that is identified by the Connection_Handle parameter. If one or more Connection Handles for synchronous connections exist to that device, depending on the implementation, a Command Status event or a Mode Change event (following a Command Status event where Status=0x00) will be returned with the error code 0x0C *Command Disallowed*.
If the Host sends data to the Controller with a Connection_Handle corresponding to a parked connection, the Controller will keep the data in its buffers until either the data can be transmitted (after unpark) or a flush, a flush timeout or a disconnection occurs. This is valid even if the Host has not yet been notified of park state through a Mode Change event when it sends the data.
Note: the above is not valid for an HCI Data Packet sent from the Host to the Controller on the master side where the Connection_Handle is a Connection_Handle used for Piconet Broadcast and the Broadcast_Flag is set to Piconet Broadcast. In that case, slaves in park state will also receive the broadcast data. (If the Broadcast_Flag is set to Active Broadcast, the broadcast data will usually not be received by slaves in park state.)
It is possible for the Controller to do an automatic unpark to transmit data and then park the connection again depending on the value of the Link_Policy_Settings parameter (see Write_Link_Policy_Settings) and depending on whether the imple-

mentation supports this or not (optional feature). The optional feature of automatic unpark/park can also be used for link supervision. Whether Mode Change events are returned or not at automatic unpark/park if this is implemented, is vendor specific. This could be controlled by a vendor specific HCI command.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Beacon_Max_Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Range: 0x000E to 0xFFFE; only even values are valid Mandatory Range: 0x000E to 0x1000 Time = N * 0.625 msec Time Range: 8.75 msec to 40.9 sec

Beacon_Min_Interval *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Range: 0x000E to 0xFFFE; only even values are valid Mandatory Range: 0x000E to 0x1000 Time = N * 0.625 msec Time Range: 8.75 msec to 40.9 sec

Return Parameters:

None.

Event(s) generated (unless masked away):

The Controller sends the Command Status event for this command to the Host when it has received the Park State command. The Mode Change event will occur when the Park State has started for the specified connection handle.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed. If an error occurs after the Command Status event has occurred, then the status in the Mode Change event will indicate the error.

7.2.5 Exit Park State Command

Command	OCF	Command Parameters	Return Parameters
HCI_Exit_Park_State	0x0006	Connection_Handle	

Description:

The Exit_Park_State command is used to switch the Bluetooth device from park state back to the active mode. This command may only be issued when the device associated with the specified Connection Handle is in park state. The Connection_Handle must be a Connection_Handle for an ACL connection. This function does not complete immediately.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

None.

Event(s) generated (unless masked away):

A Command Status event for this command will occur when the Controller has received the Exit_Park_State command. The Mode Change event will occur when park state has ended for the specified connection handle.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, only the Mode Change event will indicate that this command has been completed.

7.2.6 QoS Setup Command

Command	OCF	Command Parameters	Return Parameters
HCI_QoS_Setup	0x0007	Connection_Handle, Flags, Service_Type, Token_Rate, Peak_Bandwidth, Latency, Delay_Variation	

Description:

The QoS_Setup command is used to specify Quality of Service parameters for a connection handle. The Connection_Handle must be a Connection_Handle for an ACL connection. These QoS parameter are the same parameters as L2CAP QoS. For more detail see "Logical Link Control and Adaptation Protocol Specification" on page 15[vol. 4]. This allows the Link Manager to have all of the information about what the Host is requesting for each connection. The LM will determine if the QoS parameters can be met. Bluetooth devices that are both slaves and masters can use this command. When a device is a slave, this command will trigger an LMP request to the master to provide the slave with the specified QoS as determined by the LM. When a device is a master, this command is used to request a slave device to accept the specified QoS as determined by the LM of the master. The Connection_Handle command parameter is used to identify for which connection the QoS request is requested.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify which connection for the QoS Setup. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Flags: *Size: 1 Octet*

Value	Parameter Description
0x00 – 0xFF	Reserved for Future Use.

Service_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	No Traffic.
0x01	Best Effort.
0x02	Guaranteed.
0x03-0xFF	Reserved for Future Use.

Token_Rate: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Token Rate in octets per second.

Peak_Bandwidth: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Peak Bandwidth in octets per second.

Latency: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Latency in microseconds.

Delay_Variation: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Delay Variation in microseconds.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the QoS_Setup command, the Controller sends the Command Status event to the Host. When the Link Manager has completed the LMP messages to establish the requested QoS parameters, the Controller on the local Bluetooth device will send a QoS Setup Complete event to the Host, and the event may also be generated on the remote side if there was LMP negotiation. The values of the parameters of the QoS Setup Complete event may, however, be different on the initiating and the remote side. The QoS Setup Complete event returned by the Controller on the local side contains the status of this command, and returned QoS parameters describing the supported QoS for the connection.
Note: No Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the QoS Setup Complete event will indicate that this command has been completed.

7.2.7 Role Discovery Command

Command	OCF	Command Parameters	Return Parameters
HCI_Role_Discovery	0x0009	Connection_Handle	Status, Connection_Handle, Current_Role

Description:

The Role_Discovery command is used for a Bluetooth device to determine which role the device is performing for a particular Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Role_Discovery command succeeded,
0x01-0xFF	Role_Discovery command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify which connection the Role_Discovery command was issued on. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Current_Role: *Size: 1 Octet*

Value	Parameter Description
0x00	Current Role is Master for this Connection Handle.
0x01	Current Role is Slave for this Connection Handle.

Event(s) generated (unless masked away):

When the Role_Discovery command has completed, a Command Complete event will be generated.

7.2.8 Switch Role Command

Command	OCF	Command Parameters	Return Parameters
HCI_Switch_Role	0x000B	BD_ADDR, Role	

Description:

The Switch_Role command is used for a Bluetooth device to switch the current role the device is performing for a particular connection with another specified Bluetooth device. The BD_ADDR command parameter indicates for which connection the role switch is to be performed. The Role indicates the requested new role that the local device performs.
Note: the BD_ADDR command parameter must specify a Bluetooth device for which a connection already exists.

Note: If there is an SCO connection between the local device and the device identified by the BD_ADDR parameter, an attempt to perform a role switch shall be rejected by the local device.

Note: If the connection between the local device and the device identified by the BD_ADDR parameter is placed in Sniff Mode, an attempt to perform a role switch will be rejected by the local device.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the connected device with which a role switch is to be performed.

Role: *Size: 1 Octet*

Value	Parameter Description
0x00	Change own Role to Master for this BD_ADDR.
0x01	Change own Role to Slave for this BD_ADDR.

Return Parameters:

None.

Event(s) generated (unless masked away):

A Command Status event for this command will occur when the Controller has received the Switch_Role command. When the role switch is performed, a Role Change event will occur to indicate that the roles have been changed, and will be communicated to both Hosts.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, only the Role Change event will indicate that this command has been completed.

7.2.9 Read Link Policy Settings Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Link_Policy_ Settings	0x000C	Connection_Handle	Status, Connection_Handle Link_Policy_Settings

Description:

This command will read the Link Policy setting for the specified Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection. Section 6.19 on page 391.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Link_Policy_Settings command succeeded.
0x01-0xFF	Read_Link_Policy_Settings command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Link_Policy_Settings *Size: 2 Octets*

Value	Parameter Description
0x0000	Disable All LM Modes Default.
0x0001	Enable Role Switch.
0x0002	Enable Hold Mode.
0x0004	Enable Sniff Mode.
0x0008	Enable Park State.
0x0010 – 0x8000	Reserved for Future Use.

Event(s) generated (unless masked away):

When the Read_Link_Policy_Settings command has completed, a Command Complete event will be generated.

7.2.10 Write Link Policy Settings Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Link_Policy_Settings	0x000D	Connection_Handle, Link_Policy_Settings	Status, Connection_Handle

Description:

This command will write the Link Policy setting for the specified Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection. See Section 6.19 on page 391.

The default value is the value set by the Write Default Link Policy Settings Command.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Link_Policy_Settings *Size: 2 Octets*

Value	Parameter Description
0x0000	Disable All LM Modes.
0x0001	Enable Role Switch.
0x0002	Enable Hold Mode.
0x0004	Enable Sniff Mode.
0x0008	Enable Park State.
0x0010 – 0x8000	Reserved for Future Use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Link_Policy_Settings command succeeded.
0x01-0xFF	Write_Link_Policy_Settings command failed. See “Error Codes” on page 319 [Part D] for error codes and description.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Event(s) generated (unless masked away):

When the Write_Link_Policy_Settings command has completed, a Command Complete event will be generated.

7.2.11 Read Default Link Policy Settings Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Default_Link_Policy_Settings	0x000E		Status, Default_Link_Policy_Settings

Description:

This command will read the Default Link Policy setting for all new connections.

Note: Please refer to the Link Policy Settings configuration parameter for more information. See Section 6.19 on page 391.

Command Parameters:

None

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Link_Policy_Settings command succeeded
0x01-0xFF	Read_Link_Policy_Settings command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Default_Link_Policy_Settings *Size: 2 Octets*

Value	Parameter Description
0x0000	Disable All LM Modes Default
0x0001	Enable Role Switch
0x0002	Enable Hold Mode
0x0004	Enable Sniff Mode
0x0008	Enable Park State
0x0010 - 0x8000	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_Default_Link_Policy_Settings command has completed, a Command Complete event will be generated.

7.2.12 Write Default Link Policy Settings Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Default_Link_Policy_Settings	0x000F	Default_Link_Policy_Settings	Status

Description:

This command will write the Default Link Policy configuration value. The Default_Link_Policy_Settings parameter determines the initial value of the Link_Policy_Settings for all new connections.

Note: Please refer to the Link Policy Settings configuration parameter for more information. See Section 6.19 on page 391.

Command Parameters:

Default_Link_Policy_Settings *Size: 2 Octets*

Value	Parameter Description
0x0000	Disable All LM Modes Default
0x0001	Enable Role Switch
0x0002	Enable Hold Mode
0x0004	Enable Sniff Mode
0x0008	Enable Park State
0x0010 - 0x8000	Reserved for future use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Link_Policy_Settings command succeeded
0x01-0xFF	Write_Link_Policy_Settings command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Default_Link_Policy_Settings command has completed, a Command Complete event will be generated.

7.2.13 Flow Specification Command

Command	OCF	Command Parameters	Return Parameters
HCI_Flow_Specification	0x0010	Connection_Handle, Flags, Flow_direction, Service_Type, Token_Rate, Token_Bucket_Size, Peak_Bandwidth, Access Latency	

Description:

The Flow_Specification command is used to specify the flow parameters for the traffic carried over the ACL connection identified by the Connection_Handle. The Connection_Handle must be a Connection_Handle for an ACL connection. The Connection_Handle command parameter is used to identify for which connection the Flow Specification is requested. The flow parameters refer to the outgoing or incoming traffic of the ACL link, as indicated by the Flow_direction field. The Flow Specification command allows the Link Manager to have the parameters of the outgoing as well as the incoming flow for the ACL connection. The flow parameters are defined in the L2CAP specification "Quality of Service (QoS) Option" on page 60[vol. 4]. The Link Manager will determine if the flow parameters can be supported. Bluetooth devices that are both master and slave can use this command.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle used to identify for which ACL connection the Flow is specified. Range: 0x0000 - 0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

Flags: *Size: 1 Octet*

Value	Parameter Description
0x00 – 0xFF	Reserved for Future Use.

Flow_direction: *Size: 1 Octet*

Value	Parameter Description
0x00	Outgoing Flow i.e. traffic send over the ACL connection
0x01	Incoming Flow i.e. traffic received over the ACL connection
0x02 – 0xFF	Reserved for Future Use.

Service_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	No Traffic
0x01	Best Effort
0x02	Guaranteed
0x03 – 0xFF	Reserved for Future Use

Token Rate: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Token Rate in octets per second

Token Bucket Size: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Token Bucket Size in octets

Peak_Bandwidth: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Peak Bandwidth in octets per second

Access Latency: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Latency in microseconds

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Flow Specification command, the Controller sends the Command Status event to the Host. When the Link Manager has determined if the Flow specification can be supported, the Controller on the local Bluetooth device sends a Flow Specification Complete event to the Host. The Flow Specification Complete event returned by the Controller on the local side contains the status of this command, and returned Flow parameters describing the supported QoS for the ACL connection.

Note: No Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Flow Specification Complete event will indicate that this command has been completed.

7.3 CONTROLLER & BASEBAND COMMANDS

The Controller & Baseband Commands provide access and control to various capabilities of the Bluetooth hardware. These parameters provide control of Bluetooth devices and of the capabilities of the Controller, Link Manager, and Baseband. The host device can use these commands to modify the behavior of the local device. For the HCI Control and Baseband Commands, the OGF is defined as 0x03.

7.3.1 Set Event Mask Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_Event_Mask	0x0001	Event_Mask	Status

Description:

The Set_Event_Mask command is used to control which events are generated by the HCI for the Host. If the bit in the Event_Mask is set to a one, then the event associated with that bit will be enabled. The Host has to deal with each event that occurs by the Bluetooth devices. The event mask allows the Host to control how much it is interrupted.

Command Parameters:

Event_Mask: *Size: 8 Octets*

Value	Parameter Description
0x0000000000000000	No events specified
0x0000000000000001	Inquiry Complete Event
0x0000000000000002	Inquiry Result Event
0x0000000000000004	Connection Complete Event
0x0000000000000008	Connection Request Event
0x0000000000000010	Disconnection Complete Event
0x0000000000000020	Authentication Complete Event
0x0000000000000040	Remote Name Request Complete Event
0x0000000000000080	Encryption Change Event
0x0000000000000100	Change Connection Link Key Complete Event
0x0000000000000200	Master Link Key Complete Event
0x0000000000000400	Read Remote Supported Features Complete Event
0x0000000000000800	Read Remote Version Information Complete Event
0x0000000000001000	QoS Setup Complete Event
0x0000000000002000	Reserved
0x0000000000004000	Reserved
0x0000000000008000	Hardware Error Event

Value	Parameter Description
0x0000000000010000	Flush Occurred Event
0x0000000000020000	Role Change Event
0x0000000000040000	Reserved
0x0000000000080000	Mode Change Event
0x0000000000100000	Return Link Keys Event
0x0000000000200000	PIN Code Request Event
0x0000000000400000	Link Key Request Event
0x0000000000800000	Link Key Notification Event
0x0000000001000000	Loopback Command Event
0x0000000002000000	Data Buffer Overflow Event
0x0000000004000000	Max Slots Change Event
0x0000000008000000	Read Clock Offset Complete Event
0x0000000010000000	Connection Packet Type Changed Event
0x0000000020000000	QoS Violation Event
0x0000000040000000	Page Scan Mode Change Event [deprecated]
0x0000000080000000	Page Scan Repetition Mode Change Event
0x0000000100000000	Flow Specification Complete Event
0x0000000200000000	Inquiry Result with RSSI Event
0x0000000400000000	Read Remote Extended Features Complete Event
0x0000000800000000	Reserved
0x0000001000000000	Reserved
0x0000002000000000	Reserved
0x0000004000000000	Reserved
0x0000008000000000	Reserved
0x0000010000000000	Reserved
0x0000020000000000	Reserved
0x0000040000000000	Reserved
0x0000080000000000	Synchronous Connection Complete Event
0x0000100000000000	Synchronous Connection Changed event
0xFFFFE00000000000	Reserved for future use
0x00001FFFFFFFFFFF	**Default** (All events enabled)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_Event_Mask command succeeded.
0x01-0xFF	Set_Event_Mask command failed. See "Error Codes" on page 319 [Part D] for error codes and description.

Event(s) generated (unless masked away):

When the Set_Event_Mask command has completed, a Command Complete event will be generated.

7.3.2 Reset Command

Command	OCF	Command Parameters	Return Parameters
HCI_Reset	0x0003		Status

Description:

The Reset command will reset the Controller and the Link Manager. The reset command shall not affect the used HCI transport layer since the HCI transport layers may have reset mechanisms of their own. After the reset is completed, the current operational state will be lost, the Bluetooth device will enter standby mode and the Controller will automatically revert to the default values for the parameters for which default values are defined in the specification.

Note: the Host is not allowed to send additional HCI commands before the Command Complete event related to the Reset command has been received.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Reset command succeeded, was received and will be executed.
0x01-0xFF	Reset command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the reset has been performed, a Command Complete event will be generated.

7.3.3 Set Event Filter Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_Event_Filter	0x0005	Filter_Type, Filter_Condition_Type, Condition	Status

Description:

The Set_Event_Filter command is used by the Host to specify different event filters. The Host may issue this command multiple times to request various conditions for the same type of event filter and for different types of event filters. The event filters are used by the Host to specify items of interest, which allow the Controller to send only events which interest the Host. Only some of the events have event filters. By default (before this command has been issued after power-on or Reset) no filters are set, and the Auto_Accept_Flag is off (incoming connections are not automatically accepted). An event filter is added each time this command is sent from the Host and the Filter_Condition_Type is not equal to 0x00. (The old event filters will not be overwritten). To clear all event filters, the Filter_Type = 0x00 is used. The Auto_Accept_Flag will then be set to off. To clear event filters for only a certain Filter_Type, the Filter_Condition_Type = 0x00 is used.

The Inquiry Result filter allows the Controller to filter out Inquiry Result events. The Inquiry Result filter allows the Host to specify that the Controller only sends Inquiry Results to the Host if the Inquiry Result event meets one of the specified conditions set by the Host. For the Inquiry Result filter, the Host can specify one or more of the following Filter Condition Types:

1. Return responses from all devices during the Inquiry process
2. A device with a specific Class of Device responded to the Inquiry process
3. A device with a specific BD_ADDR responded to the Inquiry process

The Inquiry Result filter is used in conjunction with the Inquiry and Periodic Inquiry command.

The Connection Setup filter allows the Host to specify that the Controller only sends a Connection Complete or Connection Request event to the Host if the event meets one of the specified conditions set by the Host. For the Connection Setup filter, the Host can specify one or more of the following Filter Condition Types:

1. Allow Connections from all devices
2. Allow Connections from a device with a specific Class of Device
3. Allow Connections from a device with a specific BD_ADDR

For each of these conditions, an Auto_Accept_Flag parameter allows the Host to specify what action should be done when the condition is met. The Auto_Accept_Flag allows the Host to specify if the incoming connection should be auto accepted (in which case the Controller will send the Connection Complete event to the Host when the connection is completed) or if the Host should make the decision (in which case the Controller will send the Connection Request event to the Host, to elicit a decision on the connection).

The Connection Setup filter is used in conjunction with the Read/Write_Scan_Enable commands. If the local device is in the process of a page scan, and is paged by another device which meets one on the conditions set by the Host, and the Auto_Accept_Flag is off for this device, then a Connection Request event will be sent to the Host by the Controller. A Connection Complete event will be sent later on after the Host has responded to the incoming connection attempt. In this same example, if the Auto_Accept_Flag is on, then a Connection Complete event will be sent to the Host by the Controller. (No Connection Request event will be sent in that case.)

The Controller will store these filters in volatile memory until the Host clears the event filters using the Set_Event_Filter command or until the Reset command is issued. The number of event filters the Controller can store is implementation dependent. If the Host tries to set more filters than the Controller can store, the Controller will return the *Memory Full* error code and the filter will not be installed.
Note: the Clear All Filters has no Filter Condition Types or Conditions.
Note: In the condition that a connection is auto accepted, a Link Key Request event and possibly also a PIN Code Request event and a Link Key Notification event could be sent to the Host by the Controller before the Connection Complete event is sent.

If there is a contradiction between event filters, the latest set event filter will override older ones. An example is an incoming connection attempt where more than one Connection Setup filter matches the incoming connection attempt, but the Auto-Accept_Flag has different values in the different filters.

Command Parameters:

Filter_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Clear All Filters (Note: In this case, the Filter_Condition_type and Condition parameters should not be given, they should have a length of 0 octets. Filter_Type should be the only parameter.)
0x01	Inquiry Result.
0x02	Connection Setup.
0x03-0xFF	Reserved for Future Use.

Filter Condition Types: For each Filter Type one or more Filter Condition types exists.

Inquiry_Result_Filter_Condition_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Return responses from all devices during the Inquiry process. (Note: A device may be reported to the Host in an Inquiry Result event more than once during an inquiry or inquiry period depending on the implementation, see description in Section 7.1.1 on page 399 and Section 7.1.3 on page 402)
0x01	A device with a specific Class of Device responded to the Inquiry process.
0x02	A device with a specific BD_ADDR responded to the Inquiry process.
0x03-0xFF	Reserved for Future Use

Connection_Setup_Filter_Condition_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Allow Connections from all devices.
0x01	Allow Connections from a device with a specific Class of Device.
0x02	Allow Connections from a device with a specific BD_ADDR.
0x03-0xFF	Reserved for Future Use.

Condition: For each Filter Condition Type defined for the Inquiry Result Filter and the Connection Setup Filter, zero or more Condition parameters are required – depending on the filter condition type and filter type.

Condition for Inquiry_Result_Filter_Condition_Type = 0x00

Condition: *Size: 0 Octet*

Value	Parameter Description
	The Condition parameter is not used.

Condition for Inquiry_Result_Filter_Condition_Type = 0x01

Condition:

Size: 6 Octets

Class_of_Device: *Size: 3 Octets*

Value	Parameter Description
0x000000	Default, Return All Devices.
0xXXXXXX	*Class of Device* of Interest.

Class_of_Device_Mask: *Size: 3 Octets*

Value	Parameter Description
0xXXXXXX	Bit Mask used to determine which bits of the Class of Device parameter are ‘don’t care’. Zero-value bits in the mask indicate the ‘don’t care’ bits of the Class of Device.

Condition for Inquiry_Result_Filter_Condition_Type = 0x02

Condition:

Size: 6 Octets

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device of Interest

Condition for Connection_Setup_Filter_Condition_Type = 0x00

Condition:

Size: 1 Octet

Auto_Accept_Flag: *Size:1 Octet*

Value	Parameter Description
0x01	Do NOT Auto accept the connection. (Auto accept is off)
0x02	Do Auto accept the connection with role switch disabled. (Auto accept is on).
0x03	Do Auto accept the connection with role switch enabled. (Auto accept is on). Note: When auto accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto_Accept_Flag will then get the same effect as if the value had been 0x02.
0x04 – 0xFF	Reserved for future use.

Condition for Connection_Setup_Filter_Condition_Type = 0x01

Condition:

Size: 7 Octets

Class_of_Device: *Size: 3 Octets*

Value	Parameter Description
0x000000	Default, Return All Devices.
0xXXXXXX	*Class of Device* of Interest.

Class_of_Device_Mask: *Size: 3 Octets*

Value	Parameter Description
0xXXXXXX	Bit Mask used to determine which bits of the Class of Device parameter are 'don't care'. Zero-value bits in the mask indicate the 'don't care' bits of the Class of Device. Note: For an incoming SCO connection, if the class of device is unknown then the connection will be accepted.

Auto_Accept_Flag: *Size: 1 Octet*

Value	Parameter Description
0x01	Do NOT Auto accept the connection. (Auto accept is off)
0x02	Do Auto accept the connection with role switch disabled. (Auto accept is on).
0x03	Do Auto accept the connection with role switch enabled. (Auto accept is on). Note: When auto accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto_Accept_Flag will then get the same effect as if the value had been 0x02.
0x04 – 0xFF	Reserved for future use.

Condition for Connection_Setup_Filter_Condition_Type = 0x02

Condition:

Size: 7 Octets

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device of Interest.

Auto_Accept_Flag: *Size: 1 Octet*

Value	Parameter Description
0x01	Do NOT Auto accept the connection. (Auto accept is off)
0x02	Do Auto accept the connection with role switch disabled. (Auto accept is on).
0x03	Do Auto accept the connection with role switch enabled. (Auto accept is on). **Note: When auto accepting an incoming synchronous connection, no role switch will be performed. The value 0x03 of the Auto_Accept_Flag will then get the same effect as if the value had been 0x02**.
0x04 – 0xFF	Reserved for future use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_Event_Filter command succeeded.
0x01-0xFF	Set_Event_Filter command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

A Command Complete event for this command will occur when the Controller has enabled the filtering of events. When one of the conditions are met, a specific event will occur.

7.3.4 Flush Command

Command	OCF	Command Parameters	Return Parameters
HCI_Flush	0x0008	Connection_Handle	Status, Connection_Handle

Description:

The Flush command is used to discard all data that is currently pending for transmission in the Controller for the specified connection handle, even if there currently are chunks of data that belong to more than one L2CAP packet in the Controller. After this, all data that is sent to the Controller for the same connection handle will be discarded by the Controller until an HCI Data Packet with the start Packet_Boundary_Flag (0x02) is received. When this happens, a new transmission attempt can be made. This command will allow higher-level software to control how long the baseband should try to retransmit a baseband packet for a connection handle before all data that is currently pending for transmission in the Controller should be flushed. Note that the Flush command

is used for ACL connections ONLY. In addition to the Flush command, the automatic flush timers (see section 7.3.31 on page 505) can be used to automatically flush the L2CAP packet that is currently being transmitted after the specified flush timer has expired.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify which connection to flush. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Flush command succeeded.
0x01-0xFF	Flush command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify which connection the flush command was issued on. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Event(s) generated (unless masked away):

The Flush Occurred event will occur once the flush is completed. A Flush Occurred event could be from an automatic Flush or could be cause by the Host issuing the Flush command. When the Flush command has completed, a Command Complete event will be generated, to indicate that the Host caused the Flush.

7.3.5 Read PIN Type Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_PIN_Type	0x0009		Status, PIN_Type

Description:

The Read PIN Type command is used to read the PIN_Type configuration parameter. See Section 6.13 on page 388.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_PIN_Type command succeeded.
0x01-0xFF	Read_PIN_Type command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

PIN_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Variable PIN.
0x01	Fixed PIN.

Event(s) generated (unless masked away):

When the Read_PIN_Type command has completed, a Command Complete event will be generated.

7.3.6 Write PIN Type Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_PIN_Type	0x000A	PIN_Type	Status

Description:

The Write_PIN_Type command is used to write the PIN Type configuration parameter. See Section 6.13 on page 388.

Command Parameters:

PIN_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Variable PIN.
0x01	Fixed PIN.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write PIN Type command succeeded.
0x01-0xFF	Write PIN Type command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_PIN_Type command has completed, a Command Complete event will be generated.

7.3.7 Create New Unit Key Command

Command	OCF	Command Parameters	Return Parameters
HCI_Create_New_Unit_Key	0x000B		Status

Description:

The Create_New_Unit_Key command is used to create a new unit key. The Bluetooth hardware will generate a random seed that will be used to generate the new unit key. All new connection will use the new unit key, but the old unit key will still be used for all current connections.
Note: this command will not have any effect for a device which doesn't use unit keys (i.e. a device which uses only combination keys).

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Create New Unit Key command succeeded.
0x01-0xFF	Create New Unit Key command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Create_New_Unit_Key command has completed, a Command Complete event will be generated.

7.3.8 Read Stored Link Key Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Stored_ Link_Key	0x000D	BD_ADDR, Read_All_Flag	Status, Max_Num_Keys, Num_Keys_Read

Description:

The Read_Stored_Link_Key command provides the ability to read one or more link keys stored in the Bluetooth Controller. The Bluetooth Controller can store a limited number of link keys for other Bluetooth devices. Link keys are shared between two Bluetooth devices, and are used for all security transactions between the two devices. A Host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the Bluetooth Controller when needed. The Read_All_Flag parameter is used to indicate if all of the stored Link Keys should be returned. If Read_All_Flag indicates that all Link Keys are to be returned, then the BD_ADDR command parameter must be ignored The BD_ADDR command parameter is used to identify which link key to read. The stored Link Keys are returned by one or more Return Link Keys events. See Section 6.14 on page 388.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the stored link key to be read.

Read_All_Flag: *Size: 1 Octet*

Value	Parameter Description
0x00	Return Link Key for specified BD_ADDR.
0x01	Return all stored Link Keys.
0x02-0xFF	Reserved for future use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Stored_Link_Key command succeeded.
0x01-0xFF	Read_Stored_Link_Key command failed.See “Error Codes” on page 319 [Part D] for error codes and description.

Max_Num_Keys: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Total Number of Link Keys that the Controller can store. Range: 0x0000 – 0xFFFF

Num_Keys_Read: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Number of Link Keys Read. Range: 0x0000 – 0xFFFF

Event(s) generated (unless masked away):

Zero or more instances of the Return Link Keys event will occur after the command is issued. When there are no link keys stored, no Return Link Keys events will be returned. When there are link keys stored, the number of link keys returned in each Return Link Keys event is implementation specific. When the Read Stored Link Key command has completed a Command Complete event will be generated.

7.3.9 Write Stored Link Key Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Stored_Link_Key	0x0011	Num_Keys_To_Write, *BD_ADDR*[i], Link_Key[i]	Status, Num_Keys_Written

Description:

The Write_Stored_Link_Key command provides the ability to write one or more link keys to be stored in the Bluetooth Controller. The Bluetooth Controller can store a limited number of link keys for other Bluetooth devices. If no additional space is available in the Bluetooth Controller then no additional link keys will be stored. If space is limited and if all the link keys to be stored will not fit in the limited space, then the order of the list of link keys without any error will determine which link keys are stored. Link keys at the beginning of the list will be stored first. The Num_Keys_Written parameter will return the number of link keys that were successfully stored. If no additional space is available, then the Host must delete one or more stored link keys before any additional link keys are stored. The link key replacement algorithm is implemented by the Host and not the Controller. Link keys are shared between two Bluetooth devices and are used for all security transactions between the two devices. A Host device may have additional storage capabilities, which can be used to save additional link keys to be reloaded to the

Bluetooth Controller when needed. See Section 6.14 on page 388.
Note: Link Keys are only stored by issuing this command.

Command Parameters:

Num_Keys_To_Write: *Size: 1 Octet*

Value	Parameter Description
0xXX	Number of Link Keys to Write. Range: 0x01 - 0x0B

BD_ADDR [i]: *Size: 6 Octets * Num_Keys_To_Write*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the associated Link Key.

Link_Key: *Size: 16 Octets*

Value	Parameter Description
0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX	Link Key for an associated BD_ADDR.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Stored_Link_Key command succeeded.
0x01-0xFF	Write_Stored_Link_Key command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Num_Keys_Written: *Size: 1 Octets*

Value	Parameter Description
0xXX	Number of Link Keys successfully written. Range: 0x00 – 0x0B

Event(s) generated (unless masked away):

When the Write_Stored_Link_Key command has completed, a Command Complete event will be generated.

7.3.10 Delete Stored Link Key Command

Command	OCF	Command Parameters	Return Parameters
HCI_Delete_Stored_ Link_Key	0x0012	BD_ADDR, Delete_All_Flag	Status, Num_Keys_Deleted

Description:

The Delete_Stored_Link_Key command provides the ability to remove one or more of the link keys stored in the Bluetooth Controller. The Bluetooth Controller can store a limited number of link keys for other Bluetooth devices. Link keys are shared between two Bluetooth devices and are used for all security transactions between the two devices. The Delete_All_Flag parameter is used to indicate if all of the stored Link Keys should be deleted. If the Delete_All_Flag indicates that all Link Keys are to be deleted, then the BD_ADDR command parameter must be ignored This command provides the ability to negate all security agreements between two devices. The BD_ADDR command parameter is used to identify which link key to delete. If a link key is currently in use for a connection, then the link key will be deleted when all of the connections are disconnected. See Section 6.14 on page 388.

Command Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the link key to be deleted.

Delete_All_Flag: *Size: 1 Octet*

Value	Parameter Description
0x00	Delete only the Link Key for specified BD_ADDR.
0x01	Delete all stored Link Keys.
0x02-0xFF	Reserved for future use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Delete_Stored_Link_Key command succeeded.
0x01-0xFF	Delete_Stored_Link_Key command failed. See "Error Codes" on page 319 [Part D] for error codes and description.

Num_Keys_Deleted: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Number of Link Keys Deleted

Event(s) generated (unless masked away):

When the Delete_Stored_Link_Key command has completed, a Command Complete event will be generated.

7.3.11 Write Local Name Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Local_Name	0x0013	Local Name	Status

Description:

The Write_Local_Name command provides the ability to modify the user-friendly name for the Bluetooth device. See Section 6.24 on page 394.

Command Parameters:

Local Name: *Size: 248 Octets*

Value	Parameter Description
	A UTF-8 encoded User-Friendly Descriptive Name for the device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.
	Null terminated Zero length String. **Default.**

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Local_Name command succeeded.
0x01-0xFF	Write_Local_Name command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Local_Name command has completed, a Command Complete event will be generated.

7.3.12 Read Local Name Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Name	0x0014		Status, Local Name

Description:

The Read_Local_Name command provides the ability to read the stored user-friendly name for the Bluetooth device. See Section 6.24 on page 394.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Local_Name command succeeded
0x01-0xFF	Read_Local_Name command failed see "Error Codes" on page 319 [Part D] for list of Error Codes

Local Name: *Size: 248 Octets*

Value	Parameter Description
	A UTF-8 encoded User Friendly Descriptive Name for the device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.

Event(s) generated (unless masked away):

When the Read_Local_Name command has completed a Command Complete event will be generated.

7.3.13 Read Connection Accept Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Connection_ Accept_Timeout	0x0015		Status, Conn_Accept_Timeout

Description:

This command will read the value for the Connection_Accept_Timeout configuration parameter. See Section 6.7 on page 385.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Connection_Accept_Timeout command succeeded.
0x01-0xFF	Read_Connection_Accept_Timeout command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Conn_Accept_Timeout: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Connection Accept Timeout measured in Number of Baseband slots. Interval Length = N * 0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xB540 Time Range: 0.625 msec -29 seconds

Event(s) generated (unless masked away):

When the Read_Connection_Timeout command has completed, a Command Complete event will be generated.

7.3.14 Write Connection Accept Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Connection_ Accept_Timeout	0x0016	Conn_Accept_Timeout	Status

Description:

This command will write the value for the Connection_Accept_Timeout configuration parameter. See Section 6.7 on page 385.

Command Parameters:

Conn_Accept_Timeout: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Connection Accept Timeout measured in Number of Baseband slots. Interval Length = N * 0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xB540 Time Range: 0.625 msec - 29 seconds Default: N = 0x1FA0 Time = 5.06 Sec Mandatory Range for Controller: 0x00A0 to 0xB540

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Connection_Accept_Timeout command succeeded.
0x01-0xFF	Write_Connection_Accept_Timeout command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Connection_Accept_Timeout command has completed, a Command Complete event will be generated.

7.3.15 Read Page Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Page_Timeout	0x0017		Status, Page_Timeout

Description:

This command will read the value for the Page_Timeout configuration parameter. See Section 6.6 on page 385.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Page_Timeout command succeeded.
0x01-0xFF	Read_Page_Timeout command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Page_Timeout: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Page Timeout measured in Number of Baseband slots. Interval Length = N * 0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xFFFF Time Range: 0.625 msec -40.9 Seconds

Event(s) generated (unless masked away):

When the Read_Page_Timeout command has completed, a Command Complete event will be generated.

7.3.16 Write Page Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Page_Timeout	0x0018	Page_Timeout	Status

Description:

This command will write the value for the Page_Timeout configuration parameter. The Page_Timeout configuration parameter defines the maximum time the local Link Manager will wait for a baseband page response from the remote device at a locally initiated connection attempt. If this time expires and the remote device has not responded to the page at baseband level, the connection attempt will be considered to have failed.

Command Parameters:

Page_Timeout: *Size: 2 Octets*

Value	Parameter Description
0	Illegal Page Timeout. Must be larger than 0.
N = 0xXXXX	Page Timeout measured in Number of Baseband slots. Interval Length = N * 0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xFFFF Time Range: 0.625 msec -40.9 Seconds **Default: N = 0x2000 Time = 5.12 Sec** Mandatory Range for Controller: 0x0016 to 0xFFFF

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Page_Timeout command succeeded.
0x01-0xFF	Write_Page_Timeout command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Page_Timeout command has completed, a Command Complete event will be generated.

7.3.17 Read Scan Enable Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Scan_Enable	0x0019		Status, Scan_Enable

Description:

This command will read the value for the Scan_Enable parameter configuration parameter. See Section 6.1 on page 383.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Scan_Enable command succeeded.
0x01-0xFF	Read_Scan_Enable command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Scan_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	No Scans enabled.
0x01	Inquiry Scan enabled. Page Scan disabled.
0x02	Inquiry Scan disabled. Page Scan enabled.
0x03	Inquiry Scan enabled. Page Scan enabled.
0x04-0xFF	Reserved

Event(s) generated (unless masked away):

When the Read_Scan_Enable command has completed, a Command Complete event will be generated.

7.3.18 Write Scan Enable Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Scan_Enable	0x001A	Scan_Enable	Status

Description:

This command will write the value for the Scan_Enable configuration parameter. See Section 6.1 on page 383.

Command Parameters:

Scan_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	No Scans enabled. **Default.**
0x01	Inquiry Scan enabled. Page Scan disabled.
0x02	Inquiry Scan disabled. Page Scan enabled.
0x03	Inquiry Scan enabled. Page Scan enabled.
0x04-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Scan_Enable command succeeded.
0x01-0xFF	Write_Scan_Enable command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Scan_Enable command has completed, a Command Complete event will be generated.

7.3.19 Read Page Scan Activity Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Page_Scan_ Activity	0x001B		Status, Page_Scan_Interval, Page_Scan_Window

Description:

This command will read the value for Page_Scan_Interval and Page_Scan_Window configuration parameters. See Section 6.8 on page 386 and Section 6.9 on page 386.
Note: Page Scan is only performed when Page_Scan is enabled (see 6.1, 7.3.17 and 7.3.18).

A changed Page_Scan_Interval could change the local Page_Scan_ Repetition_Mode (see "Baseband Specification" on page 55 [Part B], Keyword: SR-Mode).

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Page_Scan_Activity command succeeded.
0x01-0xFF	Read_Page_Scan_Activity command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Page_Scan_Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0012 – 0x1000 Time = N * 0.625 msec Range: 11.25 msec - 2560 msec; only even values are valid

Page_Scan_Window: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0011 - 0x1000 Time = N * 0.625 msec Range: 10.625 msec - 2560 msec

Event(s) generated (unless masked away):

When the Read_Page_Scan_Activity command has completed, a Command Complete event will be generated.

7.3.20 Write Page Scan Activity Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Page_Scan_ Activity	0x001C	Page_Scan_Interval, Page_Scan_Window	Status

Description:

This command will write the values for the Page_Scan_Interval and Page_Scan_Window configuration parameters. The Page_Scan_Window shall be less than or equal to the Page_Scan_Interval. See Section 6.8 on page 386 and Section 6.9 on page 386.
Note: Page Scan is only performed when Page_Scan is enabled (see 6.1, 7.3.17 and 7.3.18). A changed Page_Scan_Interval could change the local Page_Scan_Repetition_Mode (see Baseband Specification, Section 8.3.1, on page 154).

Command Parameters:

Page_Scan_Interval: *Size: 2 Octets*

Value	Parameter Description
See Section 6.8 on page 386	

Page_Scan_Window: *Size: 2 Octets*

Value	Parameter Description
See Section 6.9 on page 386	

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Page_Scan_Activity command succeeded.
0x01-0xFF	Write_Page_Scan_Activity command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Page_Scan_Activity command has completed, a Command Complete event will be generated.

7.3.21 Read Inquiry Scan Activity Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_ Inquiry_Scan_Activity	0x001D		Status, Inquiry_Scan_Interval, Inquiry_Scan_Window

Description:

This command will read the value for Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameter. See Section 6.2 on page 383 and Section 6.3 on page 384.
Note: Inquiry Scan is only performed when Inquiry_Scan is enabled see 6.1, 7.3.17 and 7.3.18).

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Inquiry_Scan_Activity command succeeded.
0x01-0xFF	Read_Inquiry_Scan_Activity command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Inquiry_Scan_Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0012 – 0x1000 Time = N * 0.625 msec Range: 11.25 - 2560 msec; only even values are valid

Inquiry_Scan_Window: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Size: 2 Octets Range: 0x0011 - 0x1000 Time = N * 0.625 msec Range: 10.625 msec - 2560 msec

Event(s) generated (unless masked away):

When the Read_Inquiry_Scan_Activity command has completed, a Command Complete event will be generated.

7.3.22 Write Inquiry Scan Activity Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Inquiry_ Scan_Activity	0x001E	Inquiry_Scan_Interval, Inquiry_Scan_Window	Status

Description:

This command will write the values for the Inquiry_Scan_Interval and Inquiry_Scan_Window configuration parameters. The Inquiry_Scan_Window shall be less than or equal to the Inquiry_Scan_Interval. See Section 6.2 on page 383 and Section 6.3 on page 384.
Note: Inquiry Scan is only performed when Inquiry_Scan is enabled (see 6.1, 7.3.17 and 7.3.18).

Command Parameters:

Inquiry_Scan_Interval: *Size: 2 Octets*

Value	Parameter Description
See Section 6.2 on page 383.	

Inquiry_Scan_Window: *Size: 2 Octets*

Value	Parameter Description
See Section 6.3 on page 384.	

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Inquiry_Scan_Activity command succeeded.
0x01-0xFF	Write_Inquiry_Scan_Activity command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Inquiry_Scan_Activity command has completed, a Command Complete event will be generated.

7.3.23 Read Authentication Enable Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_ Authentication_Enable	0x001F		Status, Authentication_Enable

Description:

This command will read the value for the Authentication_Enable configuration parameter. See Section 6.15 on page 388.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Authentication_Enable command succeeded.
0x01-0xFF	Read_Authentication_Enable command failed.See "Error Codes" on page 319 [Part D] for list of Error Codes.

Authentication_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	Authentication not required.
0x01	Authentication required for all connections.
0x02-0xFF	Reserved

Event(s) generated (unless masked away):

When the Read_Authentication_Enable command has completed, a Command Complete event will be generated.

7.3.24 Write Authentication Enable Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_ Authentication_Enable	0x0020	Authentication_Enable	Status

Description:

This command will write the value for the Authentication_Enable configuration parameter. See Section 6.15 on page 388.

Command Parameters:

Authentication_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	Authentication not required. **Default.**
0x01	Authentication required for all connections.
0x02-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write Authentication_Enable command succeeded.
0x01-0xFF	Write Authentication_Enable command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Authentication_Enable command has completed, a Command Complete event will be generated.

7.3.25 Read Encryption Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Encryption_Mode	0x0021		Status, Encryption_Mode

Description:

This command will read the value for the Encryption_Mode configuration parameter. See Section 6.16 on page 389.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Encryption_Mode command succeeded.
0x01-0xFF	Read_Encryption_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Encryption_Mode: *Size: 1 Octet*

Value	Parameter Description
See Section 6.16 on page 389	

Event(s) generated (unless masked away):

When the Read_Encryption_Mode command has completed, a Command Complete event will be generated.

7.3.26 Write Encryption Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Encryption_Mode	0x0022	Encryption_Mode	Status

Description:

This command will write the value for the Encryption_Mode configuration parameter. See Section 6.16 on page 389.

Command Parameters:

Encryption_Mode: *Size: 1 Octet*

Value	Parameter Description
See Section 6.16 on page 389.	

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Encryption_Mode command succeeded.
0x01-0xFF	Write_Encryption_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Encryption_Mode command has completed, a Command Complete event will be generated.

7.3.27 Read Class of Device Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Class_of_Device	0x0023		Status, Class_of_Device

Description:

This command will read the value for the Class_of_Device parameter. See Section 6.25 on page 394.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Class_of_Device command succeeded.
0x01-0xFF	Read_Class_of_Device command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Class_of_Device: *Size: 3 Octets*

Value	Parameter Description
0xXXXXXX	Class of Device for the device.

Event(s) generated (unless masked away):

When the Read_Class_of_Device command has completed, a Command Complete event will be generated.

7.3.28 Write Class of Device Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Class_of_Device	0x0024	Class_of_Device	Status

Description:

This command will write the value for the Class_of_Device parameter. See Section 6.25 on page 394.

Command Parameters:

Class_of_Device: *Size: 3 Octets*

Value	Parameter Description
0xXXXXXX	Class of Device for the device.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Class_of_Device command succeeded.
0x01-0xFF	Write_Class_of_Device command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Class_of_Device command has completed, a Command Complete event will be generated.

7.3.29 Read Voice Setting Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Voice_Setting	0x0025		Status, Voice_Setting

Description:

This command will read the values for the Voice_Setting configuration parameter. See Section 6.12 on page 387.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Voice_Setting command succeeded.
0x01-0xFF	Read_Voice_Setting command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Voice_Setting: *Size: 2 Octets (10 Bits meaningful)*

Value	Parameter Description
See Section 6.12 on page 387.	

Event(s) generated (unless masked away):

When the Read_Voice_Setting command has completed, a Command Complete event will be generated.

7.3.30 Write Voice Setting Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Voice_Setting	0x0026	Voice_Setting	Status

Description:

This command will write the values for the Voice_Setting configuration parameter. See Section 6.12 on page 387.

Command Parameters:

Voice_Setting: *Size: 2 Octets (10 Bits meaningful)*

Value	Parameter Description
See Section 6.12 on page 387.	

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Voice_Setting command succeeded.
0x01-0xFF	Write_Voice_Setting command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Voice_Setting command has completed, a Command Complete event will be generated.

7.3.31 Read Automatic Flush Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Automatic_Flush_Timeout	0x0027	Connection_Handle	Status, Connection_Handle, Flush_Timeout

Description:

This command will read the value for the Flush_Timeout parameter for the specified connection handle. See Section 6.20 on page 392.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Flush Timeout to read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Automatic_Flush_Timeout command succeeded.
0x01-0xFF	Read_Automatic_Flush_Timeout command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Flush Timeout has been read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Flush_Timeout: *Size: 2 Octets*

Value	Parameter Description
0	Timeout = ∞; No Automatic Flush
N = 0xXXXX	Flush Timeout = N * 0.625 msec Size: 11 bits Range: 0x0001 – 0x07FF

Event(s) generated (unless masked away):

When the Read_Automatic_Flush_Timeout command has completed, a Command Complete event will be generated.

7.3.32 Write Automatic Flush Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Automatic_Flush_Timeout	0x0028	Connection_Handle, Flush_Timeout	Status, Connection_Handle

Description:

This command will write the value for the Flush_Timeout parameter for the specified connection handle. See Section 6.20 on page 392.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Flush Timeout to write to. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Flush_Timeout: *Size: 2 Octets*

Value	Parameter Description
0	Timeout = ∞; No Automatic Flush. **Default.**
N = 0xXXXX	Flush Timeout = N * 0.625 msec Size: 11 bits Range: 0x0001 – 0x07FF Mandatory Range for Controller: 0x0002 to 0x07FF

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Automatic_Flush_Timeout command succeeded.
0x01-0xFF	Write_Automatic_Flush_Timeout command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Flush Timeout has been written. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Event(s) generated (unless masked away):

When the Write_Automatic_Flush_Timeout command has completed, a Command Complete event will be generated.

7.3.33 Read Num Broadcast Retransmissions Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Num_Broadcast_ Retransmissions	0x0029		Status, Num_Broadcast_Retransmissions

Description:

This command will read the device's parameter value for the Number of Broadcast Retransmissions. See Section 6.21 on page 392

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Num_Broadcast_Retransmissions command succeeded.
0x01-0xFF	Read_Num_Broadcast_Retransmissions command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Num_Broadcast_Retransmissions: *Size: 1 Octet*

Value	Parameter Description
See Section 6.21 on page 392.	

Event(s) generated (unless masked away):

When the Read_Num_Broadcast_Retransmission command has completed, a Command Complete event will be generated.

7.3.34 Write Num Broadcast Retransmissions Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Num_Broadcast_ Retransmissions	0x002A	Num_Broadcast_ Retransmissions	Status

Description:

This command will write the device's parameter value for the Number of Broadcast Retransmissions. See Section 6.21 on page 392.

Command Parameters:

Num_Broadcast_Retransmissions: *Size: 1 Octet*

Value	Parameter Description
See Section 6.21 on page 392.	

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Num_Broadcast_Retransmissions command succeeded.
0x01-0xFF	Write_Num_Broadcast_Retransmissions command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Num_Broadcast_Retransmissions command has completed, a Command Complete event will be generated.

7.3.35 Read Hold Mode Activity Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Hold_Mode_Activity	0x002B		Status, Hold_Mode_Activity

Description:

This command will read the value for the Hold_Mode_Activity parameter. See Section 6.18 on page 390.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Hold_Mode_Activity command succeeded.
0x01-0xFF	Read_Hold_Mode_Activity command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Hold_Mode_Activity: *Size: 1 Octet*

Value	Parameter Description
0x00	Maintain current Power State.
0x01	Suspend Page Scan.
0x02	Suspend Inquiry Scan.
0x04	Suspend Periodic Inquiries.
0x08-0xFF	Reserved for Future Use.

Event(s) generated (unless masked away):

When the Read_Hold_Mode_Activity command has completed, a Command Complete event will be generated.

7.3.36 Write Hold Mode Activity Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Hold_Mode_Activity	0x002C	Hold_Mode_Activity	Status

Description:

This command will write the value for the Hold_Mode_Activity parameter. See Section 6.18 on page 390.

Command Parameters:

Hold_Mode_Activity: *Size: 1 Octet*

Value	Parameter Description
0x00	Maintain current Power State. **Default.**
0x01	Suspend Page Scan.
0x02	Suspend Inquiry Scan.
0x04	Suspend Periodic Inquiries.
0x08-0xFF	Reserved for Future Use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Hold_Mode_Activity command succeeded.
0x01-0xFF	Write_Hold_Mode_Activity command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Hold_Mode_Activity command has completed, a Command Complete event will be generated.

7.3.37 Read Transmit Power Level Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Transmit_ Power_Level	0x002D	Connection_Handle, Type	Status, Connection_Handle, Transmit_Power_Level

Description:

This command will read the values for the Transmit_Power_Level parameter for the specified Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Transmit Power Level setting to read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Read Current Transmit Power Level.
0x01	Read Maximum Transmit Power Level.
0x02-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Transmit_Power_Level command succeeded.
0x01-0xFF	Read_Transmit_Power_Level command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Transmit Power Level setting is returned. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Transmit_Power_Level: *Size: 1 Octet*

Value	Parameter Description
N = 0xXX	Size: 1 Octet (signed integer) Range: -30 ≤ N ≤ 20 Units: dBm

Event(s) generated (unless masked away):

When the Read_Transmit_Power_Level command has completed, a Command Complete event will be generated.

7.3.38 Read Synchronous Flow Control Enable Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_ Synchronous_Flow_ Control_Enable	0x002E		Status, Synchronous_Flow_ Control_Enable

Description:

The Read_Synchronous_Flow_Control_Enable command provides the ability to read the Synchronous_Flow_Control_Enable setting. See Section 6.23 on page 393.
Note: the Synchronous_Flow_Control_Enable setting can only be changed if no connections exist.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Synchronous_Flow_Control_Enable command succeeded
0x01-0xFF	Read_Synchronous_Flow_Control_Enable command failed see “Error Codes” on page 319 [Part D] for list of Error Codes

Synchronous_Flow_Control_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	Synchronous Flow Control is disabled. No Number of Completed Packets events will be sent from the Controller for Synchronous Connection Handles.
0x01	Synchronous Flow Control is enabled. Number of Completed Packets events will be sent from the Controller for Synchronous Connection Handles.

Event(s) generated (unless masked away):

When the Read_Synchronous_Flow_Control_Enable command has completed a Command Complete event will be generated.

7.3.39 Write Synchronous Flow Control Enable Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Synchronous_ Flow_Control_Enable	0x002F	Synchronous_Flow_ Control_Enable	Status

Description:

The Write_Synchronous_Flow_Control_Enable command provides the ability to write the Synchronous_Flow_Control_Enable setting. See Section 6.23 on page 393.
Note: the Synchronous_Flow_Control_Enable setting can only be changed if no connections exist.

Command Parameters:

Synchronous_Flow_Control_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	Synchronous Flow Control is disabled. No Number of Completed Packets events will be sent from the Controller for synchronous Connection Handles. **Default**
0x01	Synchronous Flow Control is enabled. Number of Completed Packets events will be sent from the Controller for synchronous Connection Handles.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Synchronous_Flow_Control_Enable command succeeded
0x01-0xFF	Write_Synchronous_Flow_Control_Enable command failed see "Error Codes" on page 319 [Part D] for list of Error Codes

Event(s) generated (unless masked away):

When the Write_Synchronous_Flow_Control_Enable command has completed a Command Complete event will be generated.

7.3.40 Set Controller To Host Flow Control Command

Command	OCF	Command Parameters	Return Parameters
HCI_Set_Controller_To_ Host_Flow_Control	0x0031	Flow_Control_Enable	Status

Description:

This command is used by the Host to turn flow control on or off for data and/or voice sent in the direction from the Controller to the Host. If flow control is turned off, the Host should not send the Host_Number_Of_Completed_Packets command. That command will be ignored by the Controller if it is sent by the Host and flow control is off. If flow control is turned on for HCI ACL Data Packets and off for HCI synchronous Data Packets, Host_Number_Of_Completed_Packets commands sent by the Host should only contain Connection Handles for ACL connections. If flow control is turned off for HCI ACL Data Packets and on for HCI synchronous Data Packets, Host_Number_Of_Completed_Packets commands sent by the Host should only contain Connection Handles for synchronous connections. If flow control is turned on for HCI ACL Data Packets and HCI synchronous Data Packets, the Host will send Host_Number_Of_Completed_Packets commands both for ACL connections and synchronous connections.

The Flow_Control_Enable setting shall only be changed if no connections exist.

Command Parameters:

Flow_Control_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	Flow control **off** in direction from Controller to Host. **Default.**
0x01	Flow control **on** for HCI ACL Data Packets and **off** for HCI synchronous Data Packets in direction from Controller to Host.
0x02	Flow control **off** for HCI ACL Data Packets and **on** for HCI synchronous Data Packets in direction from Controller to Host.
0x03	Flow control **on** both for HCI ACL Data Packets and HCI synchronous Data Packets in direction from Controller to Host.
0x04-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_Controller_To_Host_Flow_Control command succeeded.
0x01-0xFF	Set_Controller_To_Host_Flow_Control command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Set_Controller_To_Host_Flow_Control command has completed, a Command Complete event will be generated.

7.3.41 Host Buffer Size Command

Command	OCF	Command Parameters	Return Parameters
HCI_Host_Buffer_Size	0x0033	Host_ACL_Data_Packet_Length, Host_Synchronous_Data_Packet_Length, Host_Total_Num_ACL_Data_Packets, Host_Total_Num_Synchronous_Data_Packets	Status

Description:

The Host_Buffer_Size command is used by the Host to notify the Controller about the maximum size of the data portion of HCI ACL and synchronous Data Packets sent from the Controller to the Host. The Controller will segment the data to be transmitted from the Controller to the Host according to these sizes, so that the HCI Data Packets will contain data with up to these sizes. The Host_Buffer_Size command also notifies the Controller about the total number of HCI ACL and synchronous Data Packets that can be stored in the data buffers of the Host. If flow control from the Controller to the Host is turned off, and the Host_Buffer_Size command has not been issued by the Host, this means that the Controller will send HCI Data Packets to the Host with any lengths the Controller wants to use, and it is assumed that the data buffer sizes of the Host are unlimited. If flow control from the Controller to the Host is turned on, the Host_Buffer_Size command must after a power-on or a reset always be sent by the Host before the first Host_Number_Of_Completed_Packets command is sent.

(The Set Controller To Host Flow Control Command command is used to turn flow control on or off.) The Host_ACL_Data_Packet_Length command parameter will be used to determine the size of the L2CAP segments contained in ACL Data Packets, which are transferred from the Controller to the Host. The Host_Synchronous_Data_Packet_Length command parameter is used to determine the maximum size of HCI synchronous Data Packets. Both the Host and the Controller must support command and event packets, where the data portion (excluding header) contained in the packets is 255 octets in size.

The Host_Total_Num_ACL_Data_Packets command parameter contains the total number of HCI ACL Data Packets that can be stored in the data buffers of the Host. The Controller will determine how the buffers are to be divided between different Connection Handles. The Host_Total_Num_Synchronous_Data_Packets command parameter gives the same information for HCI synchronous Data Packets.
Note: the Host_ACL_Data_Packet_Length and Host_Synchronous_Data_Packet_Length command parameters do not include the length of the HCI Data Packet header.

Command Parameters:

Host_ACL_Data_Packet_Length: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Maximum length (in octets) of the data portion of each HCI ACL Data Packet that the Host is able to accept.

Host_Synchronous_Data_Packet_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	Maximum length (in octets) of the data portion of each HCI synchronous Data Packet that the Host is able to accept.

Host_Total_Num_ACL_Data_Packets: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Total number of HCI ACL Data Packets that can be stored in the data buffers of the Host.

Host_Total_Num_Synchronous_Data_Packets: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Total number of HCI synchronous Data Packets that can be stored in the data buffers of the Host.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Host_Buffer_Size command succeeded.
0x01-0xFF	Host_Buffer_Size command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Host_Buffer_Size command has completed, a Command Complete event will be generated.

7.3.42 Host Number Of Completed Packets Command

Command	OCF	Command Parameters	Return Parameters
HCI_Host_Number_Of_Completed_Packets	0x0035	Number_Of_Handles, Connection_Handle[i], Host_Num_Of_Completed_Packets [i]	

Description:

The Host_Number_Of_Completed_Packets command is used by the Host to indicate to the Controller the number of HCI Data Packets that have been completed for each Connection Handle since the previous Host_Number_Of_Completed_Packetscommand was sent to the Controller. This means that the corresponding buffer space has been freed in the Host. Based on this information, and the Host_Total_Num_ACL_Data_Packets and Host_Total_Num_Synchronous_Data_Packets command parameters of the Host_Buffer_Size command, the Controller can determine for which Connection Handles the following HCI Data Packets should be sent to the Host. The command should only be issued by the Host if flow control in the direction from the Controller to the Host is on and there is at least one connection, or if the Controller is in local loopback mode. Otherwise, the command will be ignored by the Controller. When the Host has completed one or more HCI Data Packet(s) it shall send a Host_Number_Of_Completed_Packets command to the Controller, until it finally reports that all pending HCI Data Packets have been completed. The frequency at which this command is sent is manufacturer specific.

(The Set Controller To Host Flow Control Command command is used to turn flow control on or off.) If flow control from the Controller to the Host is turned on, the Host_Buffer_Size command must after a power-on or a reset always be sent by the Host before the first Host_Number_Of_Completed_Packets command is sent.
Note: the Host_Number_Of_Completed_Packets command is a special command in the sense that no event is normally generated after the command has completed. The command may be sent at any time by the Host when there is at least one connection, or if the Controller is in local loopback mode independent of other commands. The normal flow control for commands is not used for the Host_Number_Of_Completed_Packets command.

Command Parameters:

Number_Of_Handles: *Size: 1 Octet*

Value	Parameter Description
0xXX	The number of Connection Handles and Host_Num_Of_Completed_Packets parameters pairs contained in this command. Range: 0-255

*Connection_Handle[i]: Size: Number_Of_Handles*2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

*Host_Num_Of_Completed_Packets [i]: Size: Number_Of_Handles * 2 Octets*

Value	Parameter Description
N = 0xXXXX	The number of HCI Data Packets that have been completed for the associated Connection Handle since the previous time the event was returned. Range for N: 0x0000-0xFFFF

Return Parameters:

None.

Event(s) generated (unless masked away):

Normally, no event is generated after the Host_Number_Of_Completed_Packets command has completed. However, if the Host_Number_Of_Completed_Packets command contains one or more invalid parameters, the Controller will return a Command Complete event with a failure status indicating the *Invalid HCI Command Parameters error code*. The Host may send the Host_Number_Of_Completed_Packets command at any time when there is at least one connection, or if the Controller is in local loopback mode. The normal flow control for commands is not used for this command.

7.3.43 Read Link Supervision Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Link_Supervision_ Timeout	0x0036	Connection_Handle	Status, Connection_Handle, Link_Supervision_ Timeout

Description:

This command will read the value for the Link_Supervision_Timeout parameter for the device.
Note: the Connection_Handle used for this command must be the ACL connection to the appropriate device. See Section 6.22 on page 393.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Link Supervision Timeout value is to be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Link_Supervision_Timeout command succeeded.
0x01-0xFF	Read_Link_Supervision_Timeout command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Link Supervision Timeout value was read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Link_Supervision_Timeout: *Size: 2 Octets*

Value	Parameter Description
0x0000	No Link_Supervision_Timeout.
N = 0xXXXX	Measured in Number of Baseband slots Link_Supervision_Timeout = N * 0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xFFFF Time Range: 0.625ms - 40.9 sec

Event(s) generated (unless masked away):

When the Read_Link_Supervision_Timeout command has completed, a Command Complete event will be generated.

7.3.44 Write Link Supervision Timeout Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Link_Supervision_Timeout	0x0037	Connection_Handle, Link_Supervision_Timeout	Status, Connection_Handle

Description:

This command will write the value for the Link_Supervision_Timeout parameter for the device. This command shall only be issued on the master for the given connection handle. If this command is issued on a slave, the command shall be rejected with the Command Disallowed.
Note: the Connection_Handle used for this command must be the ACL connection to the appropriate device. This command will set the Link_Supervision_Timeout values for other Synchronous Connection_Handles to that device.
See Section 6.22 on page 393.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Link Supervision Timeout value is to be written. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Link_Supervision_Timeout: *Size: 2 Octets*

Value	Parameter Description
0x0000	No Link_Supervision_Timeout.
N = 0xXXXX	Measured in Number of Baseband slots Link_Supervision_Timeout = N*0.625 msec (1 Baseband slot) Range for N: 0x0001 – 0xFFFF Time Range: 0.625ms – 40.9 sec **Default**: N = 0x7D00 Link_Supervision_Timeout = 20 sec Mandatory Range for Controller: 0x0190 to 0xFFFF; plus 0 for infinite timeout

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Link_Supervision_Timeout command succeeded.
0x01-0xFF	Write_Link_Supervision_Timeout command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle’s Link Supervision Timeout value was written. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Event(s) generated (unless masked away):

When the Write_Link_Supervision_Timeout command has completed, a Command Complete event will be generated.

7.3.45 Read Number Of Supported IAC Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Number_Of_Supported_IAC	0x0038		Status, Num_Support_IAC

Description:

This command will read the value for the number of Inquiry Access Codes (IAC) that the local Bluetooth device can simultaneous listen for during an Inquiry Scan. All Bluetooth devices are required to support at least one IAC, the General Inquiry Access Code (the GIAC). Some Bluetooth devices support additional IACs.

Command Parameters:

None

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Number_Of_Supported_IAC command succeeded.
0x01-0xFF	Read_Number_Of_Supported_IAC command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Num_Support_IAC *Size: 1 Octet*

Value	Parameter Description
0xXX	Specifies the number of Supported IAC that the local Bluetooth device can simultaneous listen for during an Inquiry Scan. Range: 0x01-0x40

Event(s) generated (unless masked away):

When the Read_Number_Of_Supported_IAC command has completed, a Command Complete event will be generated.

7.3.46 Read Current IAC LAP Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Current_IAC_LAP	0x0039		Status, Num_Current_IAC, IAC_LAP[i]

Description:

This command reads the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. All Bluetooth devices are required to support at least one IAC, the General Inquiry Access Code (the GIAC). Some Bluetooth devices support additional IACs.

Command Parameters:

None

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Current_IAC_LAP command succeeded.
0x01-0xFF	Read_Current_IAC_LAP command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Num_Current_IAC *Size: 1 Octet*

Value	Parameter Description
0xXX	Specifies the number of IACs which are currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x01-0x40

IAC_LAP[i] *Size: 3 Octets * Num_Current_IAC*

Value	Parameter Description
0xXXXXXX	LAPs used to create the IAC which is currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x9E8B00-0x9E8B3F

Event(s) generated (unless masked away):

When the Read_Current_IAC_LAP command has completed, a Command Complete event will be generated.

7.3.47 Write Current IAC LAP Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Current_IAC_LAP	0x003A	Num_Current_IAC, IAC_LAP[i]	Status

Description:

This command writes the LAP(s) used to create the Inquiry Access Codes (IAC) that the local Bluetooth device is simultaneously scanning for during Inquiry Scans. All Bluetooth devices are required to support at least one IAC, the General Inquiry Access Code (the GIAC). Some Bluetooth devices support additional IACs.

This command shall clear any existing IACs and stores Num_Current_IAC and the IAC_LAPs in to the controller. If Num_Current_IAC is greater than Num_Support_IAC then only the first Num_Support_IACs shall be stored in the controller, and a Command Complete event with error code *Success (0x00)* shall be generated.

Command Parameters:

Num_Current_IAC *Size: 1 Octet*

Value	Parameter Description
0xXX	Specifies the number of IACs which are currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x01-0x40

IAC_LAP[i] *Size: 3 Octets * Num_Current_IAC*

Value	Parameter Description
0xXXXXXX	LAP(s) used to create IAC which is currently in use by the local Bluetooth device to simultaneously listen for during an Inquiry Scan. Range: 0x9E8B00-0x9E8B3F. The GIAC is the default IAC to be used. If additional IACs are supported, additional default IAC will be determined by the manufacturer.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Current_IAC_LAP command succeeded.
0x01-0xFF	Write_Current_IAC_LAP command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Current_IAC_LAP command has completed, a Command Complete event will be generated.

7.3.48 Read Page Scan Period Mode Command (Deprecated)

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Page_Scan_Period_Mode	0x003B		Status, Page_Scan_Period_Mode

Description:

This command is used to read the mandatory Page_Scan_Period_Mode configuration parameter of the local Bluetooth device. See Section 6.10 on page 386.

Command Parameters:

None

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Page_Scan_Period_Mode command succeeded.
0x01-0xFF	Read_Page_Scan_Period_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Page_Scan_Period_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	P0
0x01	P1
0x02	P2
0x03-0xFF	Reserved.

Event(s) generated (unless masked away):

When the Read_Page_Scan_Period_Mode command has completed, a Command Complete event will be generated.

7.3.49 Write Page Scan Period Mode Command (Deprecated)

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Page_Scan_Period_Mode	0x003C	Page_Scan_Period_Mode	Status

Description:

This command is used to write the mandatory Page_Scan_Period_Mode configuration parameter of the local Bluetooth device. See Section 6.10 on page 386.

Command Parameters:

Page_Scan_Period_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	P0
0x01	P1
0x02	P2. **Default.**
0x03-0xFF	Reserved.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Page_Scan_Period_Mode command succeeded.
0x01-0xFF	Write_Page_Scan_Period_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Page_Scan_Period_Mode command has completed, a Command Complete event will be generated.

7.3.50 Set AFH Host Channel Classification Command

Command	OCF	Command Parameters	Return Parameters
Set_AFH_Host_Channel_ Classification	0x003F	AFH_Host_Channel_ Classification	Status

Description:

The Set_AFH_Host_Channel_Classification command allows the Bluetooth host to specify a channel classification based on its “local information”. This classification persists until overwritten with a subsequent HCI Set_AFH_Host_Channel_Classification command or until the Controller is reset.

This command shall be supported by a device that declares support for any of the AFH_capable_master, AFH_classification_slave or AFH_classification_master features.

If this command is used, then updates should be sent within 10 seconds, of the host knowing that the channel classification has changed. The interval between two successive commands sent shall be at least 1 second.

Command Parameters:

AFH_Host_Channel_Classification: *Size: 10 Octets (79 Bits meaningful)*

Value	Parameter Description
0xXXXXXXXX XXXXXXXXXX XX	This parameter contains 79 1-bit field. The n^{th} such field (in the range 0 to 78) contains the value for channel *n*: Channel *n* is bad = 0 Channel *n* is unknown = 1 The most significant bit is reserved and shall be set to 0 At least N_{min} channels shall be marked as unknown. (See Baseband Specification, Section 2.3.1, on page 75)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Set_AFH_Host_Channel_Classification command succeeded.
0x01-0xFF	Set_AFH_Host_Channel_Classification command failed. “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Set_AFH_Host_Channel_Classification command has completed, a Command Complete event will be generated.

7.3.51 Read Inquiry Scan Type Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Inquiry_Scan_Type	0x0042		Status, Inquiry_Scan_Type

Description:

This command is used to read the Inquiry_Scan_Type configuration parameter of the local Bluetooth device. See Section 6.4 on page 384.
For details, see the Baseband Specification, "Inquiry scan substate" on page 164 [Part B].

Command Parameters:

None

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Inquiry_Scan_Type command succeeded
0x01-0xFF	Read_Inquiry_Scan_Type command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Inquiry_Scan_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Mandatory: Standard Scan (default)
0x01	Optional: Interlaced Scan
0x02-0xFF	Reserved

Event(s) generated (unless masked away):

When the Read_Inquiry_Scan_Type command has completed, a Command Complete event will be generated.

7.3.52 Write Inquiry Scan Type Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Inquiry_Scan_Type	0x0043	Scan_Type	Status

Description:

This command is used to write the Inquiry Scan Type configuration parameter of the local Bluetooth device. See Section 6.4 on page 384.
For details, see the Baseband Specification, "Inquiry scan substate" on page 164 [Part B].

Command Parameters:

Scan_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Mandatory: Standard Scan (default)
0x01	Optional: Interlaced Scan
0x02-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Inquiry_Scan_Type command succeeded
0x01-0xFF	Write_Inquiry_Scan_Type command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Inquiry_Scan_Type command has completed, a Command Complete event will be generated.

7.3.53 Read Inquiry Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Inquiry_Mode	0x0044		Status, Inquiry_Mode

Description:

This command is used to read the Inquiry_Mode configuration parameter of the local Bluetooth device. See Section 6.5 on page 384.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Inquiry_Mode command succeeded
0x01-0xFF	Read_Inquiry_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes

Inquiry_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	Standard Inquiry Result event format
0x01	Inquiry Result format with RSSI
0x02-0xFF	Reserved

Event(s) generated (unless masked away):

When the Read_Inquiry_Mode command has completed, a Command Complete event will be generated.

7.3.54 Write Inquiry Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Inquiry_Mode	0x0045	Inquiry_Mode	Status

Description:

This command is used to write the Inquiry_Mode configuration parameter of the local Bluetooth device. See Section 6.5 on page 384.

Command Parameters:

Inquiry_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	Standard Inquiry Result event format (default)
0x01	Inquiry Result format with RSSI
0x02-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Inquiry_Mode command succeeded
0x01-0xFF	Write_Inquiry_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Inquiry_Mode command has completed, a Command Complete event will be generated.

7.3.55 Read Page Scan Type Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Page_Scan_Type	0x0046		Status, Page_Scan_Type

Description:

This command is used to read the Page Scan Type configuration parameter of the local Bluetooth device. See Section 6.11 on page 387.
For details, see the Baseband Specification, "Page scan substate" on page 154 [Part B].

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Page_Scan_Type command succeeded
0x01-0xFF	Read_Page_Scan_Type command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Page_Scan_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Mandatory: Standard Scan (default)
0x01	Optional: Interlaced Scan
0x02-0xFF	Reserved

Event(s) generated (unless masked away):

When the Read_Page_Scan_Type command has completed, a Command Complete event will be generated.

7.3.56 Write Page Scan Type Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Page_Scan_Type	0x0047	Page_Scan_Type	Status

Description:

This command is used to write the Page Scan Type configuration parameter of the local Bluetooth device. See Section 6.11 on page 387.
For details, see the Baseband Specification, "Page scan substate" on page 154 [Part B].

Command Parameters:

Page_Scan_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Mandatory: Standard Scan (default)
0x01	Optional: Interlaced Scan
0x02-0xFF	Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Page_Scan_Type command succeeded
0x01-0xFF	Write_Page_Scan_Type command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Page_Scan_Type command has completed, a Command Complete event will be generated.

7.3.57 Read AFH Channel Assessment Mode Command

Command	OCF	Command Parameters	Return Parameters
Read_AFH_Channel_Assessment_Mode	0x0048		Status, AFH_Channel_Assessment_Mode

Description:

The Read_AFH_Channel_Assessment_Mode command reads the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the controller's channel assessment scheme is enabled or disabled.

This command shall be supported by a device that declares support for any of the AFH_capable_master, AFH_classification_slave or AFH_classification_master features.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_AFH_Channel_Assessment_Mode command succeeded.
0x01-0xFF	Read_AFH_Channel_Assessment_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

AFH_Channel_Assessment_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	Controller channel assessment disabled.
0x01	Controller channel assessment enabled.
0x02-0xFF	Reserved for future use.

Event(s) generated (unless masked away):

When the Read_AFH_Channel_Assessment_Mode command has completed, a Command Complete event will be generated.

7.3.58 Write AFH Channel Assessment Mode Command

Command	OCF	Command Parameters	Return Parameters
Write_AFH_Channel _Assessment_Mode	0x0049	AFH_Channel _Assessment_Mode	Status

Description:

The Write_AFH_Channel_Assessment_Mode command writes the value for the AFH_Channel_Assessment_Mode parameter. The AFH_Channel_Assessment_Mode parameter controls whether the Controller's channel assessment scheme is enabled or disabled.

Disabling channel assessment forces all channels to be unknown in the local classification, but does not affect the AFH_reporting_mode or support for the Set_AFH_Host_Channel_Classification command. A slave in the AFH_reporting_enabled state shall continue to send LMP channel classification messages for any changes to the channel classification caused by either this command (altering the AFH_Channel_Assessment_Mode) or HCI Set_AFH_Host_Channel_Classification command (providing a new channel classification from the Host).

This command shall be supported by a device that declares support for any of the AFH_capable_master, AFH_classification_slave or AFH_classification_master features.

If the AFH_Channel_Assessment_Mode parameter is enabled and the Controller does not support a channel assessment scheme, other than via the Set_AFH_Host_Channel_Classification command, then a Status parameter of 'Channel Assessment Not Supported' should be returned. See "Error Codes" on page 319 [Part D] for list of Error Codes.

If the Controller supports a channel assessment scheme then the default AFH_Channel_Assessment_Mode is enabled, otherwise the default is disabled.

Command Parameters:

AFH_Channel_Assessment_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	Controller channel assessment disabled.
0x01	Controller channel assessment enabled.
0x02-0xFF	Reserved for future use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_AFH_Channel_Assessment_Mode command succeeded.
0x01-0xFF	Write_AFH_Channel_Assessment_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_AFH_Channel_Assessment_Mode command has completed, a Command Complete event will be generated.

7.4 INFORMATIONAL PARAMETERS

The Informational Parameters are fixed by the manufacturer of the Bluetooth hardware. These parameters provide information about the Bluetooth device and the capabilities of the Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters. For Informational Parameters Commands, the OGF is defined as 0x04.

7.4.1 Read Local Version Information Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Version_ Information	0x0001		Status, HCI Version, HCI Revision, LMP Version, Manufacturer_Name, LMP Subversion

Description:

This command will read the values for the version information for the local Bluetooth device.

The HCI Version information defines the version information of the HCI layer. The LMP Version information defines the version of the LMP. The Manufacturer_Name information indicates the manufacturer of the local device.

The HCI Revision and LMP Subversion are implementation dependant.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Local_Version_Information command succeeded.
0x01-0xFF	Read_Local_Version_Information command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

HCI_Version: *Size: 1 Octet*

Value	Parameter Description
See https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers	

HCI_Revision: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Revision of the Current HCI in the Bluetooth device.

LMP_Version: *Size: 1 Octet*

Value	Parameter Description
0xXX	Version of the Current LMP in the Bluetooth device, See https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers

Manufacturer_Name: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Manufacturer Name of the Bluetooth device, See https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers

LMP_Subversion: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Subversion of the Current LMP in the Bluetooth device, see Table 5.2 on page 303 in the Link Manager Protocol for assigned values (SubVersNr).

Event(s) generated (unless masked away):

When the Read_Local_Version_Information command has completed, a Command Complete event will be generated.

7.4.2 Read Local Supported Commands Command

Command	OCF	Command Parameters	Return Parameters
HCI_ Read_Local_Supported_Commands	0x0002		Status, Supported Commands

Description:

This command reads the list of HCI commands supported for the local device.

This command will return the Supported Commands configuration parameter. It is implied that if a command is listed as supported, the feature underlying that command is also supported.

See Section 6.26 on page 395 for more information.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0	Read Local Supported Commands command succeeded
0x01-0xff	Read Local Supported Commands command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Supported Commands: *Size: 64 Octets*

Value	Parameter Description
	Bit mask for each HCI Command. If a bit is 1, the radio supports the corresponding command and the features required for the command. Unsupported or undefined commands shall be set to 0. See Section 6.26, "Supported Commands," on page 395.

Event(s) generated (unless masked away):

When the Read Local Supported Commands command has completed a Command Complete event will be generated.

7.4.3 Read Local Supported Features Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_Supported_Features	0x0003		Status, LMP_Features

Description:

This command requests a list of the supported features for the local device. This command will return a list of the LMP features. For details see "Link Manager Protocol" on page 211 [Part C].

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Local_Supported_Features command succeeded.
0x01-0xFF	Read_Local_Supported_Features command failed. See "Error Codes" on page 319 [Part D]

LMP_Features: *Size: 8 Octets*

Value	Parameter Description
0xXXXXXXXX XXXXXXXX	Bit Mask List of LMP features. For details see "Link Manager Protocol" on page 211 [Part C]

Event(s) generated (unless masked away):

When the Read_Local_Supported_Features command has completed, a Command Complete event will be generated.

7.4.4 Read Local Extended Features Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Local_ Extended_Features	0x0004	Page number	Status, Page number, Maximum Page Number, Extended_LMP_Features

Description:

The HCI_Read_Local_Extended_Features command returns the requested page of the extended LMP features.

Command Parameters:

Page Number: *Size: 1 Octet*

Value	Parameter Description
0x00	Requests the normal LMP features as returned by HCI_Read_Local_Supported_Features
0x01-0xFF	Return the corresponding page of features

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	HCI_Read_Local_Extended_Features command succeeded
0x01-0xFF	HCI_Read_Local_Extended_Features command failed. See "Error Codes" on page 319 [Part D] for list of error codes

Page Number: *Size: 1 Octet*

Value	Parameter Description
0x00	The normal LMP features as returned by HCI_Read_Local_Supported_Features
0x01-0xFF	The page number of the features returned

Maximum Page Number: *Size: 1 Octet*

Value	Parameter Description
0x00-0xFF	The highest features page number which contains non-zero bits for the local device

Extended_LMP_Features: *Size: 8 Octets*

Value	Parameter Description
0xFFFFFFFFFFFFFFFF	Bit map of requested page of LMP features. See LMP specification for details

Event(s) generated (unless masked away):

When the Controller receives the HCI_Read_Local_Extended_Features command the Controller sends the Command Complete command to the Host containing the requested information.

7.4.5 Read Buffer Size Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Buffer_Size	0x0005		Status, HC_ACL_Data_Packet_Length, HC_Synchronous_Data_Packet_Length, HC_Total_Num_ACL_Data_Packets, HC_Total_Num_Synchronous_Data_Packets

Description:

The Read_Buffer_Size command is used to read the maximum size of the data portion of HCI ACL and synchronous Data Packets sent from the Host to the Controller. The Host will segment the data to be transmitted from the Host to the Controller according to these sizes, so that the HCI Data Packets will contain data with up to these sizes. The Read_Buffer_Size command also returns the total number of HCI ACL and synchronous Data Packets that can be stored in the data buffers of the Controller. The Read_Buffer_Size command must be issued by the Host before it sends any data to the Controller.

The HC_ACL_Data_Packet_Length return parameter will be used to determine the size of the L2CAP segments contained in ACL Data Packets, which are transferred from the Host to the Controller to be broken up into baseband packets by the Link Manager. The HC_Synchronous_Data_Packet_Length return parameter is used to determine the maximum size of HCI synchronous Data Packets. Both the Host and the Controller must support command and event packets, where the data portion (excluding header) contained in the packets is 255 octets in size. The HC_Total_Num_ACL_Data_Packets return parameter contains the total number of HCI ACL Data Packets that can be stored in the data buffers of the Controller. The Host will determine how the buffers are to be divided between different Connection Handles. The HC_Total_Num_Synchronous_Data_Packets return parameter gives the same information but for HCI synchronous Data Packets.
Note: the HC_ACL_Data_Packet_Length and HC_Synchronous_Data_Packet_Length return parameters do not include the length of the HCI Data Packet header.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Buffer_Size command succeeded.
0x01-0xFF	Read_Buffer_Size command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

HC_ACL_Data_Packet_Length: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Maximum length (in octets) of the data portion of each HCI ACL Data Packet that the Controller is able to accept.

HC_Synchronous_Data_Packet_Length: *Size: 1 Octet*

Value	Parameter Description
0xXX	Maximum length (in octets) of the data portion of each HCI Synchronous Data Packet that the Controller is able to accept.

HC_Total_Num_ACL_Data_Packets: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Total number of HCI ACL Data Packets that can be stored in the data buffers of the Controller.

HC_Total_Num_Synchronous_Data_Packets: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Total number of HCI Synchronous Data Packets that can be stored in the data buffers of the Controller.

Event(s) generated (unless masked away):

When the Read_Buffer_Size command has completed, a Command Complete event will be generated.

7.4.6 Read BD_ADDR Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_BD_ADDR	0x0009		Status, BD_ADDR

Description:

This command shall read the Bluetooth device address (BD_ADDR). See the "Baseband Specification" on page 55 [Part B] for details of how BD_ADDR is used.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_BD_ADDR command succeeded.
0x01-0xFF	Read_BD_ADDR command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device

Event(s) generated (unless masked away):

When the Read_BD_ADDR command has completed, a Command Complete event will be generated.

7.5 STATUS PARAMETERS

The Controller modifies all status parameters. These parameters provide information about the current state of the Controller, Link Manager, and Baseband. The host device cannot modify any of these parameters other than to reset certain specific parameters. For the Status and baseband, the OGF is defined as 0x05.

7.5.1 Read Failed Contact Counter Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Failed_ Contact_Counter	0x0001	Connection_Handle	Status, Connection_Handle, Failed_Contact_Counter

Description:

This command will read the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Connection_Handle must be a Connection_Handle for an ACL connection.
See Section 6.17 on page 390.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter should be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Failed_Contact_Counter command succeeded.
0x01-0xFF	Read_Failed_Contact_Counter command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter has been read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Failed_Contact_Counter: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Number of consecutive failed contacts for a connection corresponding to the connection handle.

Event(s) generated (unless masked away):

When the Read_Failed_Contact_Counter command has completed, a Command Complete event will be generated.

7.5.2 Reset Failed Contact Counter Command

Command	OCF	Command Parameters	Return Parameters
HCI_Reset_Failed_ Contact_Counter	0x0002	Connection_Handle	Status, Connection_Handle

Description:

This command will reset the value for the Failed_Contact_Counter parameter for a particular connection to another device. The Connection_Handle must be a Connection_Handle for an ACL connection.
See Section 6.17 on page 390.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter should be reset. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Reset_Failed_Contact_Counter command succeeded.
0x01-0xFF	Reset_Failed_Contact_Counter command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Failed Contact Counter has been reset. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Event(s) generated (unless masked away):

When the Reset_Failed_Contact_Counter command has completed, a Command
Complete event will be generated.

7.5.3 Read Link Quality Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Link_Quality	0x0003	Connection_Handle	Status, Connection_Handle, Link_Quality

Description:

This command will return the value for the Link_Quality for the specified Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection. This command will return a Link_Quality value from 0-255, which represents the quality of the link between two Bluetooth devices. The higher the value, the better the link quality is. Each Bluetooth module vendor will determine how to measure the link quality.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the connection for which link quality parameters are to be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Link_Quality command succeeded.
0x01-0xFF	Read_Link_Quality command failed.See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the connection for which the link quality parameter has been read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Link_Quality: *Size: 1 Octet*

Value	Parameter Description
0xXX	The current quality of the Link connection between the local device and the remote device specified by the Connection Handle Range: 0x00 – 0xFF The higher the value, the better the link quality is.

Event(s) generated (unless masked away):

When the Read_Link_Quality command has completed, a Command Complete event will be generated.

7.5.4 Read RSSI Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_RSSI	0x0005	Connection_Handle	Status, Connection_Handle, RSSI

Description:

This command will read the value for the difference between the measured Received Signal Strength Indication (RSSI) and the limits of the Golden Receive Power Range (see Radio Specification Section 4.1.6 on page 43) for a connection handle to another Bluetooth device. The Connection_Handle must be a Connection_Handle for an ACL connection. Any positive RSSI value returned by the Controller indicates how many dB the RSSI is above the upper limit, any negative value indicates how many dB the RSSI is below the lower limit. The value zero indicates that the RSSI is inside the Golden Receive Power Range.
Note: how accurate the dB values will be depends on the Bluetooth hardware. The only requirements for the hardware are that the Bluetooth device is able to tell whether the RSSI is inside, above or below the Golden Device Power Range.

The RSSI measurement compares the received signal power with two threshold levels, which define the Golden Receive Power Range. The lower threshold level corresponds to a received power between -56 dBm and 6 dB above the actual sensitivity of the receiver. The upper threshold level is 20 dB above the lower threshold level to an accuracy of +/- 6 dB.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the RSSI is to be read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_RSSI command succeeded.
0x01-0xFF	Read_RSSI command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the RSSI has been read. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

RSSI: *Size: 1 Octet*

Value	Parameter Description
N = 0xXX	Size: 1 Octet (signed integer) Range: -128 ≤ N ≤ 127 Units: dB

Event(s) generated (unless masked away):

When the Read_RSSI command has completed, a Command Complete event will be generated.

7.5.5 Read AFH Channel Map Command

Command	OCF	Command Parameters	Return Parameters
Read_AFH _Channel_Map	0x0006	Connection_Handle	Status, Connection_Handle, AFH_Mode, AFH_Channel_Map

Description:

This command will return the values for the AFH_Mode and AFH_Channel_Map for the specified Connection Handle. The Connection_Handle must be a Connection_Handle for an ACL connection.

The returned values indicate the state of the hop sequence specified by the most recent LMP_Set_AFH message for the specified Connection_Handle, regardless of whether the master has received the baseband ACK or whether the AFH_Instant has passed.

This command shall be supported by a device that declares support for either the AFH_capable_slave or AFH_capable_master feature.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Channel Map is to be read. Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use)

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_AFH_Channel_Map command succeeded.
0x01-0xFF	Read_AFH_Channel_Map command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the Channel Map is to be read. Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use)

AFH_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	AFH disabled.
0x01	AFH enabled.
0x02-0xFF	Reserved for future use.

AFH_Channel_Map: *Size: 10 Octets (79 Bits meaningful)*

Value	Parameter Description
0xXXXXXXXX XXXXXXXX XXX	If AFH_Mode is AFH enabled then this parameter contains 79 1-bit fields, otherwise the contents are reserved. The n^{th} such field (in the range 0 to 78) contains the value for channel *n*: Channel *n* is unused = 0 Channel *n* is used = 1 Range: 0x00000000000000000000-0x7FFFFFFFFFFFFFFFFFFF (0x80000000000000000000-0xFFFFFFFFFFFFFFFFFFFF Reserved for future use)

Event(s) generated (unless masked away):

When the Read_AFH_Channel_Map command has completed, a Command Complete event will be generated.

7.5.6 Read Clock Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Clock	0x0007	Connection_Handle Which_Clock	Status, Connection_Handle, Clock, Accuracy

Description:

This command will read the estimate of the value of the Bluetooth Clock.

If the Which_Clock value is 0, then the Connection_Handle shall be ignored, and the local Bluetooth Clock value shall be returned, and the accuracy parameter shall be set to 0.

If the Which_Clock value is 1, then the Connection_Handle must be a valid ACL connection handle. If the current role of this ACL connection is Master, then the Bluetooth Clock of this device shall be returned. If the current role is Slave, then an estimate of the Bluetooth Clock of the remote master and the accuracy of this value shall be returned.

The accuracy reflects the clock drift that might have occurred since the slave last received a valid transmission from the master.

Note: The Bluetooth Clock has a minimum accuracy of 250ppm, or about 22 seconds drift in one day.

Note: See Baseband Specification, Section 1.1, on page 64 for more information about the Bluetooth Clock.

Command Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify which connection to be used for reading the masters Bluetooth Clock. Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

Which_Clock: *Size 1 Octet)*

Value	Parameter Description
0xXX	0x00 = Local Clock (Connection Handle does not have to be a valid handle) 0x01 = Piconet Clock (Connection Handle shall be a valid ACL Handle) 0x02 to 0xFF = Reserved

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_BD_CLOCK command succeeded.
0x01 – 0xFF	Read_BD_CLOCK command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The Connection Handle for the Connection for which the master clock has been read. If the Which_Clock was 0, then the Connection_Handle shall be set to 0 and ignored upon receipt Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

Clock: *Size: 4 Octets (28 bits meaningful)*

Value	Parameter Description
0xXXXXXXXX	Bluetooth Clock of the device requested.

Accuracy: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	+/- maximum Bluetooth Clock error. Value of 0xFFFF means Unknown. Accuracy = +/- N * 0.3125 msec (1 Bluetooth Clock) Range for N: 0x0000 - 0xFFFE Time Range for N: 0 - 20479.375 msec

Event(s) generated (unless masked away):

When the Read_Clock command has completed, a Command Complete event will be generated.

7.6 TESTING COMMANDS

The Testing commands are used to provide the ability to test various functionalities of the Bluetooth hardware. These commands provide the ability to arrange various conditions for testing. For the Testing Commands, the OGF is defined as 0x06.

7.6.1 Read Loopback Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Read_Loopback_Mode	0x0001		Status, Loopback_Mode

Description:

This command will read the value for the setting of the Controller's Loopback Mode. The setting of the Loopback Mode will determine the path of information. In Non-testing Mode operation, the Loopback Mode is set to Non-testing Mode and the path of the information is as specified by the Bluetooth specifications. In Local Loopback Mode, every Data Packet (ACL, SCO and eSCO) and Command Packet that is sent from the Host to the Controller is sent back with no modifications by the Controller, as shown in Figure 7.1 on page 560.
]For details of loopback modes see Section 7.6.2 on page 559.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Loopback_Mode command succeeded.
0x01-0xFF	Read_Loopback_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Loopback_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	No Loopback mode enabled. **Default.**
0x01	Enable Local Loopback.
0x02	Enable Remote Loopback.
0x03-0xFF	Reserved for Future Use.

Event(s) generated (unless masked away):

When the Read_Loopback_Mode command has completed, a Command Complete event will be generated.

7.6.2 Write Loopback Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Write_Loopback_Mode	0x0002	Loopback_Mode	Status

Description:

This command will write the value for the setting of the Controller's Loopback Mode. The setting of the Loopback Mode will determine the path of information. In Non-testing Mode operation, the Loopback Mode is set to Non-testing Mode and the path of the information as specified by the Bluetooth specifications. In Local Loopback Mode, every Data Packet (ACL, SCO and eSCO) and Command Packet that is sent from the Host to the Controller is sent back with no modifications by the Controller, as shown in Figure 7.1 on page 560.

When the Bluetooth Host Controller enters Local Loopback Mode, it shall respond with one to four connection handles, one for an ACL connection and zero to three for synchronous connections. The Host should use these connection handles when sending data in Local Loopback Mode. The number of connection handles returned for synchronous connections (between zero and three) is implementation specific. When in Local Loopback Mode, the Controller loops back commands and data to the Host. The Loopback Command event is used to loop back commands that the Host sends to the Controller.

There are some commands that are not looped back in Local Loopback Mode: Reset, Set_Controller_To_Host_Flow_Control, Host_Buffer_Size, Host_Number_Of_Completed_Packets, Read_Buffer_Size, Read_Loopback_Mode and Write_Loopback_Mode. These commands should be executed in the way they are normally executed. The commands Reset and Write_Loopback_Mode can be used to exit local loopback mode.

If Write_Loopback_Mode is used to exit Local Loopback Mode, Disconnection Complete events corresponding to the Connection Complete events that were sent when entering Local Loopback Mode should be sent to the Host. Furthermore, no connections are allowed in Local Loopback mode. If there is a connection, and there is an attempt to set the device to Local Loopback Mode, the attempt will be refused. When the device is in Local Loopback Mode, the Controller will refuse incoming connection attempts. This allows the Host Controller Transport Layer to be tested without any other variables.

If a device is set to Remote Loopback Mode, it will send back all data (ACL, SCO and eSCO) that comes over the air. It will only allow a maximum of one ACL connection and three synchronous connections, and these shall all be to the same remote device. If there are existing connections to a remote device and there is an attempt to set the local device to Remote Loopback Mode, the attempt shall be refused.

See Figure 7.2 on page 560, where the rightmost device is set to Remote Loopback Mode and the leftmost device is set to Non-testing Mode. This allows the Bluetooth Air link to be tested without any other variables.

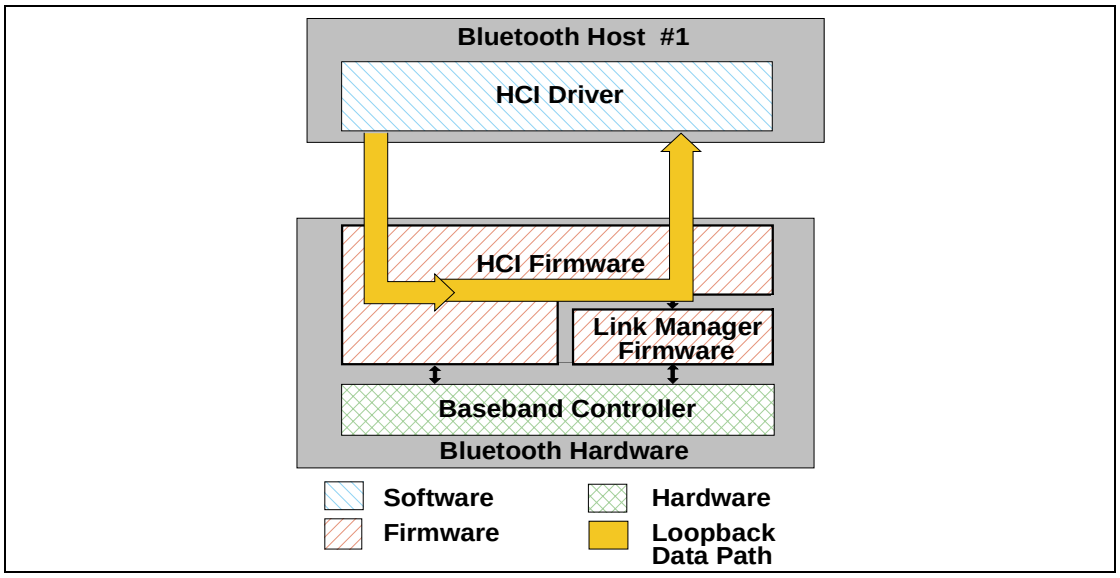


Figure 7.1: Local Loopback Mode

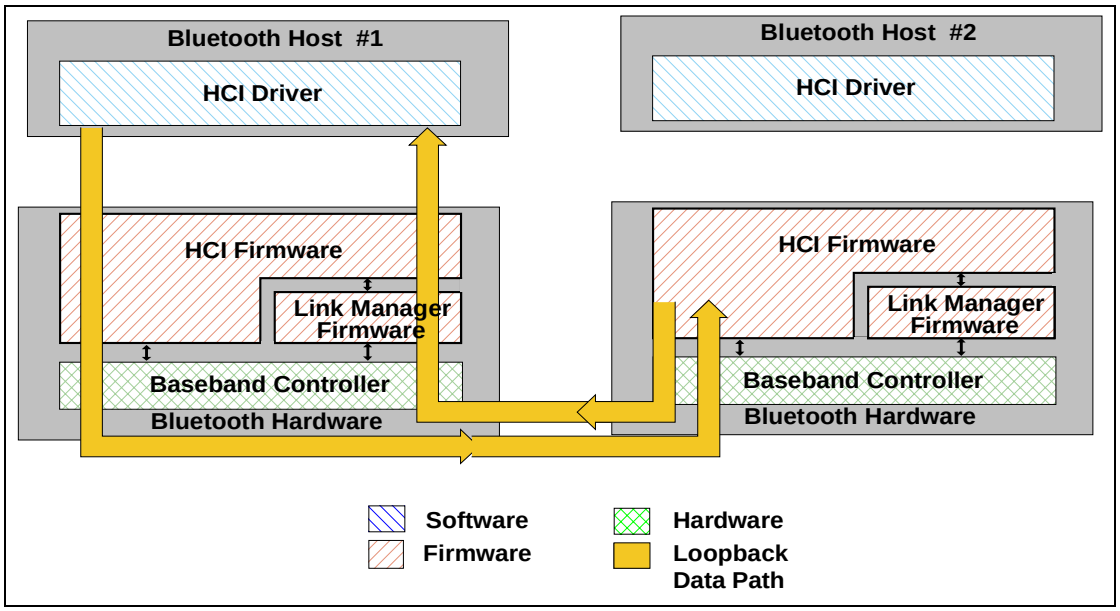


Figure 7.2: Remote Loopback Mode

Command Parameters:

Loopback_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	No Loopback mode enabled. **Default.**
0x01	Enable Local Loopback.
0x02	Enable Remote Loopback.
0x03-0xFF	Reserved for Future Use.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Loopback_Mode command succeeded.
0x01-0xFF	Write_Loopback_Mode command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Loopback_Mode command has completed, a Command Complete event will be generated.

7.6.3 Enable Device Under Test Mode Command

Command	OCF	Command Parameters	Return Parameters
HCI_Enable_Device_Under_ Test_Mode	0x0003		Status

Description:

The Enable_Device_Under_Test_Mode command will allow the local Bluetooth module to enter test mode via LMP test commands. For details see “Link Manager Protocol” on page 211 [Part C]. The Host issues this command when it wants the local device to be the DUT for the Testing scenarios as described in the “Test Methodology” on page 231[vol. 4]. When the Controller receives this command, it will complete the command with a Command Complete event. The Controller functions as normal until the remote tester issues the LMP test command to place the local device into Device Under Test mode. To disable and exit the Device Under Test Mode, the Host can issue the HCI_Reset command. This command prevents remote Bluetooth devices from causing the local Bluetooth device to enter test mode without first issuing this command.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Enter_Device_Under_Test_Mode command succeeded.
0x01-0xFF	Enter_Device_Under_Test_Mode command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Enter_Device_Under_Test_Mode command has completed, a Command Complete event will be generated.

7.7 EVENTS

7.7.1 Inquiry Complete Event

Event	Event Code	Event Parameters
Inquiry Complete	0x01	Status

Description:

The Inquiry Complete event indicates that the Inquiry is finished. This event contains a status parameter, which is used to indicate if the Inquiry completed successfully or if the Inquiry was not completed.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Inquiry command completed successfully.
0x01-0xFF	Inquiry command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

7.7.2 Inquiry Result Event

Event	Event Code	Event Parameters
Inquiry Result	0x02	Num_Responses, BD_ADDR[i], Page_Scan_Repetition_Mode[i], Reserved[i], Reserved[i], Class_of_Device[i] Clock_Offset[i]

Description:

The Inquiry Result event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process. This event will be sent from the Controller to the Host as soon as an Inquiry Response from a remote device is received if the remote device supports only mandatory paging scheme. The Controller may queue these Inquiry Responses and send multiple Bluetooth devices information in one Inquiry Result event. The event can be used to return one or more Inquiry responses in one event.

Event Parameters:

Num_Responses: *Size: 1 Octet*

Value	Parameter Description
0xXX	Number of responses from the Inquiry.

BD_ADDR[i]: *Size: 6 Octets * Num_Responses*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for each device which responded.

Page_Scan_Repetition_Mode[i]: *Size: 1 Octet * Num_Responses*

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03 – 0xFF	Reserved

Reserved[i]: [1] *Size: 1 Octet * Num_Responses*

Value	Parameter Description
0xXX	Reserved.

Reserved[i]: [2] *Size: 1 Octet * Num_Responses*

Value	Parameter Description
0xXX	Reserved, must be set to 0x00.

Class_of_Device[i]: *Size: 3 Octets * Num_Responses*

Value	Parameter Description
0xXXXXXX	Class of Device for the device

Clock_Offset[i]: *Size: 2 Octets * Num_Responses*

Bit format	Parameter Description
Bit 14-0	Bit 16-2 of CLKslave-CLKmaster.
Bit 15	Reserved

1. This was the Page_Scan_Period_Mode parameter in the v1.1 specification. This parameter has no meaning in v1.2 and no default value.
2. This was the Page_Scan_Mode parameter in the v1.1 specification.

7.7.3 Connection Complete Event

Event	Event Code	Event Parameters
Connection Complete	0x03	Status, Connection_Handle, BD_ADDR, Link_Type, Encryption_Mode

Description:

The Connection Complete event indicates to both of the Hosts forming the connection that a new connection has been established. This event also indicates to the Host, which issued the Create Connection, or Accept_Connection_Request or Reject_Connection_Request command and then received a Command Status event, if the issued command failed or was successful.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Connection successfully completed.
0x01-0xFF	Connection failed to Complete. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection between two Bluetooth devices. The Connection Handle is used as an identifier for transmitting and receiving voice or data. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the other connected Device forming the connection.

Link_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	SCO connection.
0x01	ACL connection (Data Channels).
0x02-0xFF	Reserved for future use.

Encryption_Mode: *Size: 1 Octet*

Value	Parameter Description
See Section 6.16 on page 389.	

7.7.4 Connection Request Event

Event	Event Code	Event Parameters
Connection Request	0x04	BD_ADDR, Class_of_Device, Link_Type

Description:

The Connection Request event is used to indicate that a new incoming connection is trying to be established. The connection may either be accepted or rejected. If this event is masked away and there is an incoming connection attempt and the Controller is not set to auto-accept this connection attempt, the Controller will automatically refuse the connection attempt. When the Host receives this event and the link type parameter is ACL, it should respond with either an Accept_Connection_Request or Reject_Connection_Request command before the timer Conn_Accept_Timeout expires. If the link type is SCO or eSCO, the Host should reply with the Accept_Synchronous_Connection_Request or the Reject_Synchronous_Connection_Request Command. If the link type is SCO, the host may respond with Accept_Connection_Request. If the Event is responded to with Accept_Connection_Request, then the default parameter settings of the Accept_Synchronous_Connection_Request (see Section 7.1.27 on page 442) should be used by the local LM when negotiating the SCO or eSCO link parameters. In that case, the Connection_Complete Event and not the Synchronous_Connection_Complete Event, shall be returned on completion of the connection.

Event Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the device that requests the connection.

Class_of_Device: *Size: 3 Octets*

Value	Parameter Description
0xXXXXXX	Class of Device for the device, which requests the connection.
0x000000	Unknown Class of Device

Link_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	SCO Connection requested
0x01	ACL Connection requested
0x02	eSCO Connection requested
0x03-0xFF	Reserved for Future Use.

7.7.5 Disconnection Complete Event

Event	Event Code	Event Parameters
Disconnection Complete	0x05	Status, Connection_Handle, Reason

Description:

The Disconnection Complete event occurs when a connection is terminated. The status parameter indicates if the disconnection was successful or not. The reason parameter indicates the reason for the disconnection if the disconnection was successful. If the disconnection was not successful, the value of the reason parameter can be ignored by the Host. For example, this can be the case if the Host has issued the Disconnect command and there was a parameter error, or the command was not presently allowed, or a connection handle that didn't correspond to a connection was given.
Note: When a physical link fails, one Disconnection Complete event will be returned for each logical channel on the physical link with the corresponding Connection handle as a parameter.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Disconnection has occurred.
0x01-0xFF	Disconnection failed to complete. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which was disconnected. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Reason: *Size: 1 Octet*

Value	Parameter Description
0xXX	Reason for disconnection. See "Error Codes" on page 319 [Part D] for error codes and description.

7.7.6 Authentication Complete Event

Event	Event Code	Event Parameters
Authentication Complete	0x06	Status, Connection_Handle

Description:

The Authentication Complete event occurs when authentication has been completed for the specified connection. The Connection_Handle must be a Connection_Handle for an ACL connection.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Authentication Request successfully completed.
0x01-0xFF	Authentication Request failed to complete. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle for which Authentication has been performed. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

7.7.7 Remote Name Request Complete Event

Event	Event Code	Event Parameters
Remote Name Request Complete	0x07	Status, BD_ADDR, Remote_Name

Description:

The Remote Name Request Complete event is used to indicate that a remote name request has been completed. The Remote_Name event parameter is a UTF-8 encoded string with up to 248 octets in length. The Remote_Name event parameter will be null-terminated (0x00) if the UTF-8 encoded string is less than 248 octets. The BD_ADDR event parameter is used to identify which device the user-friendly name was obtained from.
Note: the Remote_Name Parameter is a string parameter. Endianess does therefore not apply to the Remote_Name Parameter. The first octet of the name is received first.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Remote_Name_Request command succeeded.
0x01-0xFF	Remote_Name_Request command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the device whose name was requested.

Remote_Name: *Size: 248 Octets*

Value	Parameter Description
Name[248]	A UTF-8 encoded user-friendly descriptive name for the remote device. If the name contained in the parameter is shorter than 248 octets, the end of the name is indicated by a NULL octet (0x00), and the following octets (to fill up 248 octets, which is the length of the parameter) do not have valid values.

7.7.8 Encryption Change Event

Event	Event Code	Event Parameters
Encryption Change	0x08	Status, Connection_Handle, Encryption_Enable

Description:

The Encryption Change event is used to indicate that the change in the encryption has been completed for the Connection Handle specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. The Encryption_Enable event parameter specifies the new Encryption Enable for the Connection Handle specified by the Connection_Handle event parameter. This event will occur on both devices to notify the Hosts when Encryption has changed for the specified Connection Handle between two devices.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Encryption Change has occurred.
0x01-0xFF	Encryption Change failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle for which the link layer encryption has been enabled/disabled for all Connection Handles with the same Bluetooth device endpoint as the specified Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Encryption_Enable: *Size: 1 Octet*

Value	Parameter Description
0x00	Link Level Encryption is OFF.
0x01	Link Level Encryption is ON.

7.7.9 Change Connection Link Key Complete Event

Event	Event Code	Event Parameters
Change Connection Link Key Complete	0x09	Status, Connection_Handle

Description:

The Change Connection Link Key Complete event is used to indicate that the change in the Link Key for the Connection Handle specified by the Connection_Handle event parameter has been completed.
The Connection_Handle will be a Connection_Handle for an ACL connection.
The Change Connection Link Key Complete event is sent only to the Host which issued the Change_Connection_Link_Key command.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Change_Connection_Link_Key command succeeded.
0x01-0xFF	Change_Connection_Link_Key command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which the Link Key has been change for all Connection Handles with the same Bluetooth device end point as the specified Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

7.7.10 Master Link Key Complete Event

Event	Event Code	Event Parameters
Master Link Key Complete	0x0A	Status, Connection_Handle, Key_Flag

Description:

The Master Link Key Complete event is used to indicate that the Link Key managed by the master of the piconet has been changed. The Connection_Handle will be a Connection_Handle for an ACL connection. The link key used for the connection will be the temporary link key of the master device or the semi-permanent link key indicated by the Key_Flag. The Key_Flag event parameter is used to indicate which Link Key (temporary link key of the Master, or the semi-permanent link keys) is now being used in the piconet.
Note: for a master, the change from a semi-permanent Link Key to temporary Link Key will affect all Connection Handles related to the piconet. For a slave, this change affects only this particular connection handle. A temporary link key must be used when both broadcast and point-to-point traffic shall be encrypted.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Master_Link_Key command succeeded.
0x01-0xFF	Master_Link_Key command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle for which the Link Key has been changed for all devices in the same piconet. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Key_Flag: *Size: 1 Octet*

Value	Parameter Description
0x00	Using Semi-permanent Link Key.
0x01	Using Temporary Link Key.

7.7.11 Read Remote Supported Features Complete Event

Event	Event Code	Event Parameters
Read Remote Supported Features Complete	0x0B	Status, Connection_Handle, LMP_Features

Description:

The Read Remote Supported Features Complete event is used to indicate the completion of the process of the Link Manager obtaining the supported features of the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. The event parameters include a list of LMP features. For details see "Link Manager Protocol" on page 211[vol. 3].

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Remote_Supported_Features command succeeded.
0x01-0xFF	Read_Remote_Supported_Features command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which is used for the Read_Remote_Supported_Features command. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

LMP_Features: *Size: 8 Octets*

Value	Parameter Description
0xXXXXXXXX XXXXXXXX	Bit Mask List of LMP features. See "Link Manager Protocol" on page 211 [Part C].

7.7.12 Read Remote Version Information Complete Event

Event	Event Code	Event Parameters
Read Remote Version Information Complete	0x0C	Status, Connection_Handle, LMP_Version, Manufacturer_Name, LMP_Subversion

Description:

The Read Remote Version Information Complete event is used to indicate the completion of the process of the Link Manager obtaining the version information of the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. The LMP_Version event parameter defines the specification version of the Bluetooth device. The Manufacturer_Name event parameter indicates the manufacturer of the remote Bluetooth device. The LMP_Subversion event parameter is controlled by the manufacturer and is implementation dependant. The LMP_Subversion event parameter defines the various revisions that each version of the Bluetooth hardware will go through as design processes change and errors are fixed. This allows the software to determine what Bluetooth hardware is being used and, if necessary, to work around various bugs in the hardware.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Remote_Version_Information command succeeded.
0x01-0xFF	Read_Remote_Version_Information command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which is used for the Read_Remote_Version_Information command. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

LMP_Version: *Size: 1 Octet*

Value	Parameter Description
0xXX	Version of the Current LMP in the remote Bluetooth device. For LMP_Version information see https://www.bluetooth.org/foundry/assign-numb/document/assigned_numbers

Manufacturer_Name: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Manufacturer Name of the remote Bluetooth device, see Table 5.2 on page 303 in the Link Manager Protocol for assigned values (CompId).

LMP_Subversion: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Subversion of the Current LMP in the remote Bluetooth device, see Table 5.2 on page 303 in the Link Manager Protocol for assigned values (SubVersNr).

7.7.13 QoS Setup Complete Event

Event	Event Code	Event Parameters
QoS Setup Complete	0x0D	Status, Connection_Handle, Flags, Service_Type, Token_Rate, Peak_Bandwidth, Latency, Delay_Variation

Description:

The QoS Setup Complete event is used to indicate the completion of the process of the Link Manager setting up QoS with the remote Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection. For more detail see "Logical Link Control and Adaptation Protocol Specification" on page 15[vol. 4].

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	QoS_Setup command succeeded.
0x01-0xFF	QoS_Setup command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which is used for the QoS_Setup command. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Flags: *Size: 1 Octet*

Value	Parameter Description
0x00 – 0xFF	Reserved for Future Use.

Service_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	No Traffic Available.
0x01	Best Effort Available.
0x02	Guaranteed Available.
0x03-0xFF	Reserved for Future Use.

Token_Rate: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Available Token Rate, in octets per second.

Peak_Bandwidth: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Available Peak Bandwidth, in octets per second.

Latency: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Available Latency, in microseconds.

Delay_Variation: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Available Delay Variation, in microseconds.

7.7.14 Command Complete Event

Event	Event Code	Event Parameters
Command Complete	0x0E	Num_HCI_Command_Packets, Command_Opcode, Return_Parameters

Description:

The Command Complete event is used by the Controller for most commands to transmit return status of a command and the other event parameters that are specified for the issued HCI command.

The Num_HCI_Command_Packets event parameter allows the Controller to indicate the number of HCI command packets the Host can send to the Controller. If the Controller requires the Host to stop sending commands, the Num_HCI_Command_Packets event parameter will be set to zero. To indicate to the Host that the Controller is ready to receive HCI command packets, the Controller generates a Command Complete event with the Command_Opcode 0x0000, and the Num_HCI_Command_Packets event parameter is set to 1 or more. Command_Opcode, 0x0000 is a NOP (No OPeration), and can be used to change the number of outstanding HCI command packets that the Host can send before waiting. See each command for the parameters that are returned by this event.

Event Parameters:

Num_HCI_Command_Packets: *Size: 1 Octet*

Value	Parameter Description
N = 0xXX	The Number of HCI command packets which are allowed to be sent to the Controller from the Host. Range for N: 0 – 255

Command_Opcode: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Opcode of the command which caused this event.

Return_Parameter(s): *Size: Depends on Command*

Value	Parameter Description
0xXX	This is the return parameter(s) for the command specified in the Command_Opcode event parameter. See each command's definition for the list of return parameters associated with that command.

7.7.15 Command Status Event

Event	Event Code	Event Parameters
Command Status	0x0F	Status, Num_HCI_Command_Packets, Command_Opcode

Description:

The Command Status event is used to indicate that the command described by the Command_Opcode parameter has been received, and that the Controller is currently performing the task for this command. This event is needed to provide mechanisms for asynchronous operation, which makes it possible to prevent the Host from waiting for a command to finish. If the command can not begin to execute (a parameter error may have occurred, or the command may currently not be allowed), the Status event parameter will contain the corresponding error code, and no complete event will follow since the command was not started. The Num_HCI_Command_Packets event parameter allows the Controller to indicate the number of HCI command packets the Host can send to the Controller. If the Controller requires the Host to stop sending commands, the Num_HCI_Command_Packets event parameter will be set to zero. To indicate to the Host that the Controller is ready to receive HCI command packets, the Controller generates a Command Status event with Status 0x00 and Command_Opcode 0x0000, and the Num_HCI_Command_Packets event parameter is set to 1 or more. Command_Opcode, 0x0000 is a NOP (No OPeration) and can be used to change the number of outstanding HCI command packets that the Host can send before waiting.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Command currently in pending.
0x01-0xFF	Command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Num_HCI_Command_Packets: *Size: 1 Octet*

Value	Parameter Description
N = 0xXX	The Number of HCI command packets which are allowed to be sent to the Controller from the Host. Range for N: 0 – 255

Command_Opcode: *Size: 2 Octets*

Value	Parameter Description
0xXXXX	Opcode of the command which caused this event and is pending completion.

7.7.16 Hardware Error Event

Event	Event Code	Event Parameters
Hardware Error	0x10	Hardware_Code

Description:

The Hardware Error event is used to indicate some type of hardware failure for the Bluetooth device. This event is used to notify the Host that a hardware failure has occurred in the Bluetooth device.

Event Parameters:

Hardware_Code: *Size: 1 Octet*

Value	Parameter Description
0x00-0xFF	These Hardware_Codes will be implementation-specific, and can be assigned to indicate various hardware problems.

7.7.17 Flush Occurred Event

Event	Event Code	Event Parameters
Flush Occurred	0x11	Connection_Handle

Description:

The Flush Occurred event is used to indicate that, for the specified Connection Handle, the current user data to be transmitted has been removed. The Connection_Handle will be a Connection_Handle for an ACL connection. This could result from the flush command, or be due to the automatic flush. Multiple blocks of an L2CAP packet could have been pending in the Controller. If one baseband packet part of an L2CAP packet is flushed, then the rest of the HCI data packets for the L2CAP packet must also be flushed.

Event Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle which was flushed. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

7.7.18 Role Change Event

Event	Event Code	Event Parameters
Role Change	0x12	Status, BD_ADDR, New_Role

Description:

The Role Change event is used to indicate that the current Bluetooth role related to the particular connection has changed. This event only occurs when both the remote and local Bluetooth devices have completed their role change for the Bluetooth device associated with the BD_ADDR event parameter. This event allows both affected Hosts to be notified when the Role has been changed.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Role change has occurred.
0x01-0xFF	Role change failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device for which a role change has completed.

New_Role: *Size: 1 Octet*

Value	Parameter Description
0x00	Currently the Master for specified BD_ADDR.
0x01	Currently the Slave for specified BD_ADDR.

7.7.19 Number Of Completed Packets Event

Event	Event Code	Event Parameters
Number Of Completed Packets	0x13	Number_of_Handles, Connection_Handle[i], HC_Num_Of_Completed_Packets[i]

Description:

The Number Of Completed Packets event is used by the Controller to indicate to the Host how many HCI Data Packets have been completed (transmitted or flushed) for each Connection Handle since the previous Number Of Completed Packets event was sent to the Host. This means that the corresponding buffer space has been freed in the Controller. Based on this information, and the HC_Total_Num_ACL_Data_Packets and HC_Total_Num_Synchronous_Data_Packets return parameter of the Read_Buffer_Size command, the Host can determine for which Connection Handles the following HCI Data Packets should be sent to the Controller.The Number Of Completed Packets event must not be sent before the corresponding Connection Complete event. While the Controller has HCI data packets in its buffer, it must keep sending the Number Of Completed Packets event to the Host at least periodically, until it finally reports that all the pending ACL Data Packets have been transmitted or flushed. The rate with which this event is sent is manufacturer specific.
Note that Number Of Completed Packets events will not report on synchronous connection handles if synchronous Flow Control is disabled. (See Read/Write_Synchronous_Flow_Control_Enable on page 513 and page 514.)

Event Parameters:

Number_of_Handles: *Size: 1 Octet*

Value	Parameter Description
0xXX	The number of Connection Handles and Num_HCI_Data_Packets parameters pairs contained in this event. Range: 0-255

*Connection_Handle[i]: Size: Number_of_Handles * 2 Octets(12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

*HC_Num_Of_Completed_Packets [i]: Size: Number_of_Handles * 2 Octets*

Value	Parameter Description
N = 0xXXXX	The number of HCI Data Packets that have been completed (transmitted or flushed) for the associated Connection Handle since the previous time the event was returned. Range for N: 0x0000-0xFFFF

7.7.20 Mode Change Event

Event	Event Code	Event Parameters
Mode Change	0x14	Status, Connection_Handle, Current_Mode, Interval

Description:

The Mode Change event is used to indicate when the device associated with the Connection Handle changes between Active mode, Hold mode, Sniff mode, and Park state. The Connection_Handle will be a Connection_Handle for an ACL connection. The Connection_Handle event parameter is used to indicate which connection the Mode Change event is for. The Current_Mode event parameter is used to indicate which state the connection is currently in. The Interval parameter is used to specify a time amount specific to each state. Each Controller that is associated with the Connection Handle which has changed Modes will send the Mode Change event to its Host.

Event Parameters:

Status: Size: 1 Octet

Value	Parameter Description
0x00	A Mode Change has occurred.
0x01-0xFF	Hold_Mode, Sniff_Mode, Exit_Sniff_Mode, Park_State, or Exit_Park_State command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: Size: 2 Octets(12 Bits meaningful)

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Current_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	Active Mode.
0x01	Hold Mode.
0x02	Sniff Mode.
0x03	Park State.
0x04-0xFF	Reserved for future use.

Interval: *Size: 2 Octets*

Value	Parameter Description
N = 0xXXXX	Hold: Number of Baseband slots to wait in Hold Mode. Hold Interval = N * 0.625 msec (1 Baseband slot) Range for N: 0x0002-0xFFFE Time Range: 1.25 msec-40.9 sec Sniff: Number of Baseband slots between sniff intervals. Time between sniff intervals = 0.625 msec (1 Baseband slot) Range for N: 0x0002-0xFFFE Time Range: 1.25 msec-40.9 sec Park: Number of Baseband slots between consecutive beacons. Interval Length = N * 0.625 msec (1 Baseband slot) Range for N: 0x0002-0xFFFE Time Range: 1.25 msec-40.9 Seconds

7.7.21 Return Link Keys Event

Event	Event Code	Event Parameters
Return Link Keys	0x15	Num_Keys, BD_ADDR [i], Link_Key[i]

Description:

The Return Link Keys event is used by the Controller to send the Host one or more stored Link Keys. Zero or more instances of this event will occur after the Read_Stored_Link_Key command. When there are no link keys stored, no Return Link Keys events will be returned. When there are link keys stored, the number of link keys returned in each Return Link Keys event is implementation specific.

Event Parameters:

Num_Keys: *Size: 1 Octet*

Value	Parameter Description
0xXX	Number of Link Keys contained in this event. Range: 0x01 – 0x0B

BD_ADDR [i]: *Size: 6 Octets * Num_Keys*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for the associated Link Key.

Link_Key[i]: *Size:16 Octets * Num_Keys*

Value	Parameter Description
0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX	Link Key for the associated BD_ADDR.

7.7.22 PIN Code Request Event

Event	Event Code	Event Parameters
PIN Code Request	0x16	BD_ADDR

Description:

The PIN Code Request event is used to indicate that a PIN code is required to create a new link key. The Host must respond using either the PIN Code Request Reply or the PIN Code Request Negative Reply command, depending on whether the Host can provide the Controller with a PIN code or not.
Note: If the PIN Code Request event is masked away, then the Controller will assume that the Host has no PIN Code.

When the Controller generates a PIN Code Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a PIN_Code_Request_Reply or PIN_Code_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See “Link Manager Protocol” on page 211 [Part C].)

Event Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which a new link key is being created for.

7.7.23 Link Key Request Event

Event	Event Code	Event Parameters
Link Key Request	0x17	BD_ADDR

Description:

The Link Key Request event is used to indicate that a Link Key is required for the connection with the device specified in BD_ADDR. If the Host has the requested stored Link Key, then the Host will pass the requested Key to the Controller using the Link_Key_Request_Reply Command. If the Host does not have the requested stored Link Key, then the Host will use the Link_Key_Request_Negative_Reply Command to indicate to the Controller that the Host does not have the requested key.
Note: If the Link Key Request event is masked away, then the Controller will assume that the Host has no additional link keys.

When the Controller generates a Link Key Request event in order for the local Link Manager to respond to the request from the remote Link Manager (as a result of a Create_Connection or Authentication_Requested command from the remote Host), the local Host must respond with either a Link_Key_Request_Reply or Link_Key_Request_Negative_Reply command before the remote Link Manager detects LMP response timeout. (See “Link Manager Protocol” on page 211 [Part C].)

Event Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device which a stored link key is being requested.

7.7.24 Link Key Notification Event

Event	Event Code	Event Parameters
Link Key Notification	0x18	BD_ADDR, Link_Key, Key_Type

Description:

The Link Key Notification event is used to indicate to the Host that a new Link Key has been created for the connection with the device specified in BD_ADDR. The Host can save this new Link Key in its own storage for future use. Also, the Host can decided to store the Link Key in the Controller's Link Key Storage by using the Write_Stored_Link_Key command. The Key_Type event parameter informs the Host about which key type (combination key, local unit key or remote unit key) that has been used during pairing. If pairing with unit key is not supported, the Host can for instance discard the key or disconnect the link.

Event Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the Device for which the new link key has been generated.

Link_Key: *Size: 16 Octets*

Value	Parameter Description
0xXXXXXXXXXX XXXXXXXXXX XXXXXXXXXX	Link Key for the associated BD_ADDR.

Key_Type: *Size: 1 Octets*

Value	Parameter Description
0x00	Combination Key
0x01	Local Unit Key
0x02	Remote Unit Key
0x03-0xFF	Reserved

7.7.25 Loopback Command Event

Event	Event Code	Event Parameters
Loopback Command	0x19	HCI_Command_Packet

Description:

When in Local Loopback mode, the Controller loops back commands and data to the Host. The Loopback Command event is used to loop back all commands that the Host sends to the Controller with some exceptions. See Section 7.6.1, “Read Loopback Mode Command,” on page 558 for a description of which commands that are not looped back. The HCI_Command_Packet event parameter contains the entire HCI Command Packet including the header. Note: the event packet is limited to a maximum of 255 octets in the payload; since an HCI Command Packet has 3 octets of header data, only the first 252 octets of the command parameters will be returned.

Event Parameters:

HCI_Command_Packet: *Size: Depends on Command*

Value	Parameter Description
0xXXXXXX	HCI Command Packet, including header.

7.7.26 Data Buffer Overflow Event

Event	Event Code	Event Parameters
Data Buffer Overflow	0x1A	Link_Type

Description:

This event is used to indicate that the Controller’s data buffers have been over-flowed. This can occur if the Host has sent more packets than allowed. The Link_Type parameter is used to indicate that the overflow was caused by ACL or synchronous data.

Event Parameters:

Link_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	Synchronous Buffer Overflow (Voice Channels).
0x01	ACL Buffer Overflow (Data Channels).
0x02-0xFF	Reserved for Future Use.

7.7.27 Max Slots Change Event

Event	Event Code	Event Parameters
Max Slots Change	0x1B	Connection_Handle, LMP_Max_Slots

Description:

This event is used to notify the Host about the LMP_Max_Slots parameter when the value of this parameter changes. It will be sent each time the maximum allowed length, in number of slots, for baseband packets transmitted by the local device, changes. The Connection_Handle will be a Connection_Handle for an ACL connection.

Event Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

LMP_Max_Slots: *Size: 1 octet*

Value	Parameter Description
0x01, 0x03, 0x05	Maximum number of slots allowed to use for baseband packets, see Section 4.1.10 on page 247 and Section 5.2 on page 303 in "Link Manager Protocol" on page 211 [Part C].

7.7.28 Read Clock Offset Complete Event

Event	Event Code	Event Parameters
Read Clock Offset Complete	0x1C	Status, Connection_Handle, Clock_Offset

Description:

The Read Clock Offset Complete event is used to indicate the completion of the process of the Link Manager obtaining the Clock Offset information of the Bluetooth device specified by the Connection_Handle event parameter. The Connection_Handle will be a Connection_Handle for an ACL connection.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Clock_Offset command succeeded.
0x01-0xFF	Read_Clock_Offset command failed. See "Error Codes" on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 bits meaningful)*

Value	Parameter Description
0xXXXX	Specifies which Connection Handle's Clock Offset parameter is returned. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Clock_Offset: *Size: 2 Octets*

Bit format	Parameter Description
Bit 14-0	Bit 16-2 of CLKslave-CLKmaster.
Bit 15	Reserved.

7.7.29 Connection Packet Type Changed Event

Event	Event Code	Event Parameters
Connection Packet Type Changed	0x1D	Status, Connection_Handle, Packet_Type

Description:

The Connection Packet Type Changed event is used to indicate that the process has completed of the Link Manager changing which packet types can be used for the connection. This allows current connections to be dynamically modified to support different types of user data. The Packet_Type event parameter specifies which packet types the Link Manager can use for the connection identified by the Connection_Handle event parameter for sending L2CAP data or voice. The Packet_Type event parameter does not decide which packet types the LM is allowed to use for sending LMP PDUs.

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Connection Packet Type changed successfully.
0x01-0xFF	Connection Packet Type Changed failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Packet_Type: *Size: 2 Octets*

For ACL_Link_Type

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	2-DH1 may not be used.
0x0004	3-DH1 may not be used.
0x0008[1]	DM1 may be used.
0x0010	DH1 may be used.
0x0020	Reserved for future use.
0x0040	Reserved for future use.
0x0080	Reserved for future use.
0x0100	2-DH3 may not be used.
0x0200	3-DH3 may not be used.
0x0400	DM3 may be used.
0x0800	DH3 may be used.
0x1000	2-DH5 may not be used.
0x2000	3-DH5 may not be used.
0x4000	DM5 may be used.
0x8000	DH5 may be used.

1. This bit will be interpreted as set to 1 by Bluetooth V1.2 or later controllers.

For SCO_Link_Type

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	Reserved for future use.
0x0004	Reserved for future use.
0x0008	Reserved for future use.
0x0010	Reserved for future use.
0x0020	HV1
0x0040	HV2
0x0080	HV3
0x0100	Reserved for future use.
0x0200	Reserved for future use.

0x0400	Reserved for future use.
0x0800	Reserved for future use.
0x1000	Reserved for future use.
0x2000	Reserved for future use.
0x4000	Reserved for future use.
0x8000	Reserved for future use.

7.7.30 QoS Violation Event

Event	Event Code	Event Parameters
QoS Violation	0x1E	Connection_Handle

Description:

The QoS Violation event is used to indicate the Link Manager is unable to provide the current QoS requirement for the Connection Handle. This event indicates that the Link Manager is unable to provide one or more of the agreed QoS parameters. The Host chooses what action should be done. The Host can reissue QoS_Setup command to renegotiate the QoS setting for Connection Handle. The Connection_Handle will be a Connection_Handle for an ACL connection.

Event Parameters:

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle that the LM is unable to provide the current QoS requested for. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

7.7.31 Page Scan Repetition Mode Change Event

Event	Event Code	Event Parameters
Page Scan Repetition Mode Change	0x20	BD_ADDR, Page_Scan_Repetition_Mode

Description:

The Page Scan Repetition Mode Change event indicates that the remote Bluetooth device with the specified BD_ADDR has successfully changed the Page_Scan_Repetition_Mode (SR).

Event Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the remote device.

Page_Scan_Repetition_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03 – 0xFF	Reserved.

7.7.32 Flow Specification Complete Event

Event	Event Code	Event Parameters
HCI Flow Specification Complete	0x21	Status, Connection_Handle, Flags, Flow_direction, Service_Type, Token_Rate, Token_Bucket_Size, Peak_Bandwidth, Access Latency

Description:

The Flow Specification Complete event is used to inform the Host about the Quality of Service for the ACL connection the Controller is able to support. The Connection_Handle will be a Connection_Handle for an ACL connection. The flow parameters refer to the outgoing or incoming traffic of the ACL link, as indicated by the Flow_direction field. The flow parameters are defined in the L2CAP specification “Quality of Service (QoS) Option” on page 60[vol. 4]. When the Status parameter indicates a successful completion, the flow parameters specify the agreed values by the Controller. When the Status parameter indicates a failed completion with the Error Code *QoS Unacceptable Parameters* (0x2C), the flow parameters specify the acceptable values of the Controller. This enables the Host to continue the 'QoS negotiation' with a new HCI Flow_Specification command with flow parameter values that are acceptable for the Controller. When the Status parameter indicates a failed completion with the Error Code *QoS Rejected* (0x2D), this indicates a request of the Controller to discontinue the 'QoS negotiation'. When the Status parameter indicates a failed completion, the flow parameter values of the most recently successful completion must be assumed (or the default values when there was no success completion).

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Flow Specification command succeeded
0x01 – 0xFF	Flow Specification command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle used to identify for which ACL connection the Flow is specified. Range: 0x0000 - 0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

Flags: *Size: 1 Octet)*

Value	Parameter Description
0x00 – 0xFF	Reserved for Future Use.

Flow_direction: *Size: 1 Octet*

Value	Parameter Description
0x00	Outgoing Flow i.e. traffic send over the ACL connection
0x01	Incoming Flow i.e. traffic received over the ACL connection
0x02 – 0xFF	Reserved for Future Use.

Service_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	No Traffic
0x01	Best Effort
0x02	Guaranteed
0x03 – 0xFF	Reserved for Future Use

Token Rate: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Token Rate in octets per second

Token Bucket Size: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Token Bucket Size in octets

Peak_Bandwidth: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Peak Bandwidth in octets per second

Access Latency: *Size: 4 Octets*

Value	Parameter Description
0xXXXXXXXX	Access Latency in microseconds

7.7.33 Inquiry Result with RSSI Event

Event	Event Code	Event Parameters
Inquiry Result with RSSI	0x22	Num_responses, BD_ADDR[i], Page_Scan_Repetition_Mode[i], Reserved[i], Class_of_Device[i], Clock_Offset[i], RSSI[i]

Description:

The Inquiry Result with RSSI event indicates that a Bluetooth device or multiple Bluetooth devices have responded so far during the current Inquiry process. This event will be sent from the Controller to the Host as soon as an Inquiry Response from a remote device is received if the remote device supports only mandatory paging scheme. This Controller may queue these Inquiry Responses and send multiple Bluetooth devices information in one Inquiry Result event. The event can be used to return one or more Inquiry responses in one event. The RSSI parameter is measured during the FHS packet returned by each responding slave.

This event shall only be generated if the Inquiry Mode parameter of the last Write_Inquiry_Mode command was set to 0x01 (Inquiry Result format with RSSI).

Event Parameters:

Num_Responses: *Size: 1 Octet*

Value	Parameter Description
0xXX	Number of responses from the Inquiry.

BD_ADDR[i]: *Size: 6 Octets * Num_Responses*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR for each device which responded.

Page_Scan_Repetition_Mode[i]: *Size: 1 Octet* Num_Responses*

Value	Parameter Description
0x00	R0
0x01	R1
0x02	R2
0x03 – 0xFF	Reserved

Reserved[i]:[1] *Size: 1 Octet* Num_Responses*

Value	Parameter Description
0xXX	Reserved.

Class_of_Device[i]: *Size: 3 Octets * Num_Responses*

Value	Parameter Description
0xXXXXXX	Class of Device for the device

Clock_Offset[i]: *Size: 2 Octets * Num_Responses*

Bit format	Parameter Description
Bit 14-0	Bit 16-2 of CLKslave-CLKmaster.
Bit 15	Reserved

RSSI[i]: *Size: 1 Octet * Num_Responses*

Value	Parameter Description
0xXX	Range: -127 to +20 Units: dBm

1. This was the Page_Scan_Period_Mode parameter in the v1.1 specification. This parameter has no meaning in v1.2 and no default value.

7.7.34 Read Remote Extended Features Complete Event

Event	Event Code	Event Parameters
Read Remote Extended Features Complete	0x23	Status, Connection_Handle, Page_Number, Maximum page number, Extended_LMP_Features

Description:

The Read Remote Extended Features Complete event is used to indicate the completion of the process of the Link Manager obtaining the remote extended LMP features of the remote device specified by the connection handle event parameter. The connection handle will be a connection handle for an ACL connection. The event parameters include a page of the remote devices extended LMP features. For details see "Link Manager Protocol" on page 211 [Part C].

Event Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Request for remote extended features succeeded
0x01-0xFF	Request for remote extended features failed – standard HCI error value

Connection_Handle: *Size: 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	The connection handle identifying the device to which the remote features apply. Range: 0x0000-0x0EFF (0x0F00-0x0FFF Reserved for future use)

Page Number: *Size: 1Octet*

Value	Parameter Description
0x00	The normal LMP features as returned by HCI_Read_Remote_Supported_Features
0x01-0xFF	The page number of the features returned

Maximum Page Number: *Size: 1Octet*

Value	Parameter Description
0x00-0xFF	The highest features page number which contains non-zero bits for the local device

Extended_LMP_Features: *Size: 8 Octets*

Value	Parameter Description
0xFFFFFFFFFFFFFFFF	Bit map of requested page of LMP features. See LMP specification for details

7.7.35 Synchronous Connection Complete Event

Event	Event Code	Event Parameters
Synchronous Connection Complete	0x2C	Status, Connection_Handle, BD_ADDR, Link Type, Transmission_Interval, Retransmission Window, Rx_Packet_Length, Tx_Packet_Length Air Mode

Description:

The Synchronous Connection Complete event indicates to both the Hosts that a new Synchronous connection has been established. This event also indicates to the Host, which issued the Setup_Synchronous_Connection, or Accept_Synchronous_Connection_Request or Reject_Synchronous_Connection_Request command and then received a Command Status event, if the issued command failed or was successful.

Event Parameters:

Status: *1 octet*

Value	Parameter Description
0x00	Connection successfully completed.
0x01 –0xFF	Connection failed to complete.See “Error Codes” on page 319 [Part D] for error codes and description.

Connection_Handle: *2 octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection between two Bluetooth devices. Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

BD_ADDR: *6 octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the other connected device forming the connection.

Link_Type: *Size: 1 Octet*

Value	Parameter Description
0x00	SCO Connection
0x01	Reserved
0x02	eSCO Connection
0x03 – 0xFF	Reserved

Transmission_Interval: *1 octets*

Value	Parameter Description
0xXX	Time between two consecutive eSCO instants measured in slots. Must be zero for SCO links.

Retransmission window: *1 octets*

Value	Parameter Description
0xXX	The size of the retransmission window measured in slots. Must be zero for SCO links.

Rx_Packet_Length: *2 octets*

Value	Parameter Description
0xXXXX	Length in bytes of the eSCO payload in the receive direction. Must be zero for SCO links.

Tx_Packet_Length: *2 octets*

Value	Parameter Description
0xXXXX	Length in bytes of the eSCO payload in the transmit direction. Must be zero for SCO links.

Air Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	μ-law log
0x01	A-law log
0x02	CVSD
0x03	Transparent Data
0x04 – 0xFF	Reserved

7.7.36 Synchronous Connection Changed event

Event	Event Code	Event Parameters
Synchronous Connection Changed	0x2D	Status, Connection_Handle, Transmission_Interval, Retransmission Window, Rx_Packet_Length, Tx_Packet_Length

Description:

The Synchronous Connection Changed event indicates to the Host that an existing Synchronous connection has been reconfigured. This event also indicates to the initiating Host (if the change was host initiated) if the issued command failed or was successful.

Command Parameters:

Status: *1 octet*

Value	Parameter Description
0x00	Connection successfully reconfigured.
0x01 –0xFF	Reconfiguration failed to complete. See “Error Codes” on page 319 [Part D] for error codes and description.

Connection_Handle: *2 octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle to be used to identify a connection between two Bluetooth devices. Range: 0x0000-0x0EFF (0x0F00 – 0x0FFF Reserved for future use)

Transmission_Interval: *1 octet*

Value	Parameter Description
0xXX	Time between two consecutive SCO/eSCO instants measured in slots.

Retransmission window: *1 octet*

Value	Parameter Description
0xXX	The size of the retransmission window measured in slots. Must be zero for SCO links.

Rx_Packet_Length: *2 octets*

Value	Parameter Description
0xXXXX	Length in bytes of the SCO/eSCO payload in the receive direction.

Tx_Packet_Length: *2 octets*

Value	Parameter Description
0xXXXX	Length in bytes of the SCO/eSCO payload in the transmit direction.

8 LIST OF FIGURES

9 LIST OF TABLES

APPENDIX A: DEPRECATED COMMANDS, EVENTS AND CONFIGURATION PARAMETERS

Commands, events and configuration parameters in this section were in prior versions of the specification, but have been determined not to be required.

They may be implemented by a controller to allow for backwards compatibility with a host utilizing a prior version of the specification.

A host should not use these commands.

Contents:

Page Scan Mode

The Page_Scan_Mode parameter indicates the page scan mode that is used for default page scan. Currently one mandatory page scan mode and three optional page scan modes are defined. Following an inquiry response, if the Baseband timer T_mandatory_pscan has not expired, the mandatory page scan mode must be applied. For details see the “Baseband Specification” on page 55 [Part B] (Keyword: Page-Scan-Mode, FHS-Packet, T_mandatory_pscan)

Value	Parameter Description
0x00	Mandatory Page Scan Mode
0x01	Optional Page Scan Mode I
0x02	Optional Page Scan Mode II
0x03	Optional Page Scan Mode III
0x04-0xFF	Reserved

Read Page Scan Mode Command

Command	OGF	OCF	Command Parameters	Return Parameters
HCI_Read_Page_Scan_Mode	0x03	0x003D		Status, Page_Scan_Mode

Description:

This command is used to read the default Page Scan Mode configuration parameter of the local Bluetooth device. See “Page Scan Mode” on page 612..

Command Parameters:

None

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Page_Scan_Mode command succeeded.
0x01-0xFF	Read_Page_Scan_Mode command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Page_Scan_Mode: *Size: 1 Octet*

Value	Parameter Description
See Appendix A, “Page Scan Mode” on page 612.	

Event(s) generated (unless masked away):

When the Read_Page_Scan_Mode command has completed, a Command Complete event will be generated.

Write Page Scan Mode Command

OGF: 0x03 (Controller and baseband commands)

Command	OGF	OCF	Command Parameters	Return Parameters
HCI_Write_Page_Scan_Mode	0x03	0x003E	Page_Scan_Mode	Status

Description:

This command is used to write the default Page Scan Mode configuration parameter of the local Bluetooth device. See “Page Scan Mode” on page 612.

Command Parameters:

Page_Scan_Mode: *Size: 1 Octet*

Value	Parameter Description
See Appendix A, “Page Scan Mode” on page 612.	

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Write_Page_Scan_Mode command succeeded.
0x01-0xFF	Write_Page_Scan_Mode command failed. See “Error Codes” on page 319 [Part D] for list of Error Codes.

Event(s) generated (unless masked away):

When the Write_Page_Scan_Mode command has completed, a Command Complete event will be generated.

Read Country Code Command

Command	OGF	OCF	Command Parameters	Return Parameters
HCI_Read_Country_Code	0x05	0x0007		Status, Country_Code

Description:

This command will read the value for the Country_Code return parameter. The Country_Code defines which range of frequency band of the ISM 2.4 GHz band will be used by the device. Each country has local regulatory bodies regulating which ISM 2.4 GHz frequency ranges can be used.

Command Parameters:

None.

Return Parameters:

Status: *Size: 1 Octet*

Value	Parameter Description
0x00	Read_Country_Code command succeeded.
0x01-0xFF	Read_Country_Code command failed. See "Error Codes" on page 319 [Part D] for error codes and description.

Country_Code: *Size: 1 Octet*

Value	Parameter Description
0x00	North America & Europe* and Japan
0x01	France
0x04-FF	Reserved for Future Use.

*. Except France

Event(s) generated (unless masked away):

When the Read_Country_Code command has completed, a Command Complete event will be generated.

Add SCO Connection Command

Command	OGF	OCF	Command Parameters	Return Parameters
HCI_Add_SCO_Connection	0x01	0x0007	Connection_Handle, Packet_Type	

Description:

This command will cause the Link Manager to create a SCO connection using the ACL connection specified by the Connection_Handle command parameter. This command causes the local Bluetooth device to create a SCO connection. The Link Manager will determine how the new connection is established. This connection is determined by the current state of the device, its piconet, and the state of the device to be connected. The Packet_Type command parameter specifies which packet types the Link Manager should use for the connection. The Link Manager must only use the packet type(s) specified by the Packet_Type command parameter for sending HCI SCO Data Packets. Multiple packet types may be specified for the Packet_Type command parameter by performing a bitwise OR operation of the different packet types. The Link Manager may choose which packet type is to be used from the list of acceptable packet types. A Connection Handle for this connection is returned in the Connection Complete event (see below).
Note: An SCO connection can only be created when an ACL connection already exists and when it is not put in park. For a definition of the different packet types, see the “Baseband Specification” on page 55 [Part B].

Note: At least one packet type must be specified. The Host should enable as many packet types as possible for the Link Manager to perform efficiently. However, the Host must not enable packet types that the local device does not support.

Command Parameters:

Connection_Handle *Size 2 Octets (12 Bits meaningful)*

Value	Parameter Description
0xXXXX	Connection Handle for the ACL connection being used to create an SCO connection. Range: 0x0000-0x0EFF (0x0F00 - 0x0FFF Reserved for future use)

Packet_Type: *Size: 2 Octets*

Value	Parameter Description
0x0001	Reserved for future use.
0x0002	Reserved for future use.
0x0004	Reserved for future use.
0x0008	Reserved for future use.
0x0010	Reserved for future use.
0x0020	HV1
0x0040	HV2
0x0080	HV3
0x0100	Reserved for future use.
0x0200	Reserved for future use.
0x0400	Reserved for future use.
0x0800	Reserved for future use.
0x1000	Reserved for future use.
0x2000	Reserved for future use.
0x4000	Reserved for future use.
0x8000	Reserved for future use.

Return Parameters:

None.

Event(s) generated (unless masked away):

When the Controller receives the Add_SCO_Connection command, it sends the Command Status event to the Host. In addition, when the LM determines the connection is established, the local Controller will send a Connection Complete event to its Host, and the remote Controller will send a Connection Complete event or a Synchronous Connection Complete event to the Host. The Connection Complete event contains the Connection Handle if this command is successful.
Note: no Command Complete event will be sent by the Controller to indicate that this command has been completed. Instead, the Connection Complete event will indicate that this command has been completed.

Page Scan Mode Change Event

Event	Event Code	Event Parameters
Page Scan Mode Change	0x1F	BD_ADDR, Page_Scan_Mode

Description:

The Page Scan Mode Change event indicates that the connected remote Bluetooth device with the specified BD_ADDR has successfully changed the Page_Scan_Mode.

Event Parameters:

BD_ADDR: *Size: 6 Octets*

Value	Parameter Description
0xXXXXXXXXXXXX	BD_ADDR of the remote device.

Page_Scan_Mode: *Size: 1 Octet*

Value	Parameter Description
0x00	Mandatory Page Scan Mode.
0x01	Optional Page Scan Mode I.
0x02	Optional Page Scan Mode II.
0x03	Optional Page Scan Mode III.
0x04 – 0xFF	Reserved.

MESSAGE SEQUENCE CHARTS

Between Host and Host Controller/Link Manager

Examples of interactions between Host Controller Interface Commands and Events and Link Manager Protocol Data Units are represented in the form of message sequence charts. These charts show typical interactions and do not indicate all possible protocol behavior.

CONTENTS

1 INTRODUCTION

This section shows typical interactions between Host Controller Interface (HCI) Commands and Events and Link Manager (LM) Protocol Data Units (PDU). It focuses on the message sequence charts (MSCs) for the procedures specified in [3] “Bluetooth Host Controller Interface Functional Specification” with regard to LM Procedures from [2] “Link Manager Protocol”.

This section illustrates only the most useful scenarios, it does not cover all possible alternatives. Furthermore, the message sequence charts do not consider errors over the air interface or host interface. In all message sequence charts it is assumed that all events are not masked, so the Host Controller will not filter out any events.

The sequence of messages in these message sequence charts is for illustrative purposes. The messages may be sent in a different order where allowed by the Link Manager or HCI sections. If any of these charts differ with text in the Baseband, Link Manager, or HCI sections, the text in those sections shall be considered normative. This section is informative.

1.1 NOTATION

The notation used in the message sequence charts (MSCs) consists of ovals, elongated hexagons, boxes, lines, and arrows. The vertical lines terminated on the top by a shadow box and at the bottom by solid oval indicate a protocol entity that resides in a device. MSCs describe interactions between these entities and states those entities may be in.

The following symbols represent interactions and states:

Oval	Defines the context for the message sequence chart.
Hexagon	Indicates a condition needed to start the transactions below this hexagon. The location and width of the Hexagon indicates which entity or entities make this decision.
Box	Replaces a group of transactions. May indicate a user action, or a procedure in the baseband.
Dashed Box	Optional group of transactions.
Solid Arrow	Represents a message, signal or transaction. Can be used to show LMP and HCI traffic. Some baseband packet traffic is also shown. These are prefixed by BB followed by either the type of packet, or an indication that there is an ACK signal in a packet.
Dashed Arrow	Represents a optional message, signal or transaction. Can be used to show LMP and HCI traffic.

1.2 FLOW OF CONTROL

Some message sequences are split into several charts. These charts are marked in sequence with different step numbers with multiple paths through with optional letters after the step numbers. Numbers indicate normal or required ordering. The letters represent alternative paths. For example, Step 4 is after Step 3, and Step 5a could be executed instead of Step 5b.

1.3 EXAMPLE MSC

The protocol entities represented in the example shown in Figure 1.1 on page 624 illustrate the interactions of two devices named A and B. Note that each device includes a Host and a LM entity in this example. Other MSCs in this section may show the interactions of more than two devices.

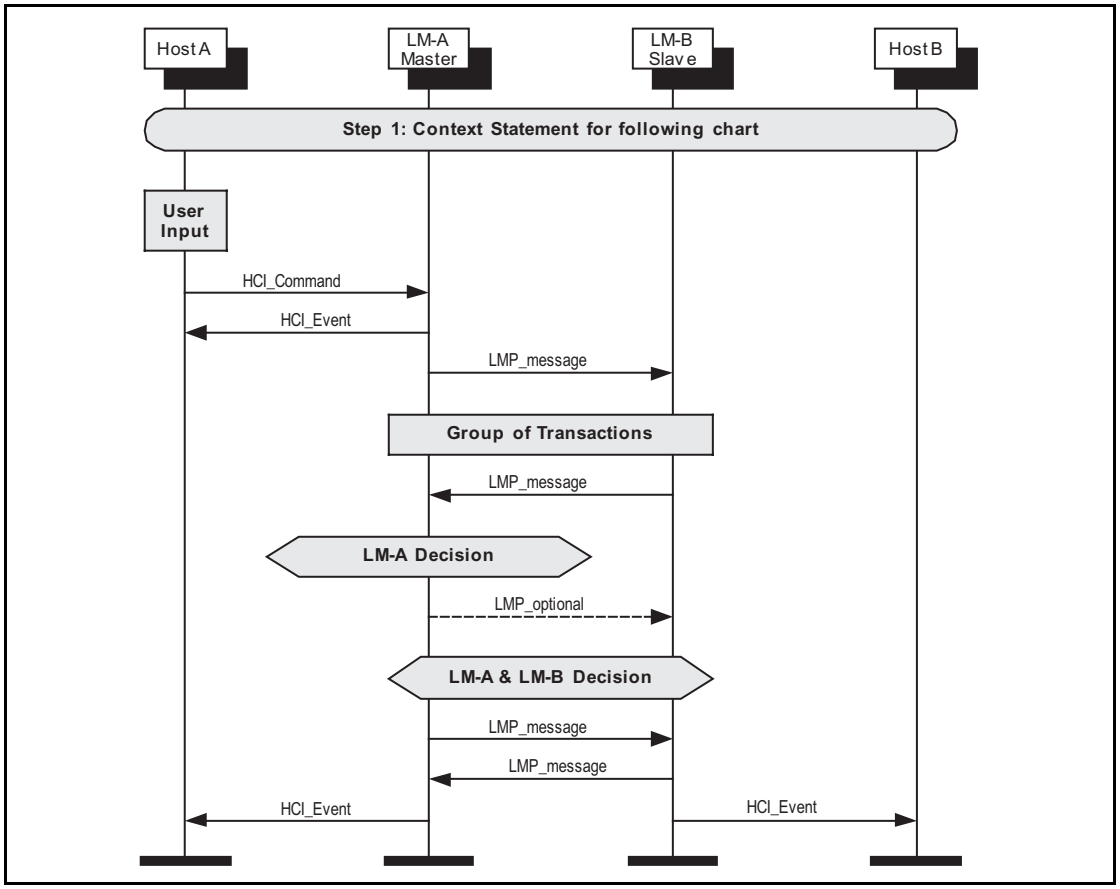


Figure 1.1: Example MSC.

2 SERVICES WITHOUT CONNECTION REQUEST

2.1 REMOTE NAME REQUEST

The service Remote Name Request is used to find out the name of the remote device without requiring an explicit ACL Connection.

Step 1: The host sends an HCI_Remote_Name_Request command expecting that its local device will automatically try to connect to the remote device. (See Figure 2.1 on page 625)

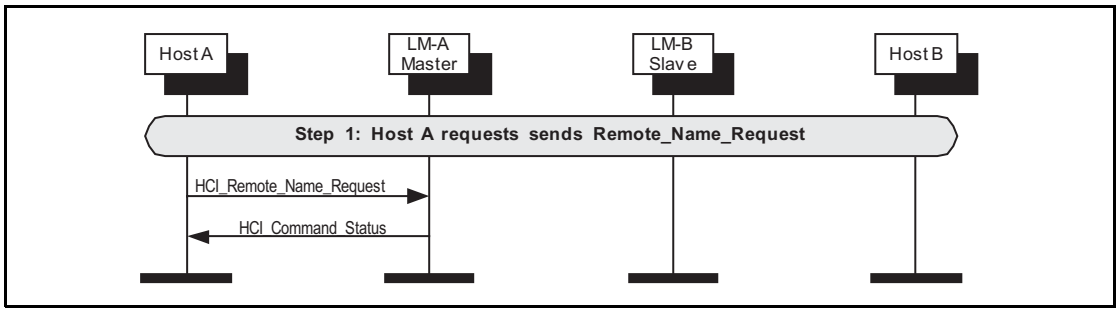


Figure 2.1: Remote name request.

Step 2a: If an ACL Connection does not exist device A pages device B. After the Baseband paging procedure, the local device attempts to get the name, disconnect, and return the name of the remote device to the Host. (See Figure 2.2 on page 625)

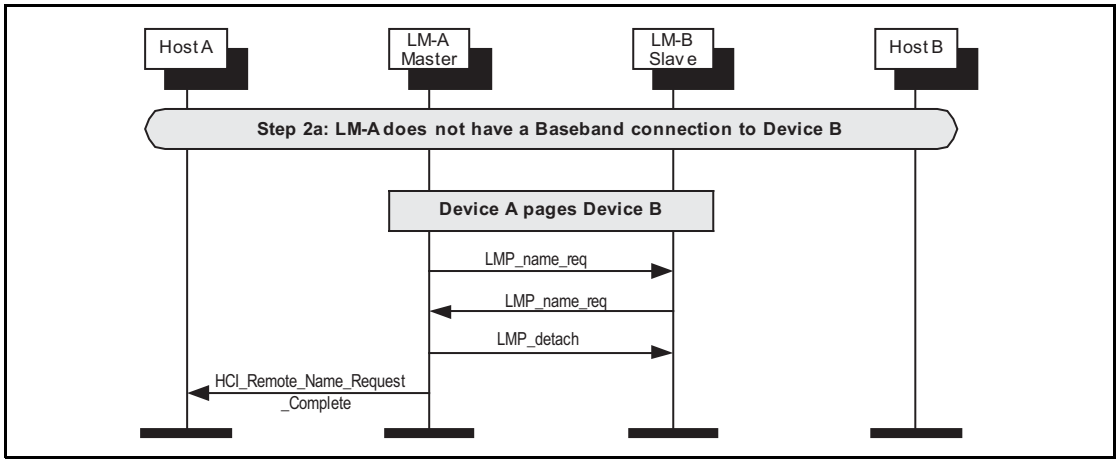


Figure 2.2: Remote name request if no current baseband connection.

Step 2b: If an ACL Connection exists when the request is made, then the Remote Name Request procedure will be executed like an optional service. No Paging and no ACL disconnect is done. (See Figure 2.3 on page 626)

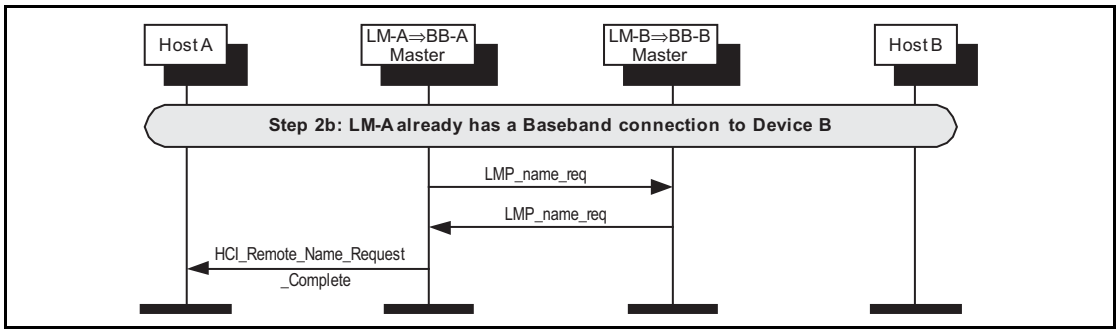


Figure 2.3: Remote name request with baseband connection.

2.2 ONE-TIME INQUIRY

Inquiry is used to detect and collect nearby devices.

Step 1: The host sends an HCI_Inquiry command. (See Figure 2.4 on page 626)

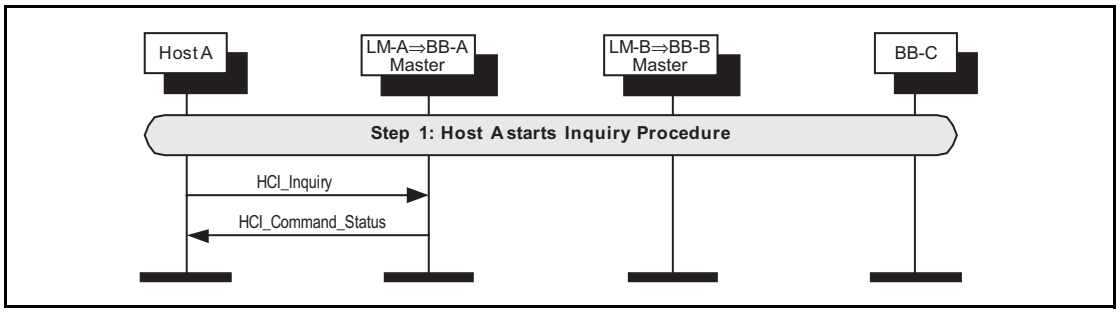


Figure 2.4: Host A starts inquiry procedure.

Step 2: The Controller will start the Baseband inquiry procedure with the specified Inquiry Access Code and Inquiry Length. When Inquiry Responses are received, the Controller extracts the required information and returns the information related to the found devices using one or more Inquiry Result events to the Host. (See Figure 2.5 on page 627)

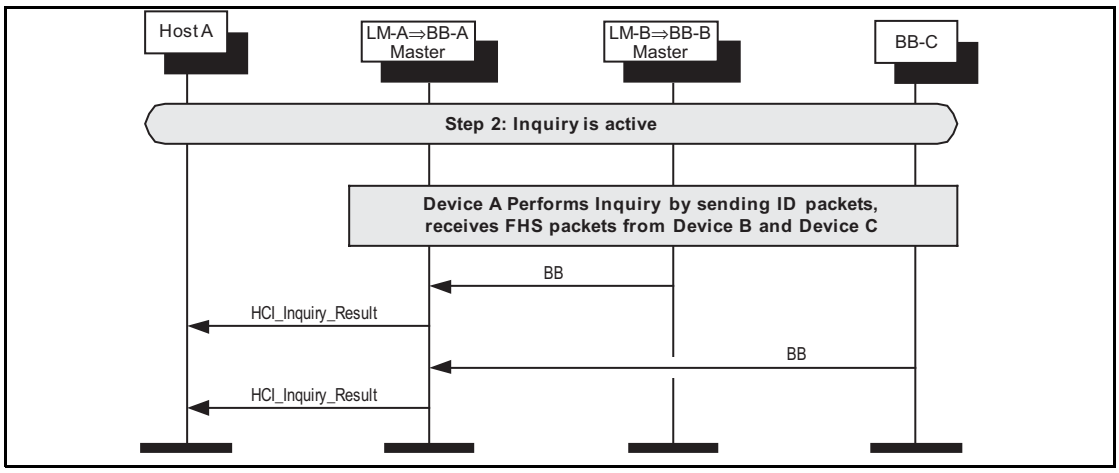


Figure 2.5: LM-A performs inquiry and reports result.

Step 3a: If the host wishes to terminate an Inquiry, the HCI_Inquiry_Cancel command is used to immediately stop the inquiry procedure.
(See Figure 2.6 on page 627)

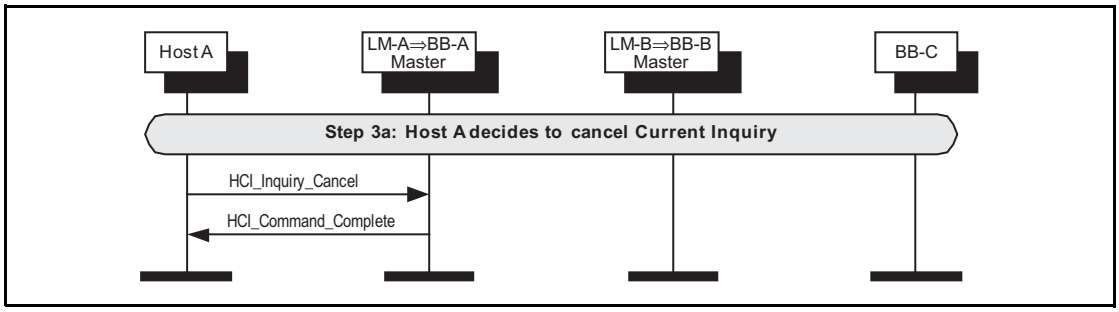


Figure 2.6: Host A cancels inquiry.

Step 3b: If the Inquiry procedure is completed due to the number of results obtained, or the Inquiry Length has expired, an Inquiry Complete event is returned to the Host. (See Figure 2.7 on page 628)

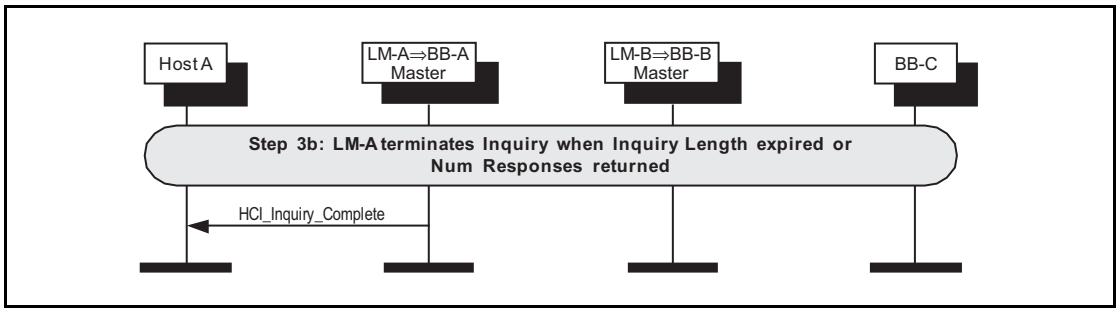


Figure 2.7: LM-A terminates current inquiry.

2.3 PERIODIC INQUIRY

Periodic inquiry is used when the inquiry procedure is to be repeated periodically.

Step 1: The hosts sends an HCI_Periodic_Inquiry_Mode command. (See Figure 2.8 on page 628)

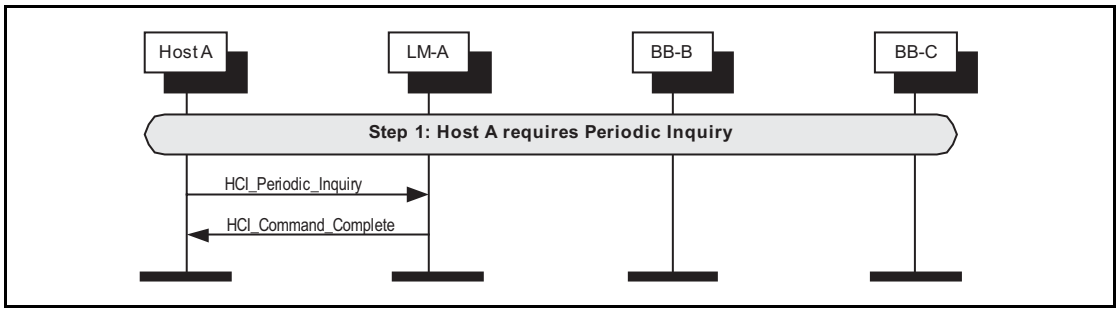


Figure 2.8: Host A starts periodic inquiry.

Step 2: The Controller will start a periodic Inquiry. In the inquiry cycle, one or several Inquiry Result events will be returned. (See Figure 2.9 on page 629)

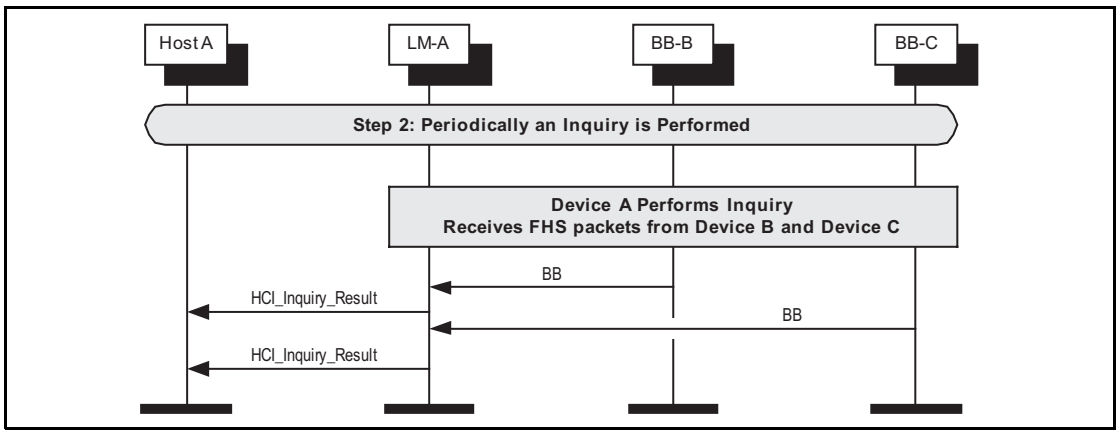


Figure 2.9: LM-A periodically performs an inquiry and reports result.

Step 3: An Inquiry Complete event will be returned to the Host when the current periodic inquiry has finished. (See Figure 2.10 on page 629)

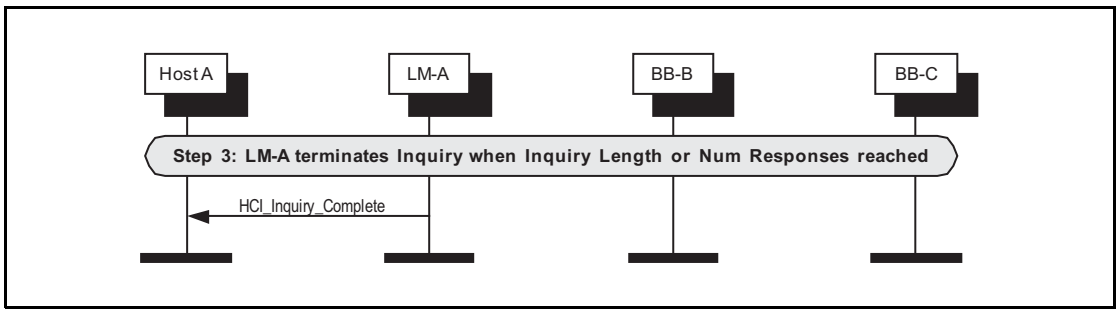


Figure 2.10: LM-A terminates current inquiry.

Step 4: The periodic Inquiry can be stopped using the HCI_Exit_Periodic_Inquiry_Mode command. (See Figure 2.11 on page 629)

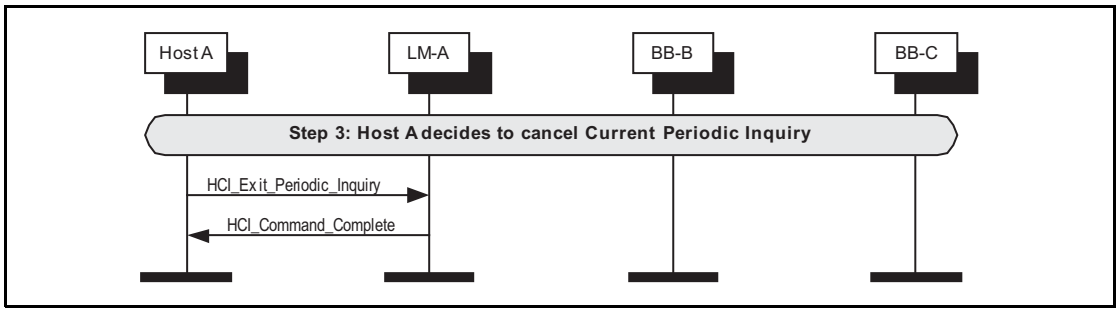


Figure 2.11: Host A decides to exit periodic inquiry.

3 ACL CONNECTION ESTABLISHMENT AND DETACHMENT

A flow diagram of the establishment and detachment of a connection between two devices is shown in Figure 3.1 on page 631. The process is illustrated in 9 distinct steps. A number of these steps may be optionally performed, such as authentication and encryption. Some steps are required, such as the Connection Request and Setup Complete steps. The steps in the overview diagram directly relate to the steps in the following message sequence charts.

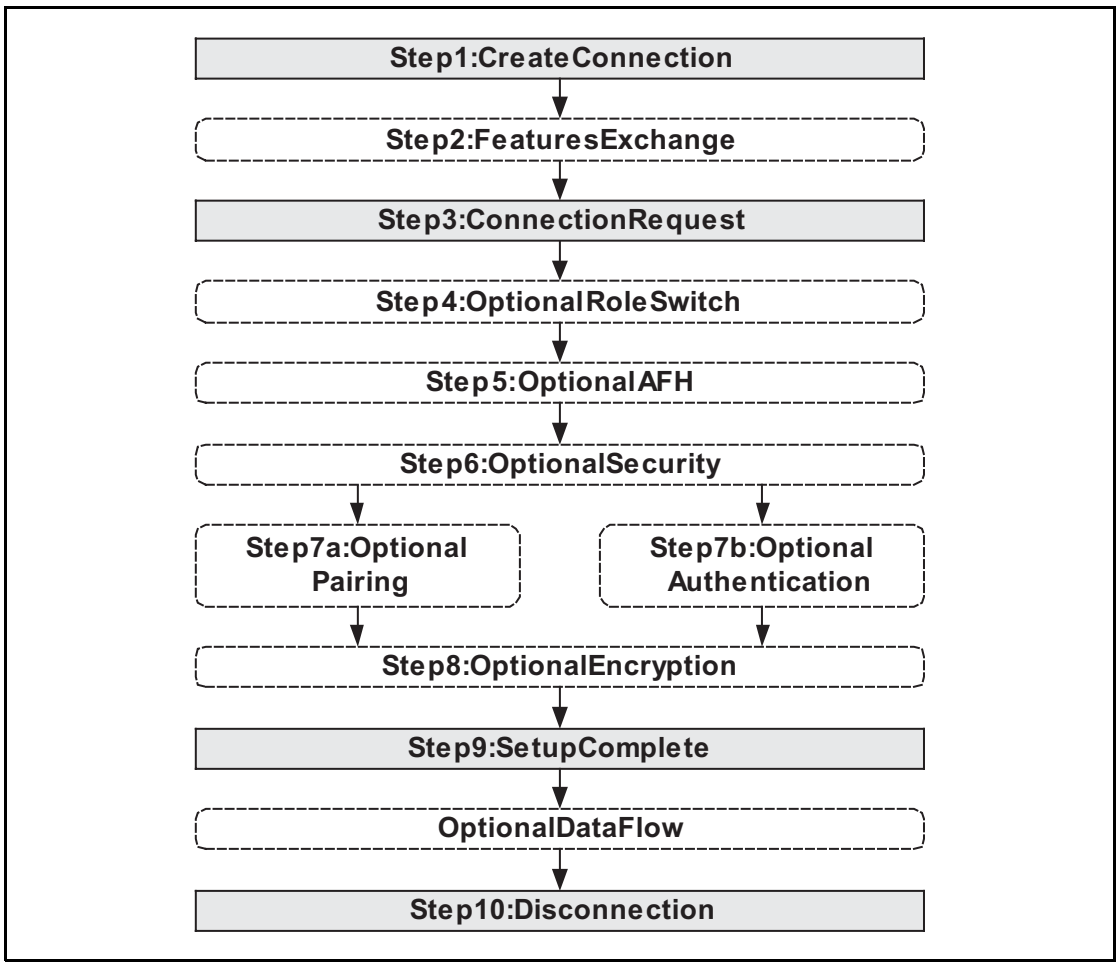


Figure 3.1: Overview diagram for connection setup.

3.1 CONNECTION SETUP

Step 1: The host sends an HCI_Create_Connection command to the Controller. The Controller then performs a Baseband paging procedure with the specified BD_ADDR. (See Figure 3.2 on page 632)

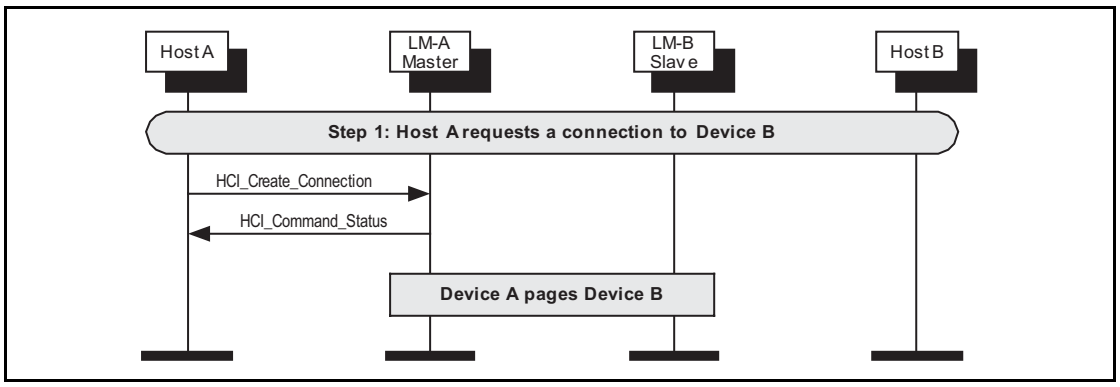


Figure 3.2: Host A requests connection with device B.

Step 2: Optionally, the LM may decide to exchange features. (See Figure 3.3 on page 632)

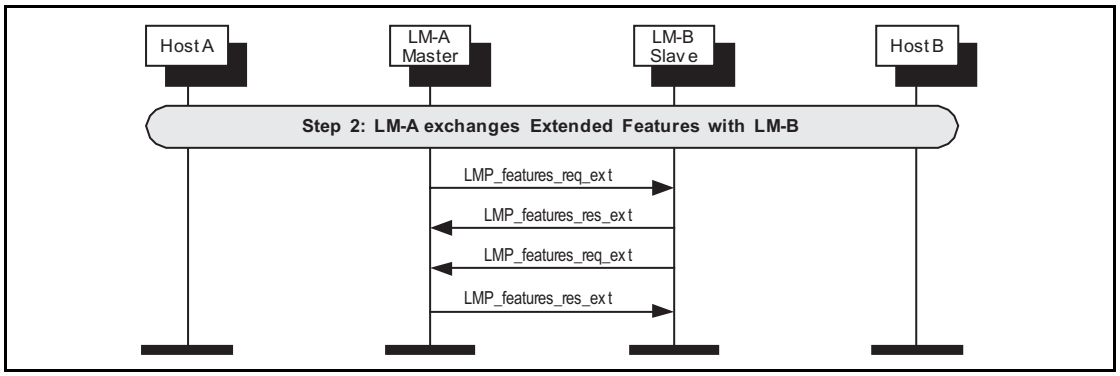


Figure 3.3: LM-A and LM-B exchange features.

Step 3: The LM on the master will request an LMP_host_connection_req PDU. The LM on the slave will then confirm that a connection is OK, and if so, what role is preferred. (See Figure 3.4 on page 632)

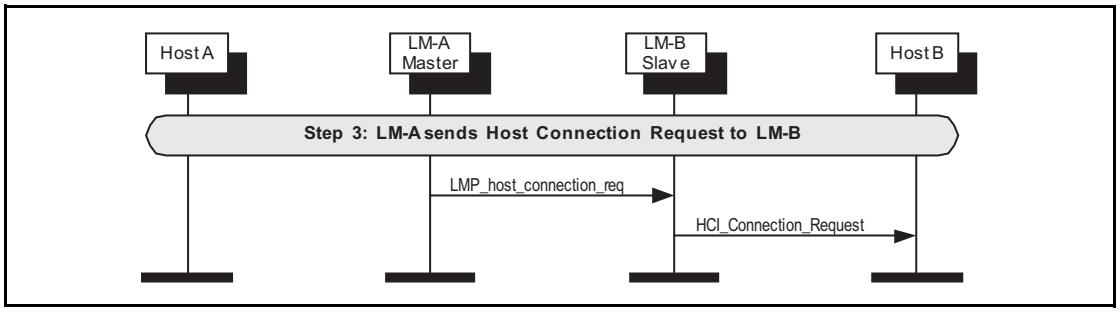


Figure 3.4: LM-A requests host connection.

Step 4a: The remote host rejects this connection, and the link is terminated. (See Figure 3.5 on page 633)

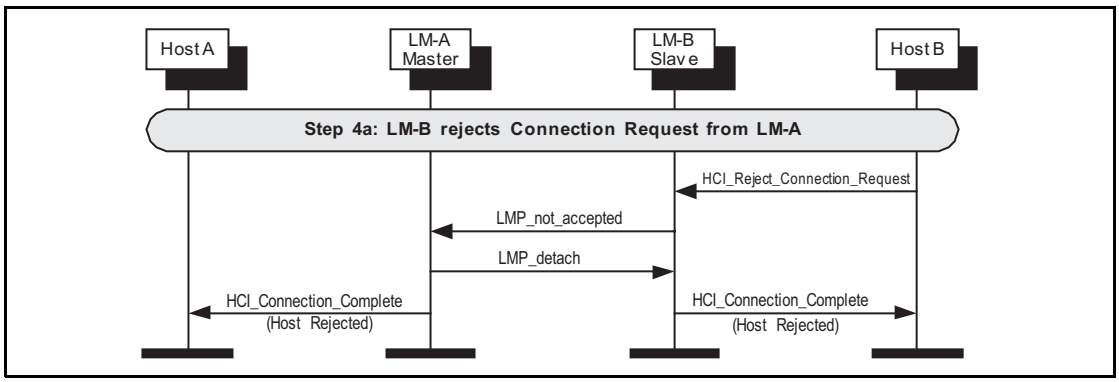


Figure 3.5: Device B rejects connection request.

Step 4b: The remote host accepts this connection. (See Figure 3.6 on page 633)

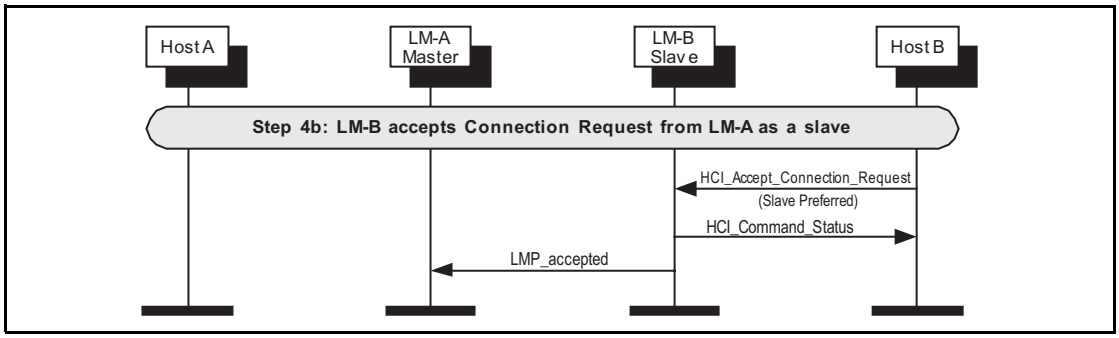


Figure 3.6: Device B accepts connection request.

Step 4c: The remote host accepts this connection but with the preference of being a master. This will cause a role switch to occur before the LMP_accepted for the LMP_host_connection_req PDU is sent. (See Figure 3.7 on page 634)

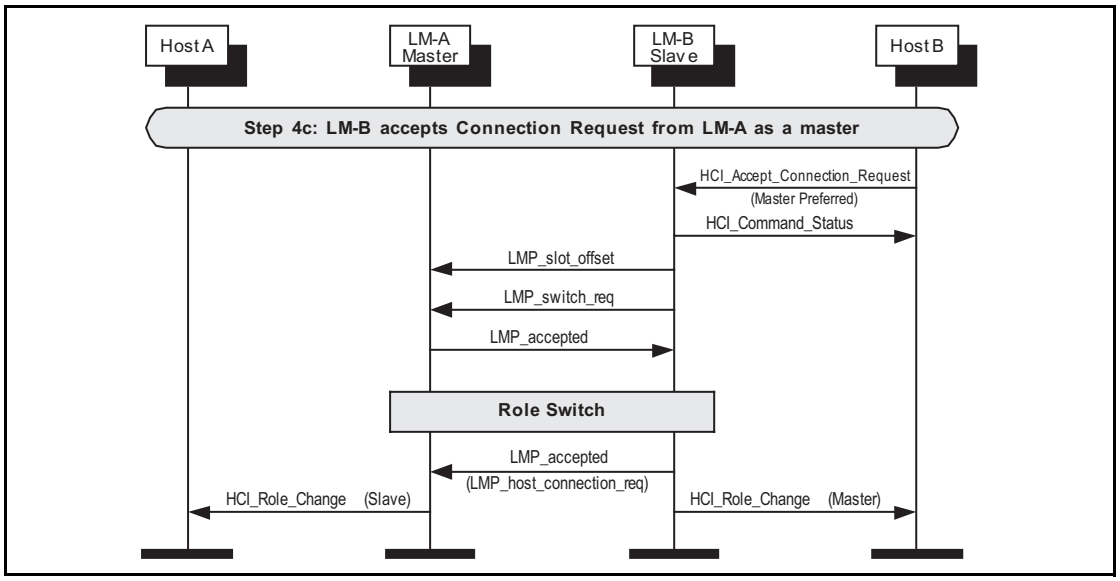


Figure 3.7: Device B accepts connection requests as master.

Step 5: After the features have been exchanged and AFH support is determined to be available, the master may at any time send an LMP_set_AFH and LMP_channel_classification_req PDU. (See Figure 3.8 on page 634)

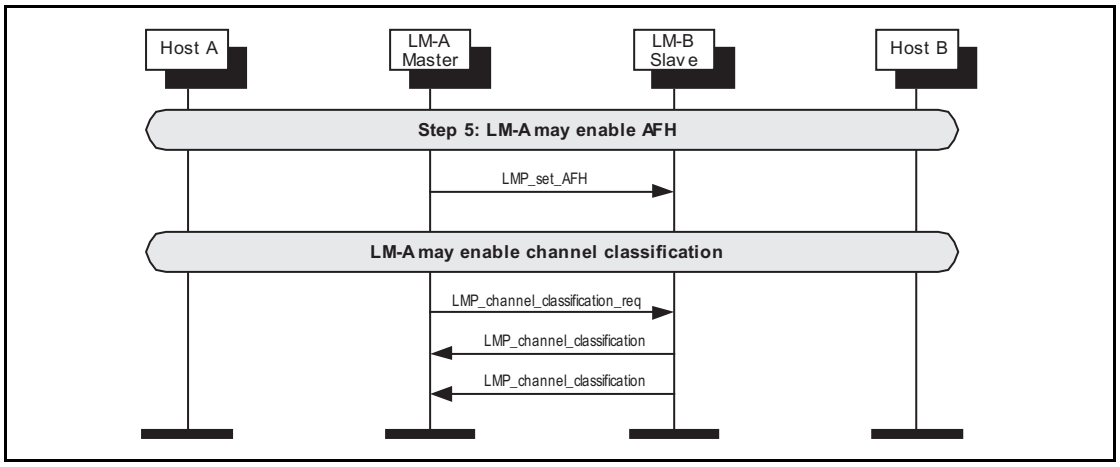


Figure 3.8: LM-A starts adaptive frequency hopping.

Step 6: The LM will request if authentication is required. It does this by requesting the Link Key for this connection from the Host. (See Figure 3.9 on page 635)

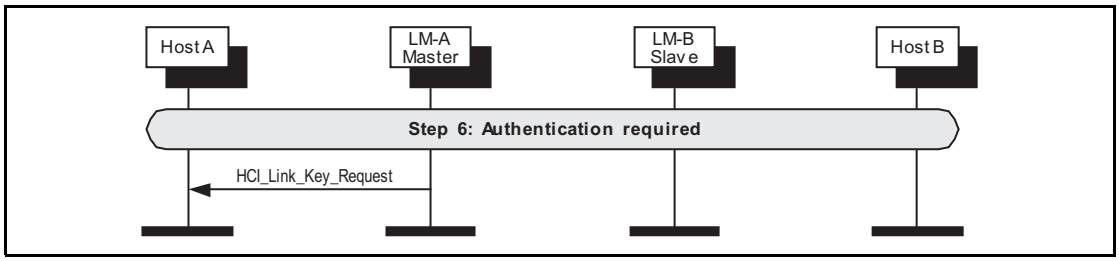


Figure 3.9: Authentication initiated.

Step 7a: If authentication is required by the higher layers and the devices to be connected do not have a common link key, a pairing procedure will be used. The LM will have requested a link key from the host for this connection. If there is a negative reply, then a PIN code will be requested. This PIN code will be requested on both sides of the connection, and authentication performed based on this PIN code. The last step is for the new link key for this connection to be passed to the host so that it may store it for future connections.
(See Figure 3.10 on page 636)

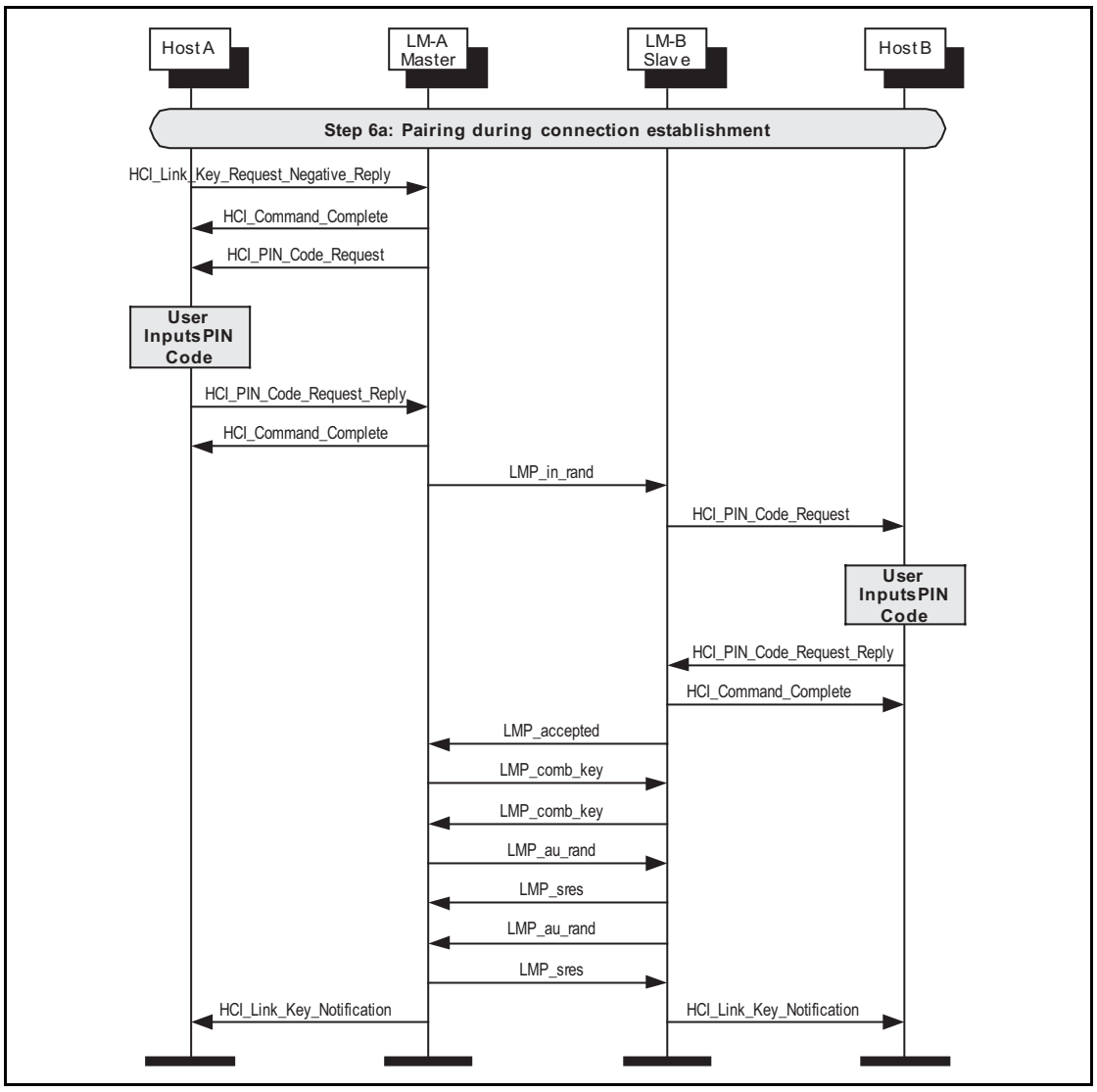


Figure 3.10: Pairing during connection setup.

Step 7b: If a common link key exists between the devices, then pairing is not needed. The LM will have asked for a link key from the host for this connection. If this is a positive reply, then the link key is used for authentication. If the configuration parameter Authentication_Enable is set, then the authentication procedure must be executed. This MSC only shows the case when Authentication_Enable is set on both sides. (See Figure 3.11 on page 637)

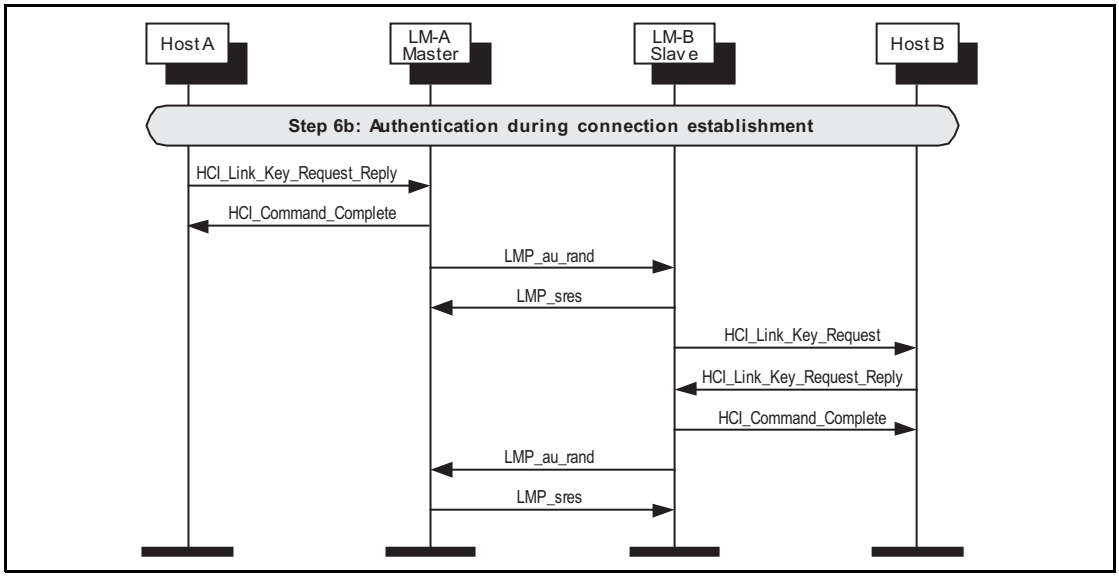


Figure 3.11: Authentication during connection setup.

Step 8: Once the pairing or authentication procedure is successful, the encryption procedure may be started. This MSC only shows the set up of an encrypted point-to-point connection. (See Figure 3.12 on page 637)

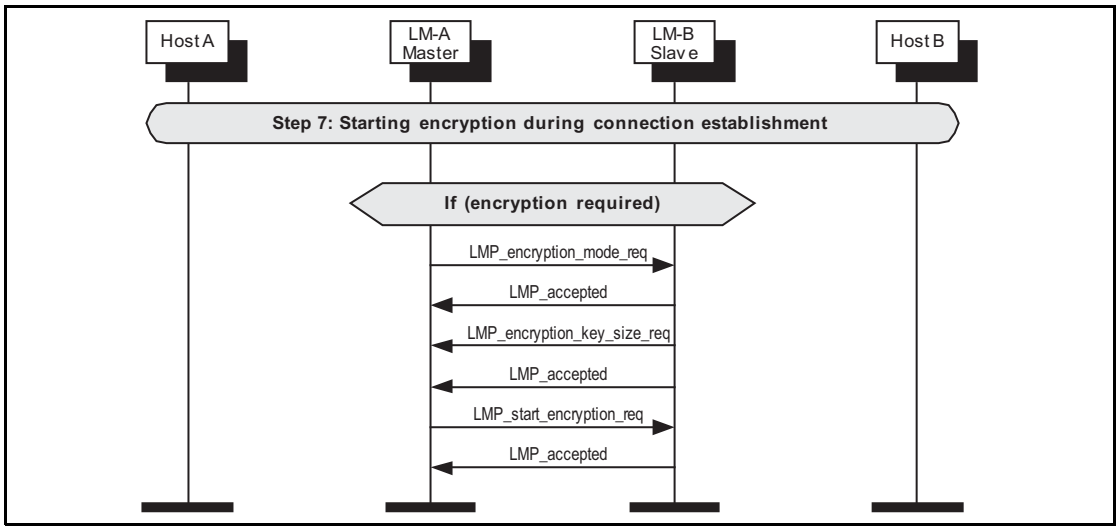


Figure 3.12: Starting encryption during connection setup.

Step 9: The LMs indicate that the connection is setup by sending LMP_setup_complete PDU. This will cause the Host to be notified of the new connection handle, and this connection may be used to send higher layer data such as L2CAP information. (See Figure 3.13 on page 638)

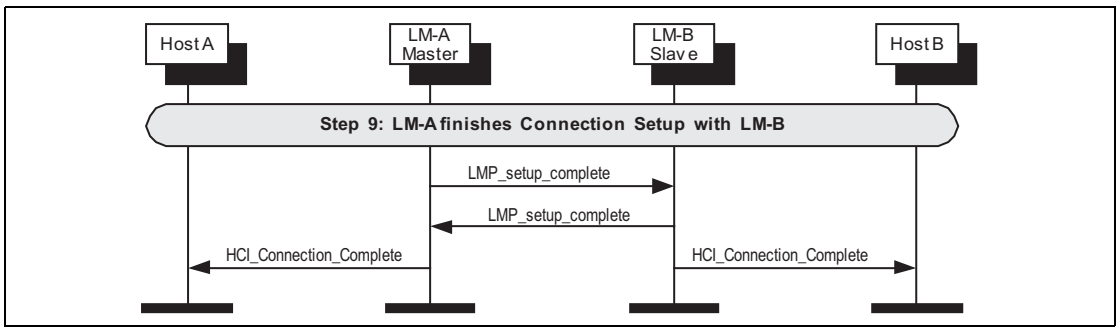


Figure 3.13: LM-A and LM-B finishes connection setup.

Step 10: Once the connection is no longer needed, either device may terminate the connection using the HCI_Disconnect command and LMP_detach message PDU. The disconnection procedure is one-sided and does not need an explicit acknowledgment from the remote LM. The use of ARQ Acknowledgment from the Baseband is needed to ensure that the remote LM has received the LMP_detach PDU. (See Figure 3.13 on page 638)

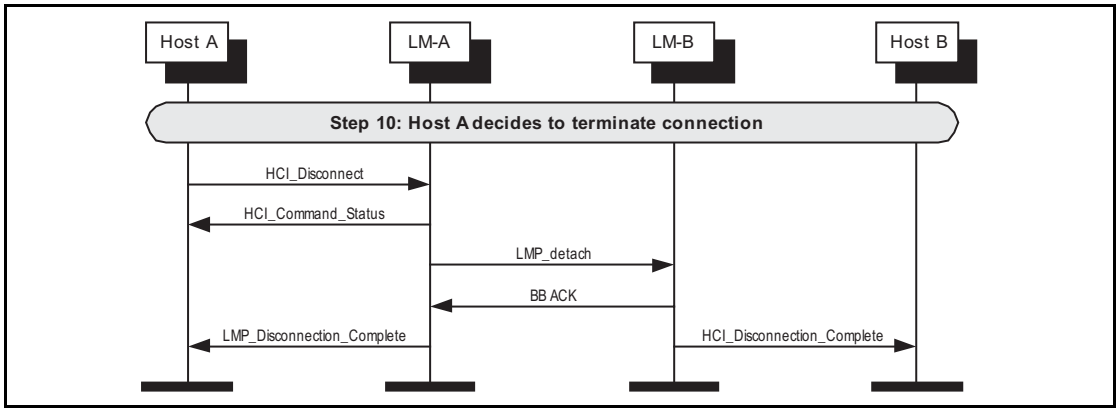


Figure 3.14: Host A decides to disconnect.

4 OPTIONAL ACTIVITIES AFTER ACL CONNECTION ESTABLISHMENT

4.1 AUTHENTICATION REQUESTED

Step 1: Authentication can be explicitly executed at any time after a connection has been established. If no Link Key is available then the Link Key is required from the Host. (See Figure 4.1 on page 639)

Note: If the Controller or LM and the Host do not have the Link Key a PIN Code Request event will be sent to the Host to request a PIN Code for pairing. A procedure identical to that used during Connection Setup (Section 3.1, Step 7a:) will be used. (See Figure 3.9 on page 635)

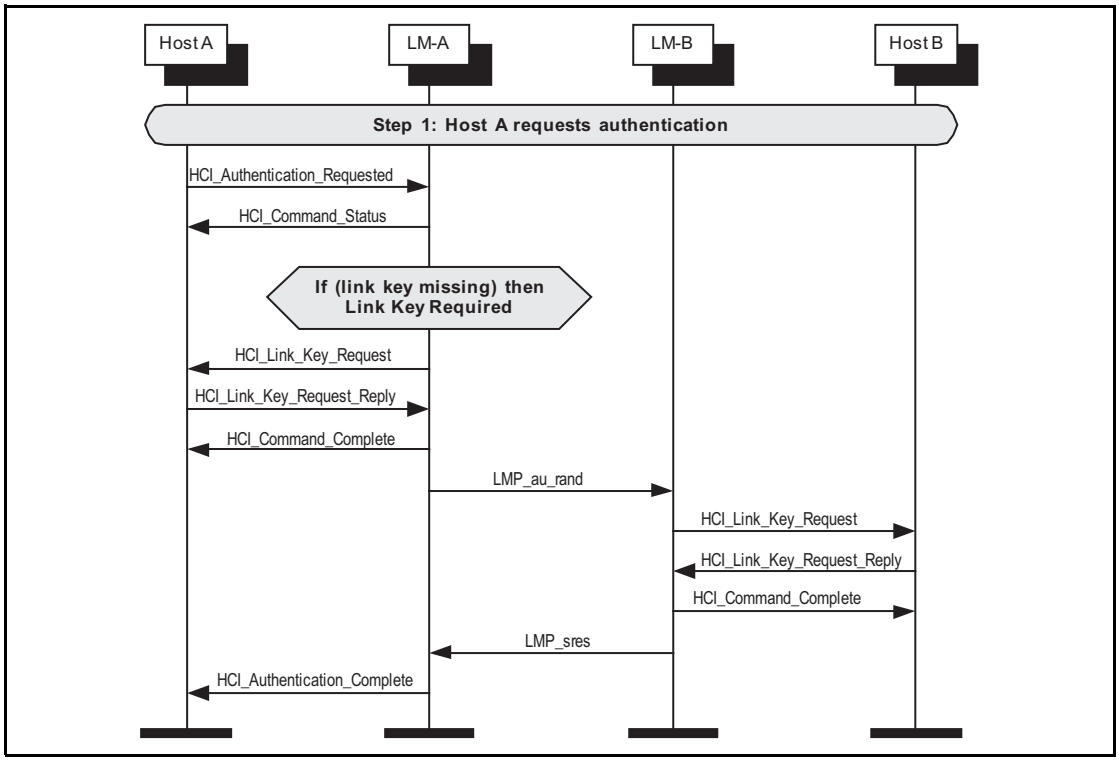


Figure 4.1: Authentication requested.

4.2 SET CONNECTION ENCRYPTION

Step 1: The host may at any time turn on encryption using the HCI_Set_Connection_Encryption command. This command can be originated from either the master or slave sides. Only the master side is shown in Figure 4.2 on page 640. If this command is sent from a slave, the only difference is that the LMP_encryption_mode_req PDU will be sent from the slave. The LMP_encryption_key_size_req and LMP_start_encryption_req PDUs will always be requested from the master. (See Figure 4.2 on page 640)

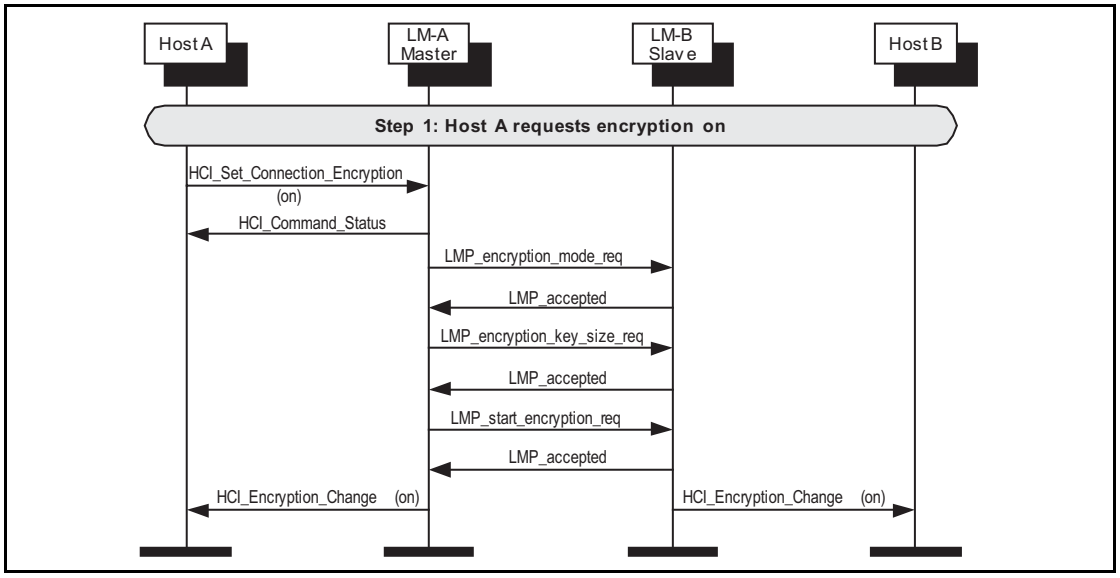


Figure 4.2: Encryption requested.

Step 2: To terminate the use of encryption, The HCI_Set_Connection_Encryption command is used. (See Figure 4.3 on page 640)

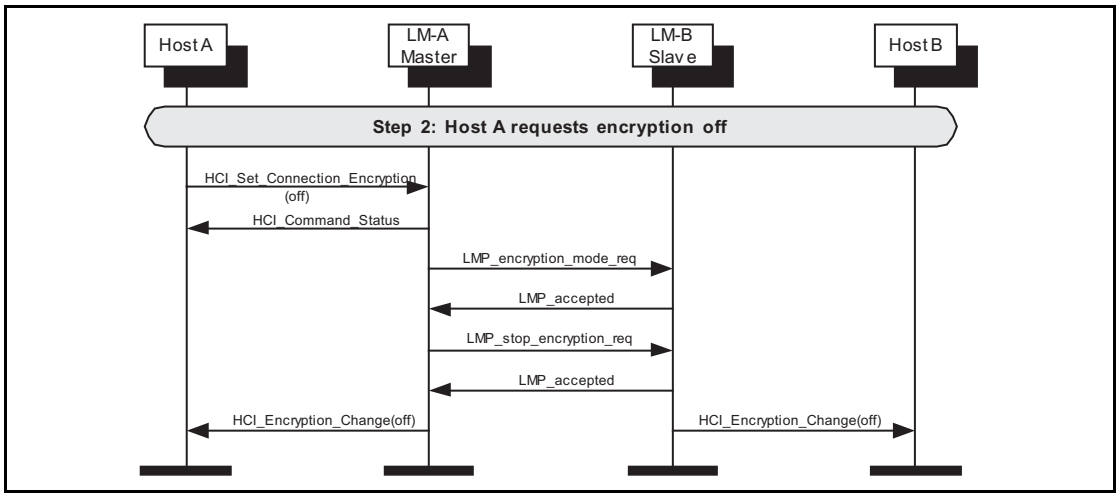


Figure 4.3: Encryption off requested.

4.3 CHANGE CONNECTION LINK KEY

Step 1: The master host (Host A) may change the connection link key using the HCI_Change_Connection_Link_Key command. A new link key will be generated and the hosts will be notified of this new link key. (See Figure 4.4 on page 641).

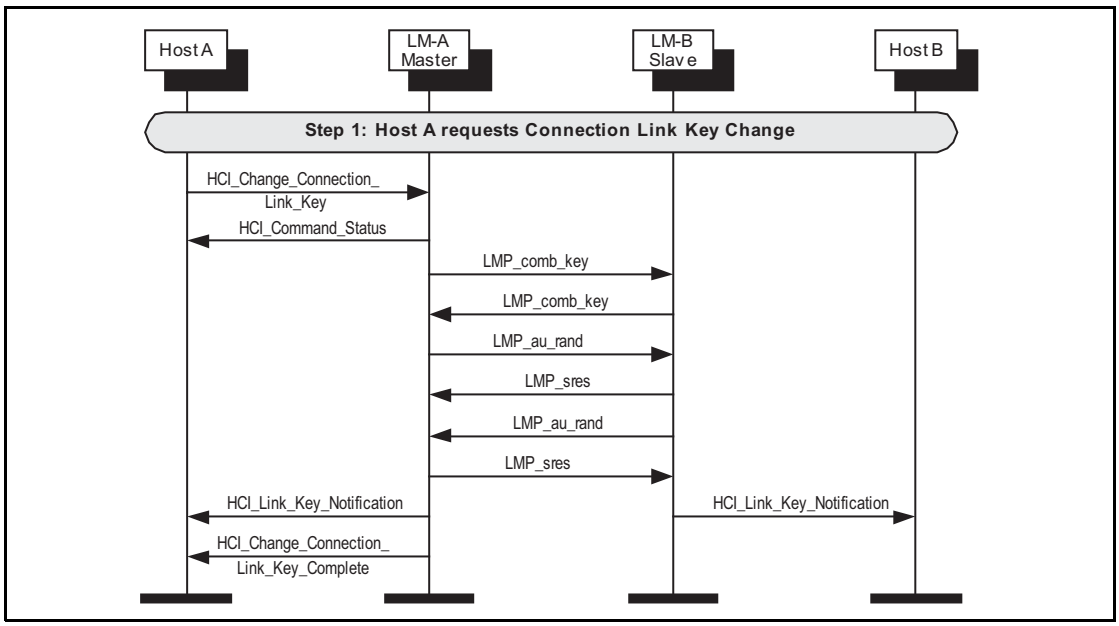


Figure 4.4: Change connection link key.

4.4 MASTER LINK KEY

Step 1: The host changes to a Master Link Key from a Semi-permanent Link Key using the HCI_Master_Link_Key command. (See Figure 4.5 on page 642)

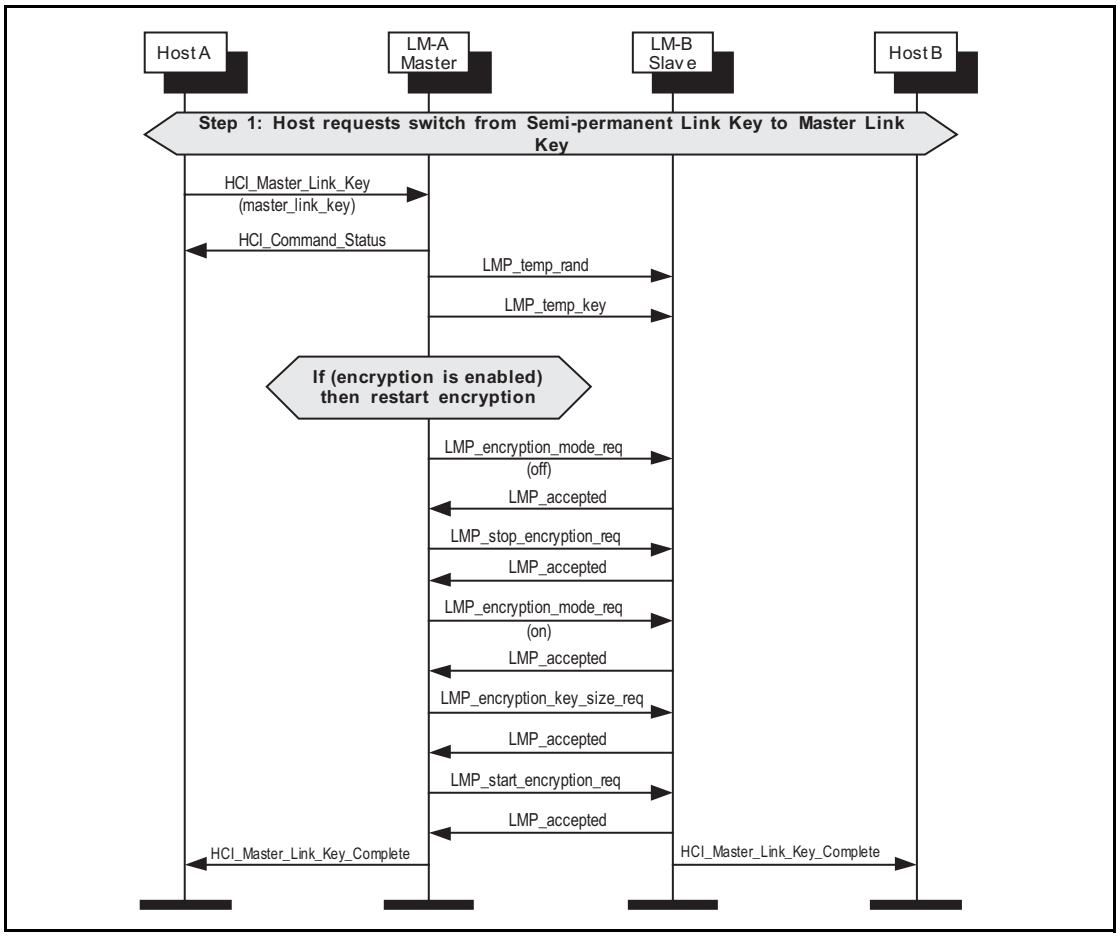


Figure 4.5: Change to master link key.

Step 2: The host changes to a Semi-permanent Link Key from a Master Link Key using the HCI_Master_Link_Key command. (See Figure 4.6 on page 643)

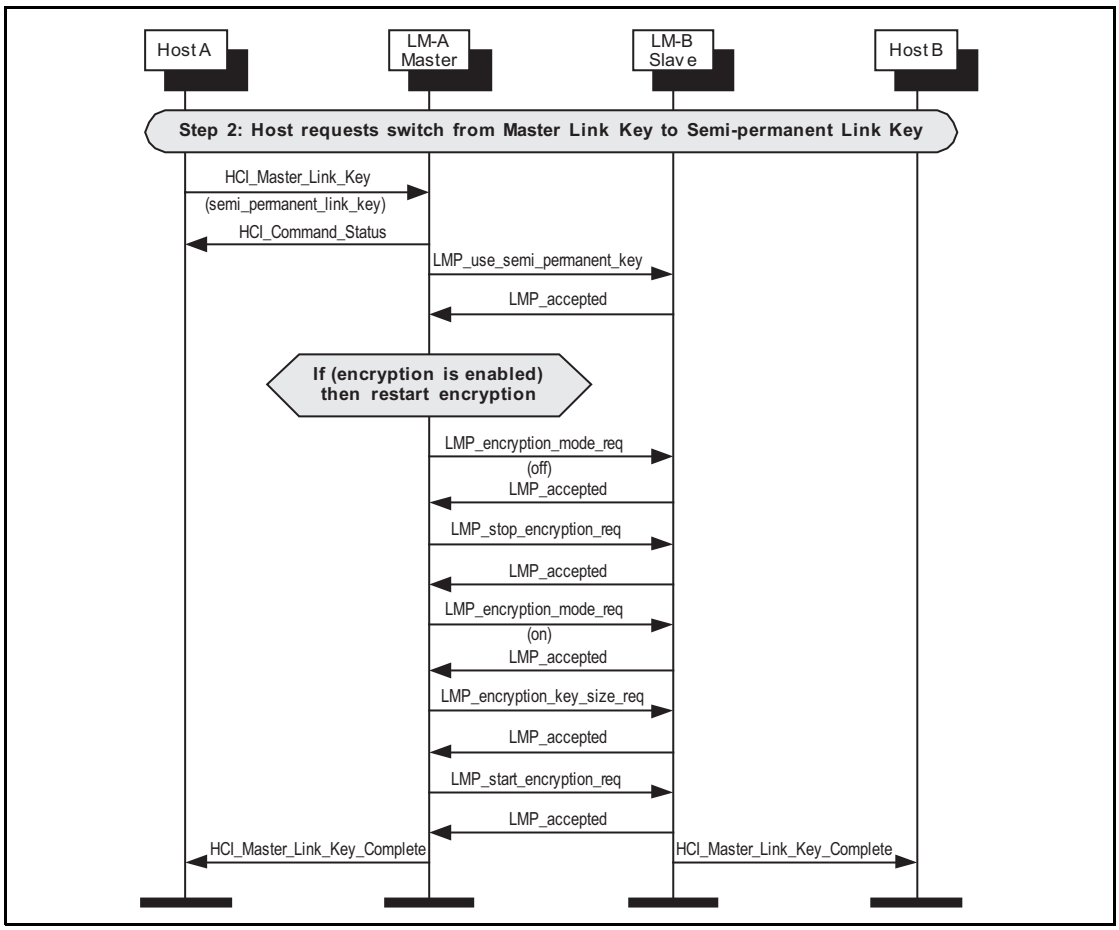


Figure 4.6: Change to semi permanent link key.

4.5 READ REMOTE SUPPORTED FEATURES

Using the HCI_Read_Remote_Supported_Features command the supported LMP Features of a remote device can be read. (See Figure 4.7 on page 644)

If the remote supported features have been obtained previously then the Controller may return them without sending any LMP PDUs.

Step 1: The host requests the supported features of a remote device.

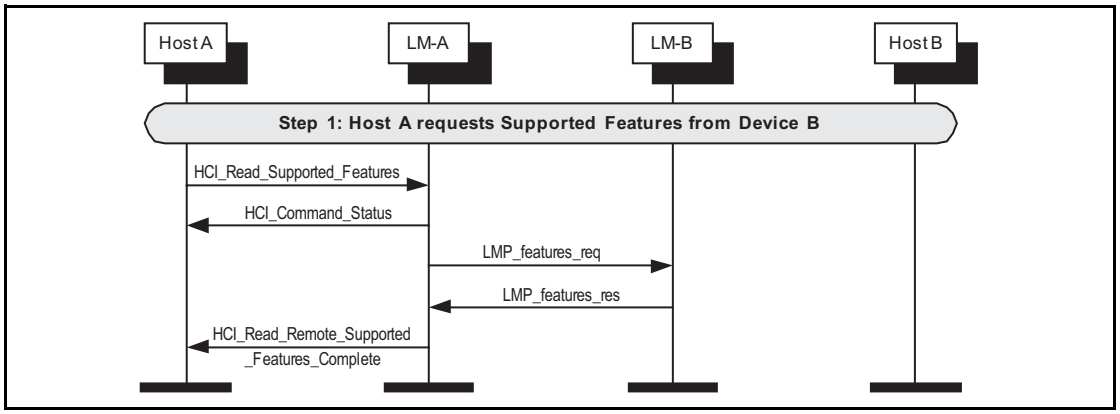


Figure 4.7: Read remote supported features.

4.6 READ REMOTE EXTENDED FEATURES

Using the HCI_Read_Remote_Extended_Features command the extended LMP features of a remote device can be read. (See Figure 4.8 on page 644)

If the remote extended features have been obtained previously then the Controller may return them without sending any LMP PDUs.

Step 1: The host requests the extended features of a remote device.

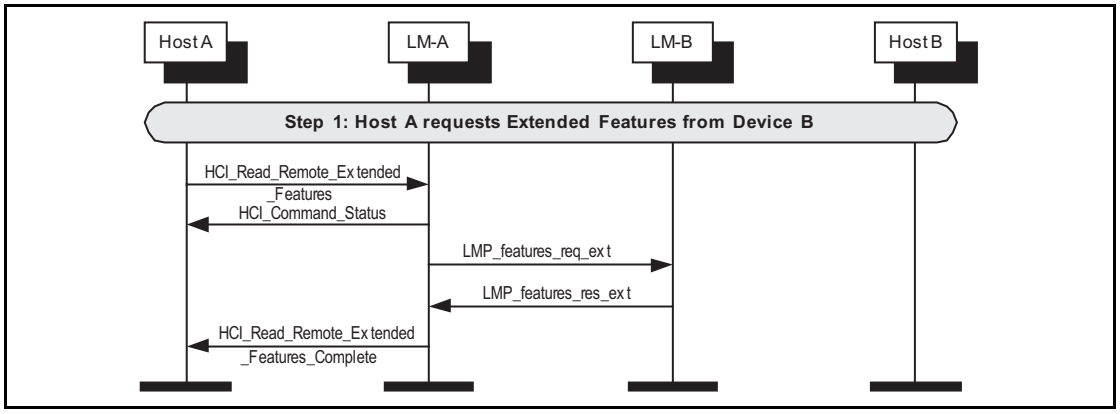


Figure 4.8: Read remote extended features.

4.7 READ CLOCK OFFSET

Using the HCI_Read_Clock_Offset command the device acting as the master can read the Clock Offset of a slave. The Clock Offset can be used to speed up the paging procedure in a later connection attempt. If the command is requested from the slave device, the Controller will directly return a Command Status event and a Read Clock Offset Complete event without sending any LMP PDUs. (See Figure 4.9 on page 645)

Step 1: The host requests the clock offset of a remote device.

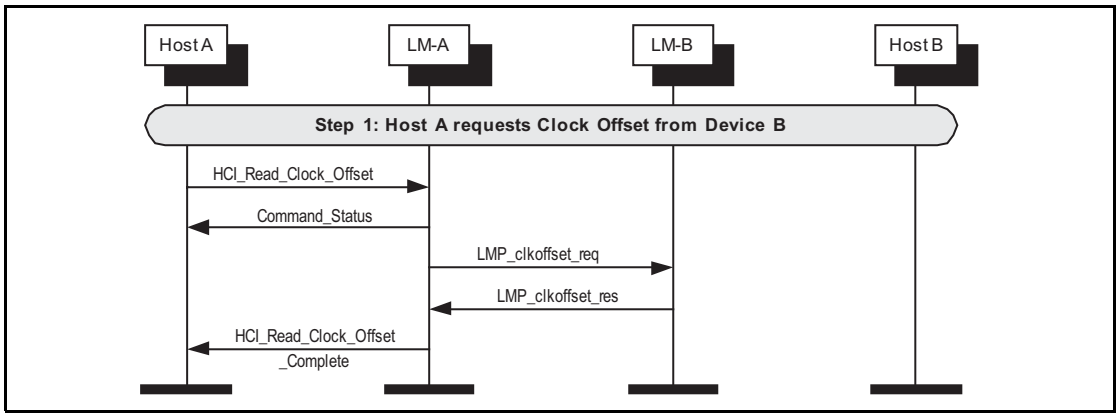


Figure 4.9: Read clock offset.

4.8 READ REMOTE VERSION INFORMATION

Using the HCI_Read_Remote_Version_Information command the version information of a remote device can be read. (See Figure 4.10 on page 645)

If the remote version information has been obtained previously then the Controller may return them without sending any LMP PDUs.

Step 1: The host requests the version information of a remote device.

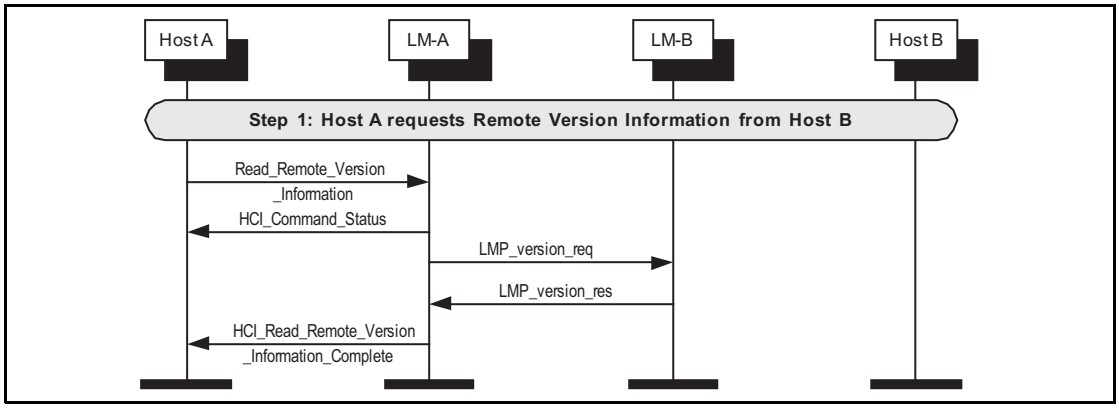


Figure 4.10: Read remote version information.

4.9 QOS SETUP

Using the HCI_Flow_Specification command the Quality of Service (QoS) and Flow Specification requirements of a connection can be notified to a Controller. The Controller may then change the quality of service parameters with a remote device. (See Figure 4.11 on page 646)

Step 1: The host sends QoS parameters to a remote device.

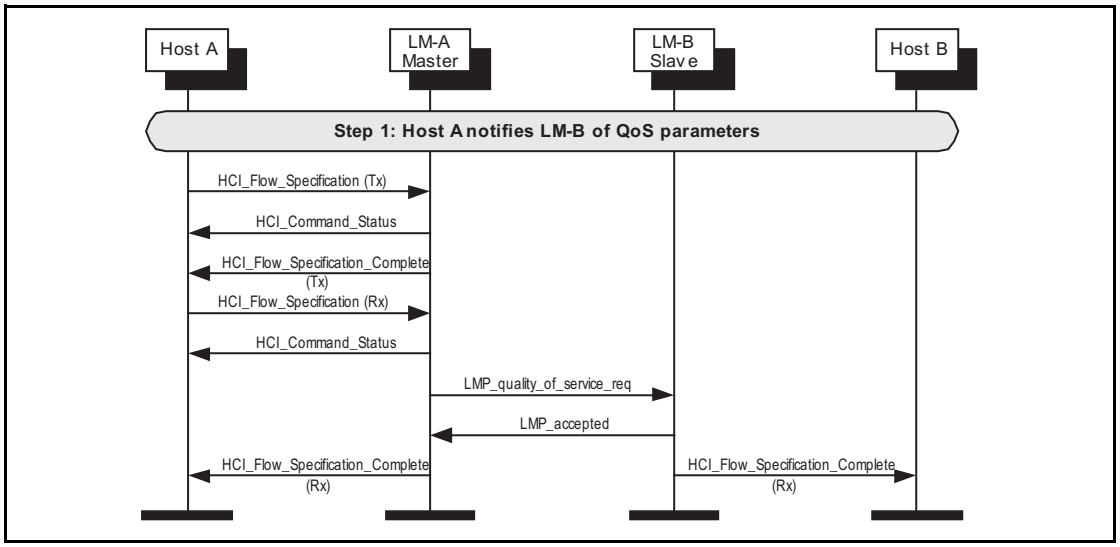


Figure 4.11: QoS flow specification.

4.10 SWITCH ROLE

The HCI_Switch_Role command can be used to explicitly switch the current master / slave role of the local device with the specified device.

Step 1a: The master host (A) requests a role switch with a slave. This will send the switch request, and the slave will respond with the slot offset and accepted. (See Figure 4.12 on page 646)

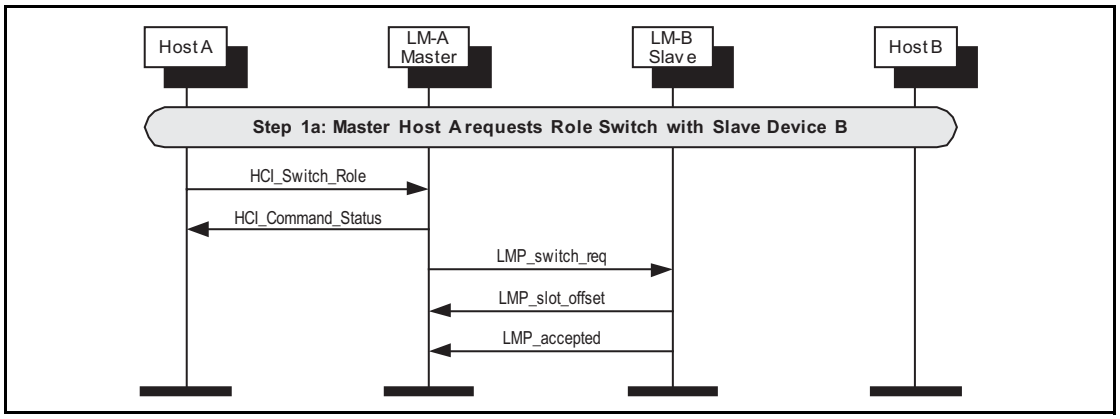


Figure 4.12: Master requests role switch.

Step 1b: The slave host (B) requests a role switch with a master. This will send the slot offset and the switch request, and the master will respond with a LMP_accepted PDU. (See Figure 4.13 on page 647)

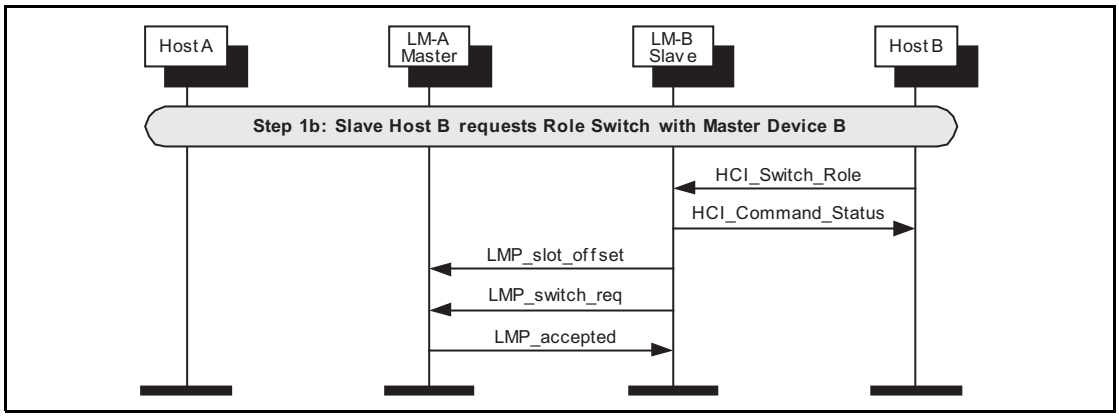


Figure 4.13: Slave requests role switch.

Step 2: The role switch is performed by doing the TDD switch and piconet switch. Finally an HCI_Role_Change event is sent on both sides.
(See Figure 4.14 on page 647)

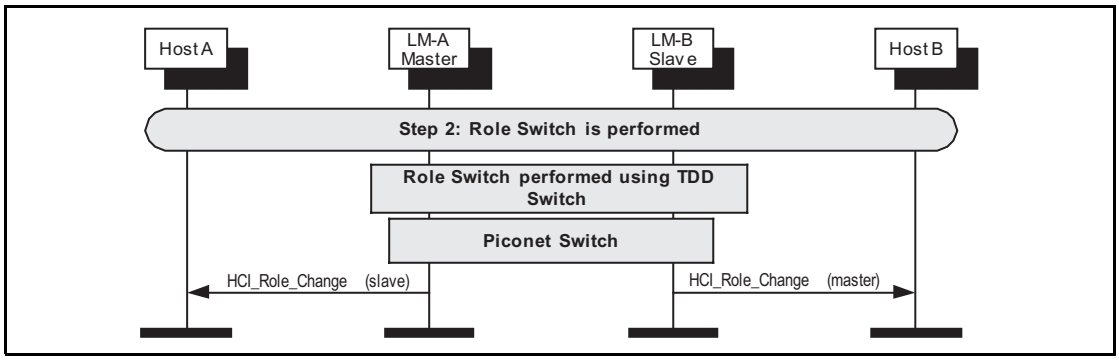


Figure 4.14: Role switch is performed.

5 SYNCHRONOUS CONNECTION ESTABLISHMENT AND DETACHMENT

5.1 SYNCHRONOUS CONNECTION SETUP

Using the HCI_Setup_Synchronous_Connection command, a host can add a synchronous logical channel to the link. A synchronous logical link can be provided by creating a SCO or an eSCO logical transport.

Note: An ACL Connection must be established before a synchronous connection can be created.

Step 1a: Master device requests a synchronous connection with a device. (See Figure 5.1 on page 649)

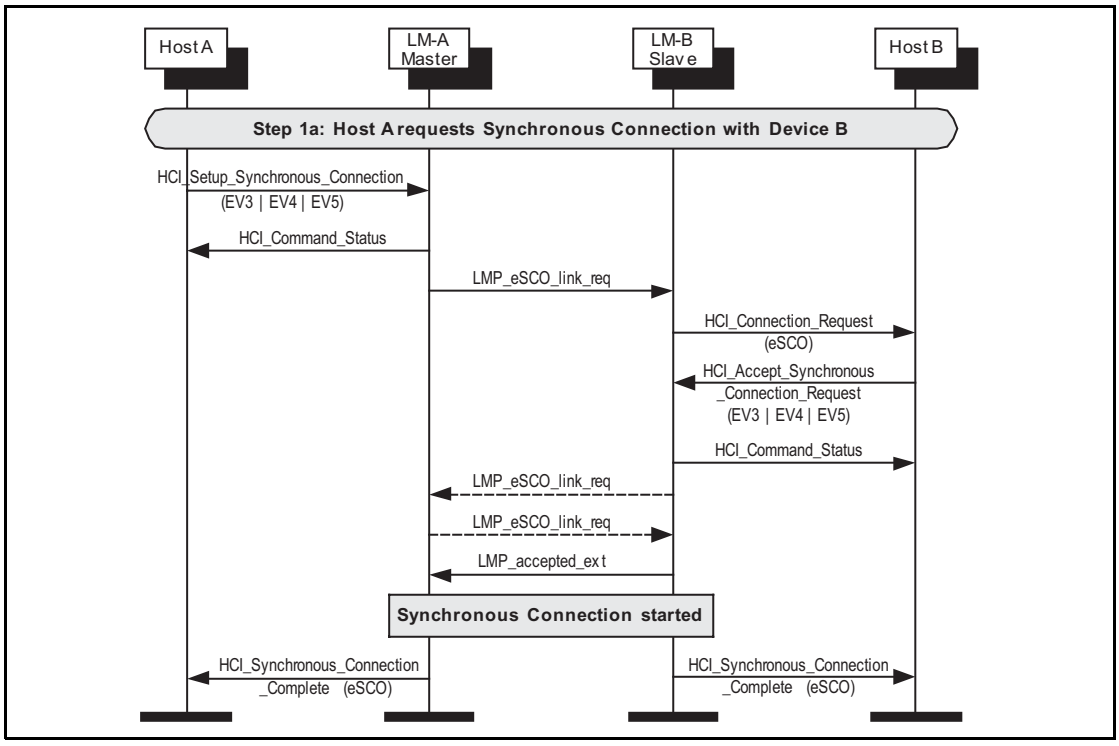


Figure 5.1: Master requests synchronous EV3, EV4 OR EV5 connection.

Step 1b: Slave device requests a synchronous connection with a device. (See Figure 5.2 on page 650)

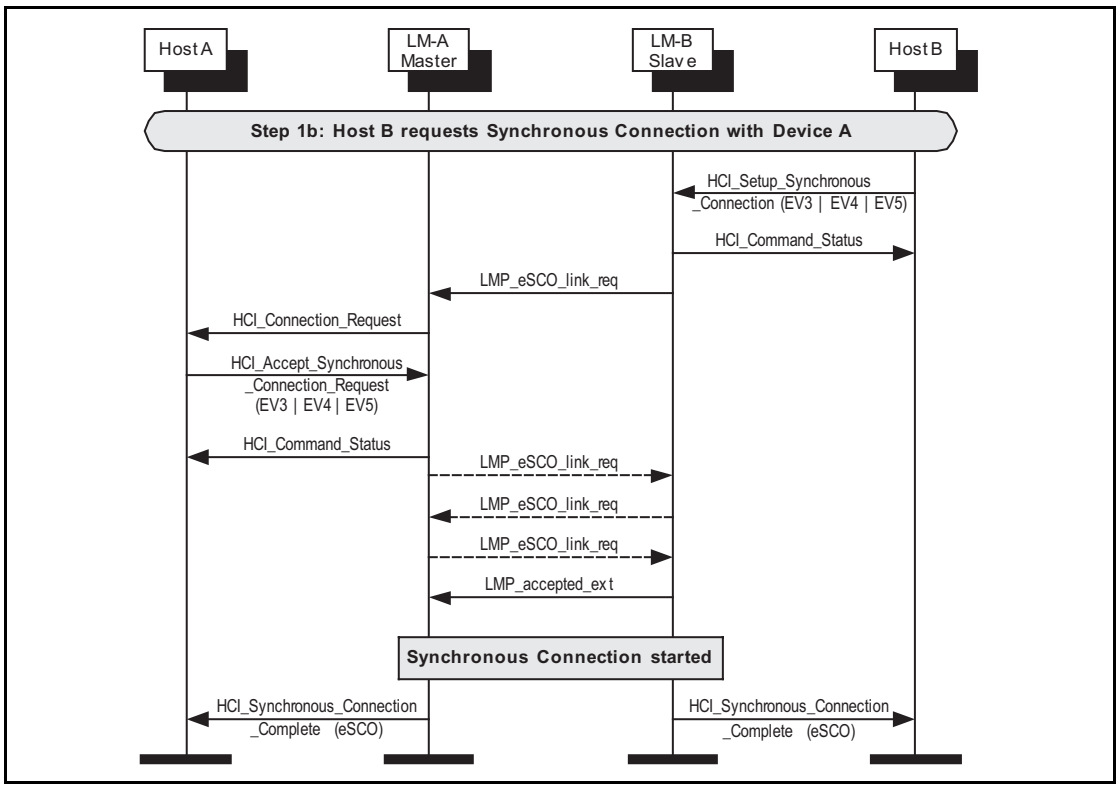


Figure 5.2: Slave requests synchronous EV3, EV4 OR EV5 connection.

Step 1c: Master device requests a SCO connection with a device. (See Figure 5.3 on page 650)

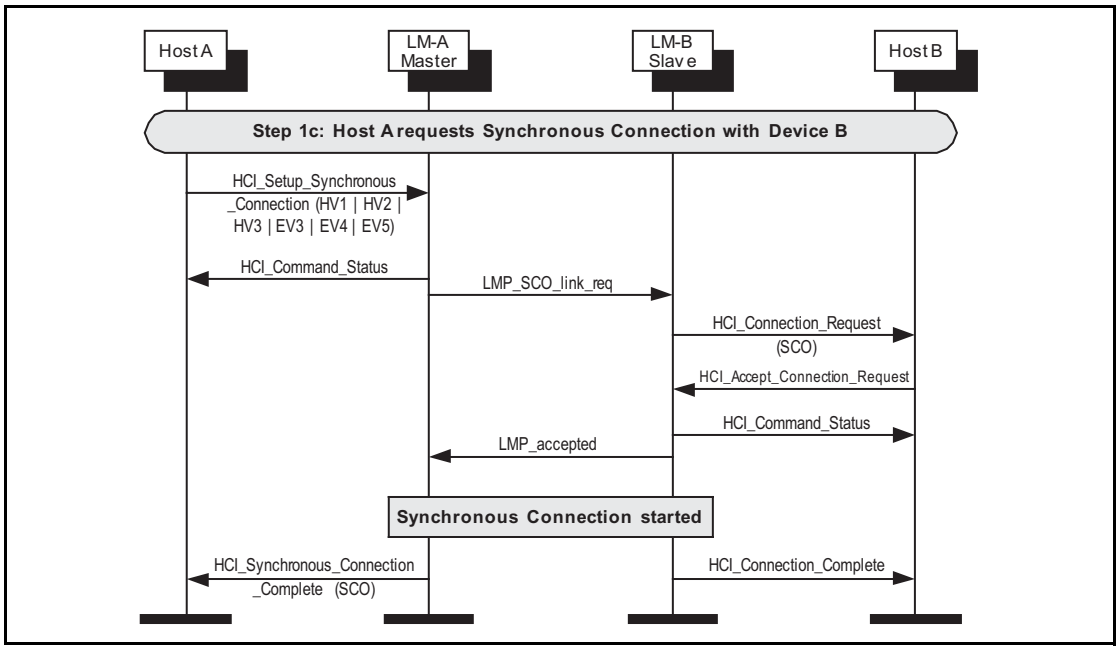


Figure 5.3: Master requests synchronous connection using SCO.

Step 1d: Master device requests a SCO connection with a device.
(See Figure 5.4 on page 651)

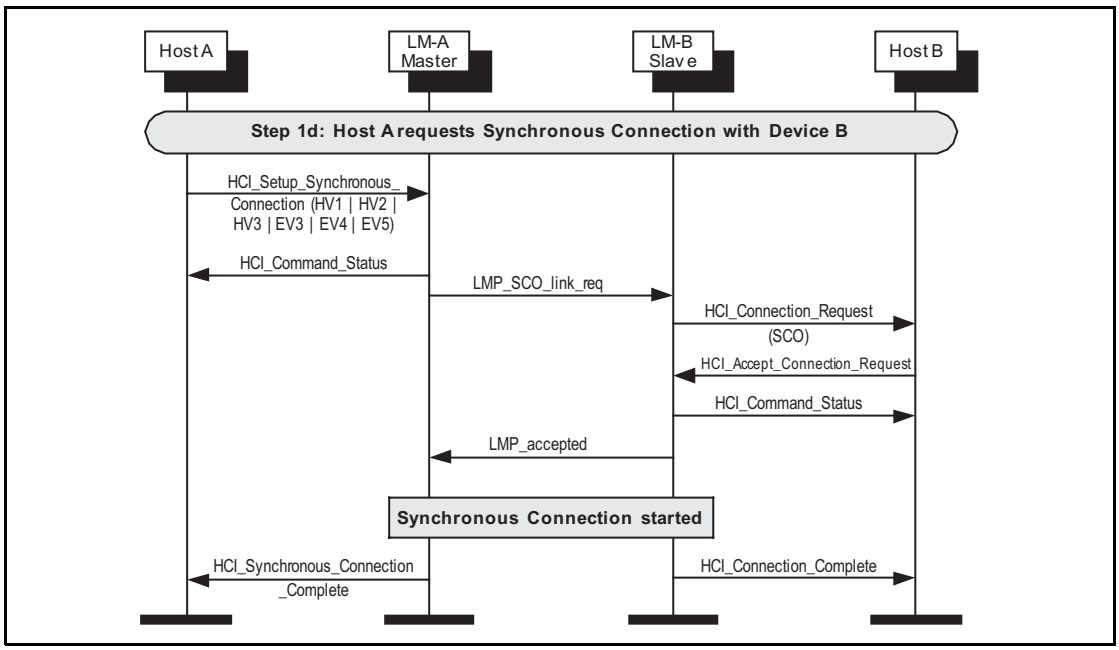


Figure 5.4: Master requests synchronous connection with legacy slave.

Step 1e: Host device requests a SCO connection with a device.
(See Figure 5.5 on page 651)

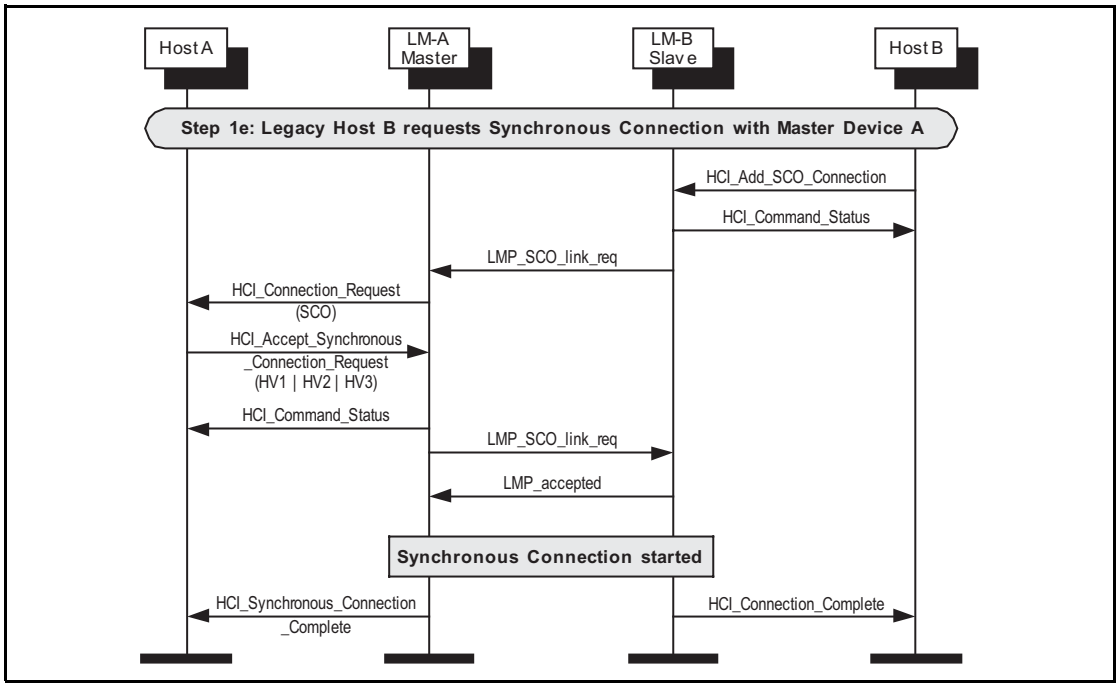


Figure 5.5: Any device that supports only SCO connections requests a synchronous connection with a device.

Step 2a: Master renegotiates eSCO connection (See Figure 5.6 on page 652).

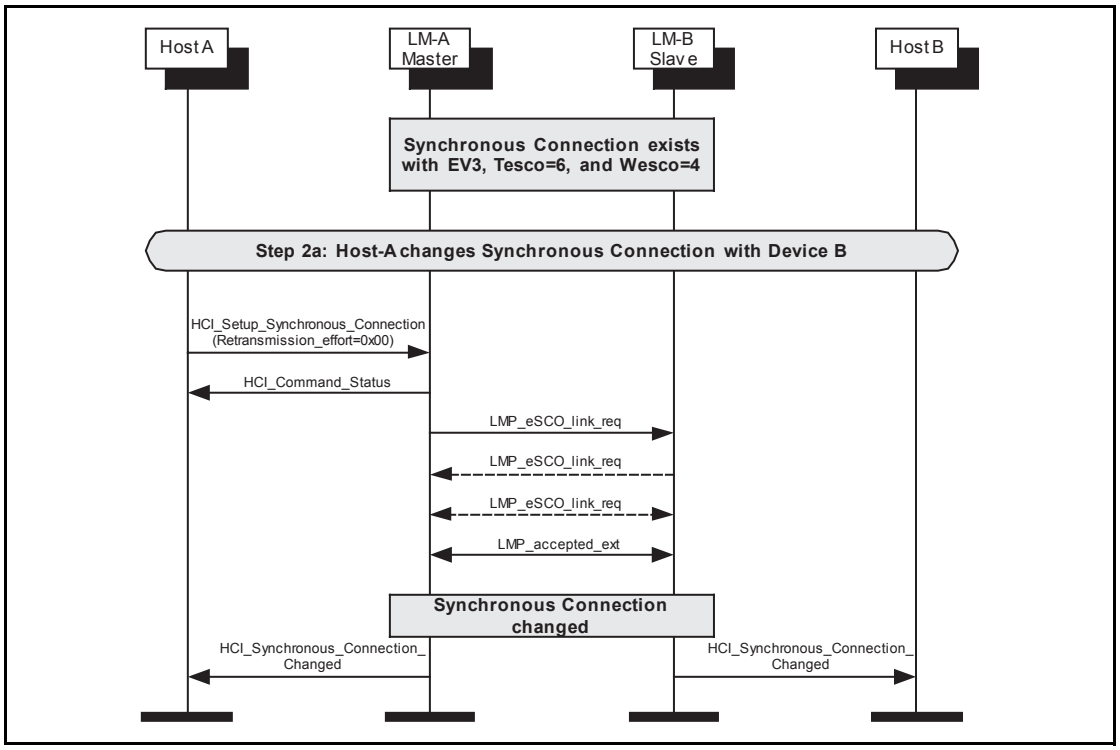


Figure 5.6: Master renegotiates eSCO connection.

Step 2b: Slave renegotiates eSCO connection (See Figure 5.7 on page 652).

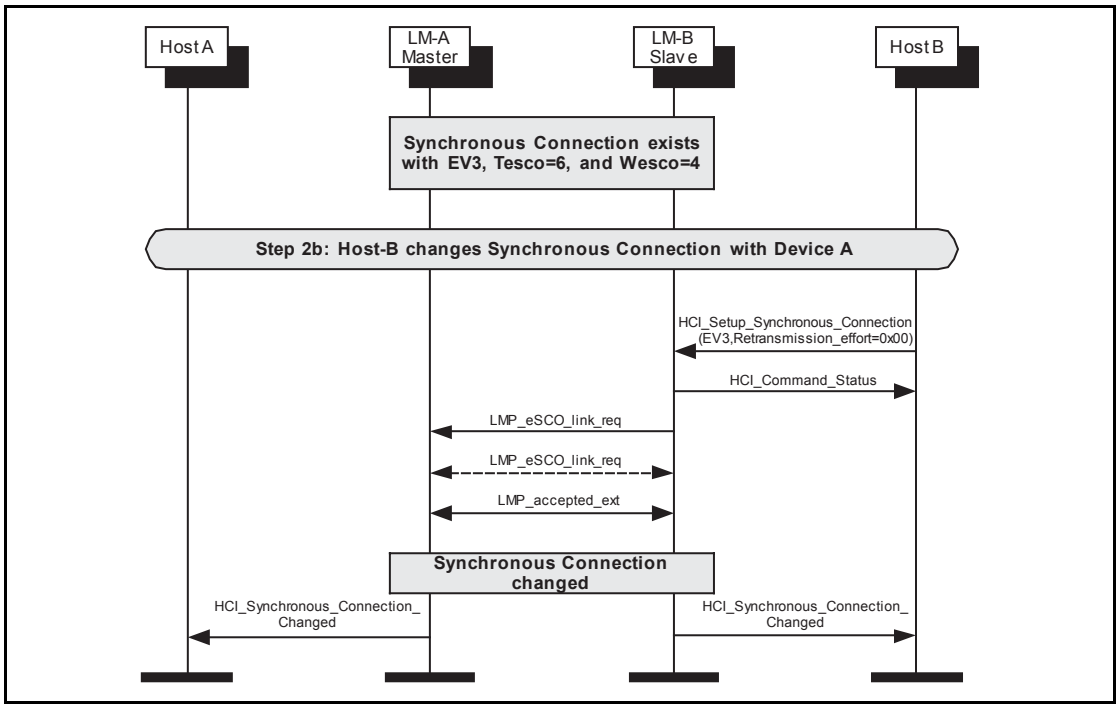


Figure 5.7: Slave renegotiates eSCO connection.

Step 3a: eSCO Disconnection. (See Figure 5.8 on page 653)

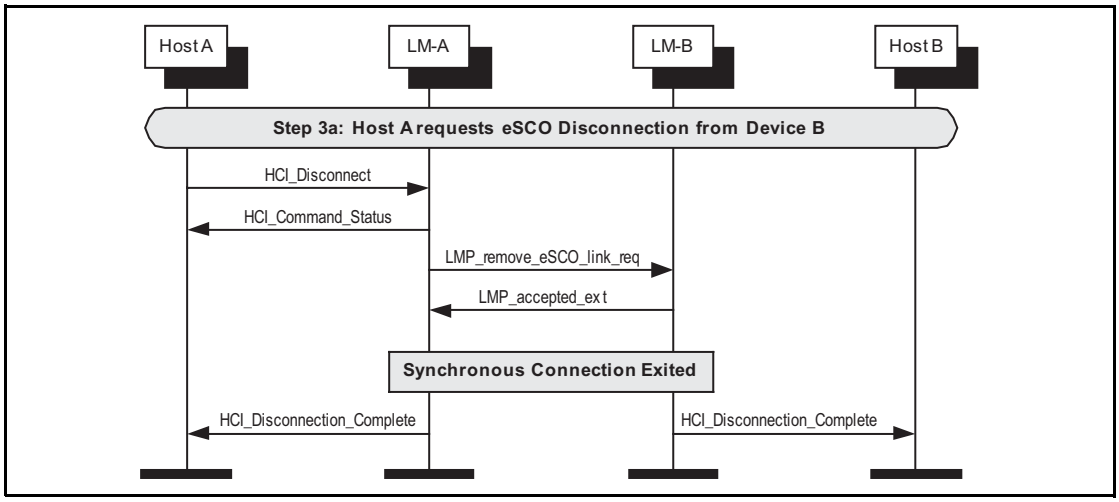


Figure 5.8: Synchronous disconnection of eSCO connection.

Step 3b: SCO Disconnection. (See Figure 5.9 on page 653)

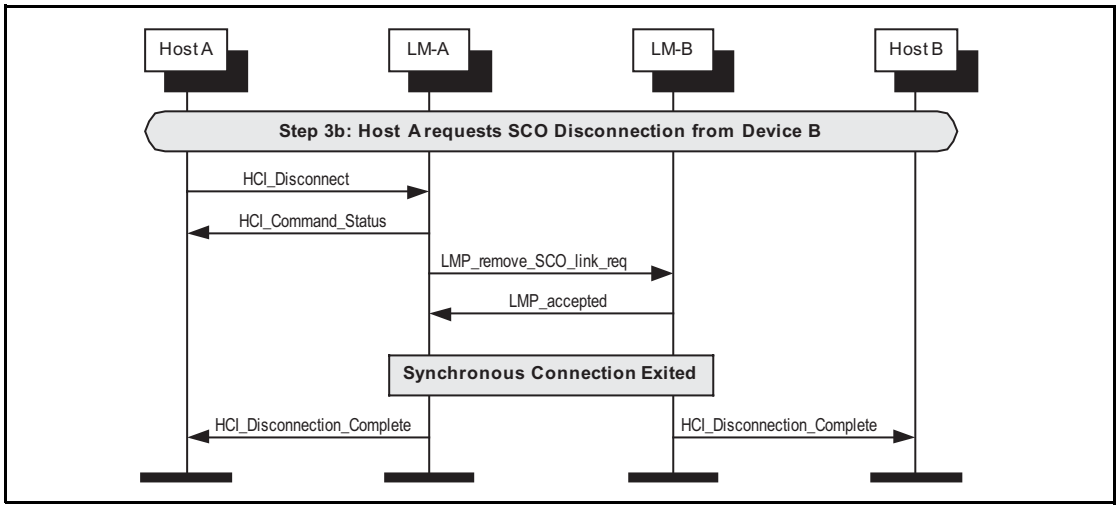


Figure 5.9: Synchronous disconnection of SCO connection.

6 SNIFF, HOLD AND PARK

Entry into sniff mode, hold mode or park state requires an established ACL Connection.

6.1 SNIFF MODE

The HCI_Sniff_Mode command is used to enter sniff mode. The HCI_Exit_Sniff_Mode command is used to exit sniff mode.

Step 1: Host requests to enter sniff mode. Multiple LMP_sniff_req PDUs may be sent as the parameters for sniff mode are negotiated.
(See Figure 6.1 on page 655)

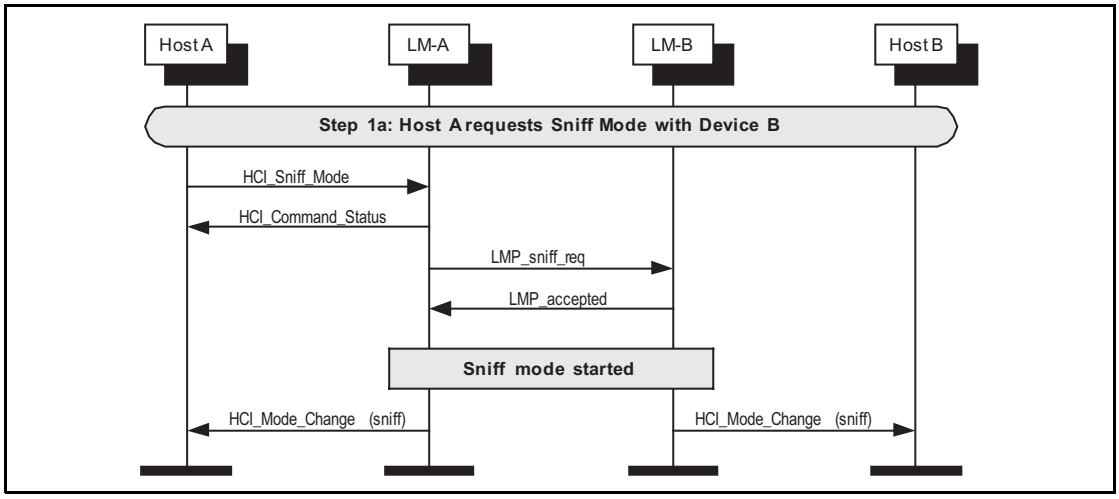


Figure 6.1: Sniff mode request.

Step 2: Host requests to exit sniff mode. (See Figure 6.2 on page 655)

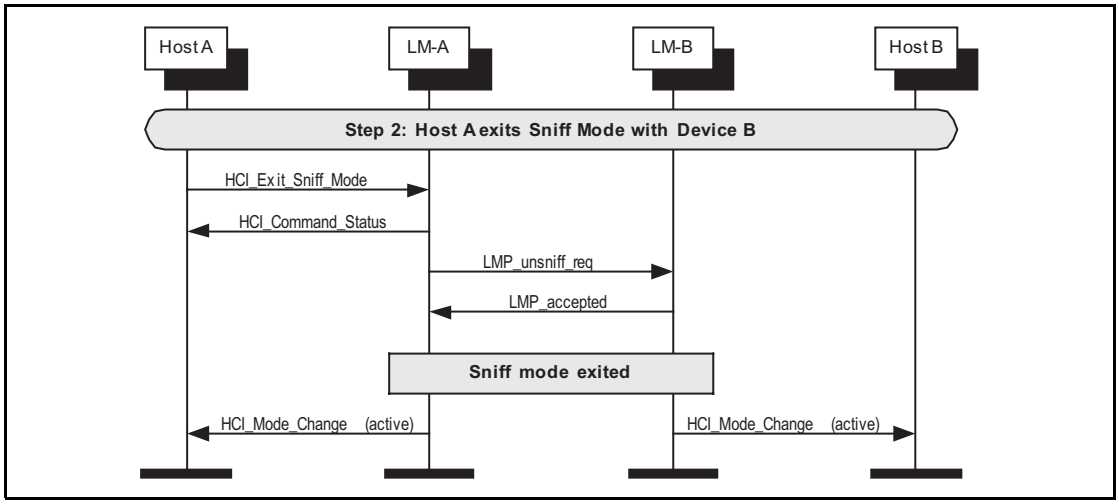


Figure 6.2: Exit sniff mode request.

6.2 HOLD MODE

The HCI_Hold_Mode command can be used to place a device into hold mode. The Controller may do this by either negotiating the hold mode parameters or forcing hold mode. Hold mode will automatically end after the negotiated length of time.

Step 1a: A host requests hold mode. (See Figure 6.3 on page 656)

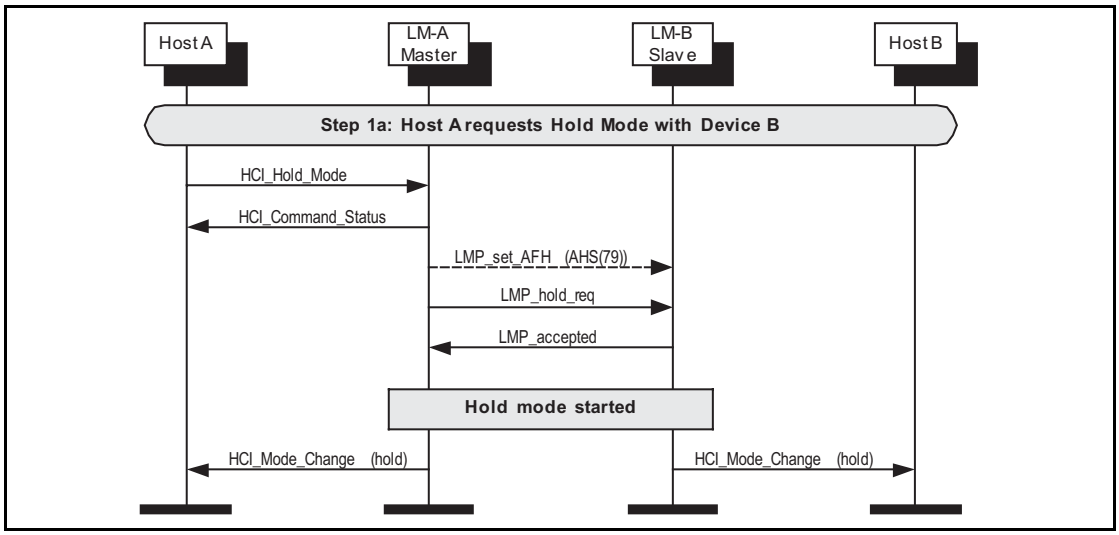


Figure 6.3: Hold request.

Step 1b: A host may force hold mode. (See Figure 6.4 on page 656)

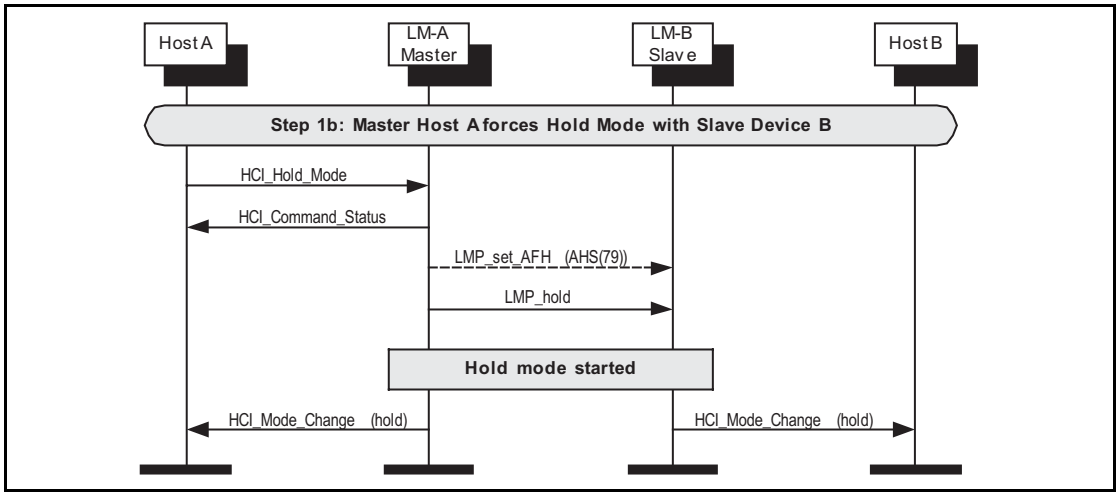


Figure 6.4: Master forces hold mode.

Step 1c: A slave device requests hold mode. (See Figure 6.5 on page 657)

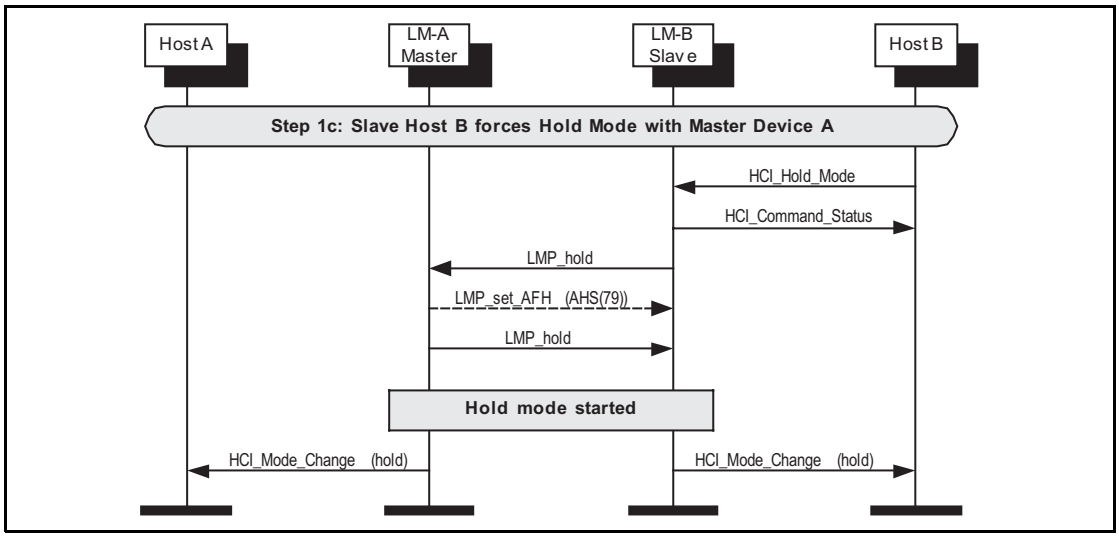


Figure 6.5: Slave forces hold mode.

Step 2: When hold mode completes the hosts are notified using the HCI_Mode_Change event. (See Figure 6.6 on page 657)

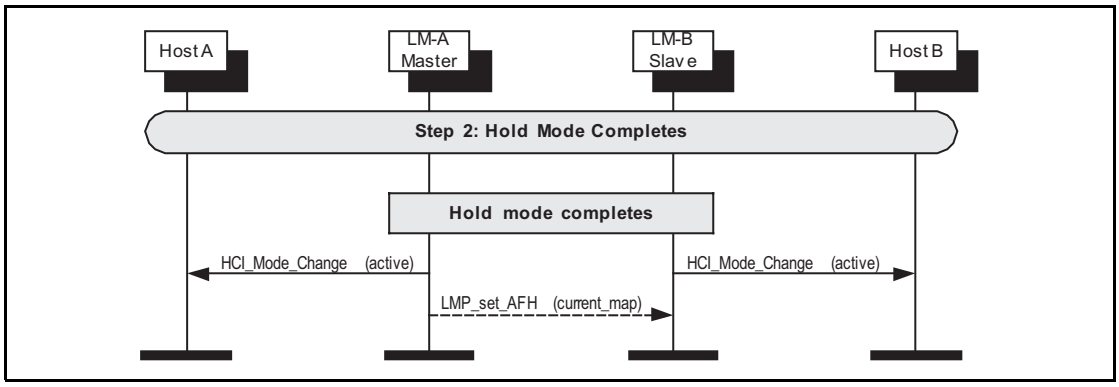


Figure 6.6: Hold mode completes.

6.3 PARK STATE

Park state can be entered by using the HCI_Park_State command.

Step 1a: The master requests to place the slave in the park state. Before sending the LMP_park_req PDU, the master may disable AFH by setting the connection into AHS(79). (See Figure 6.7 on page 658)

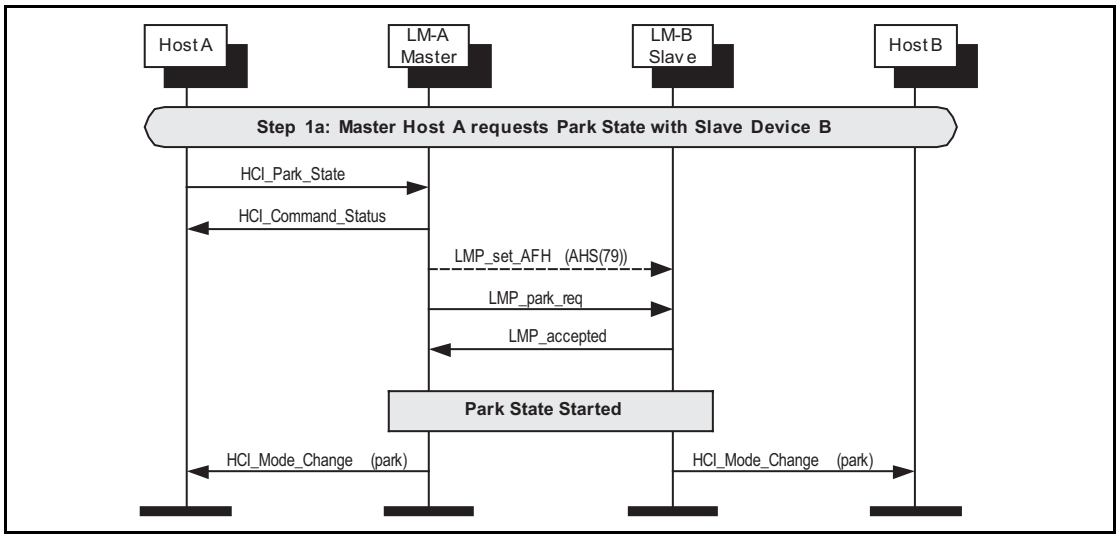


Figure 6.7: Park state request from master.

Step 1b: The slave requests to be placed in the park state. Before sending the LMP_park_req PDU back to the slave, the master may disable AFH by setting the connection into AHS(79). (See Figure 6.8 on page 658)

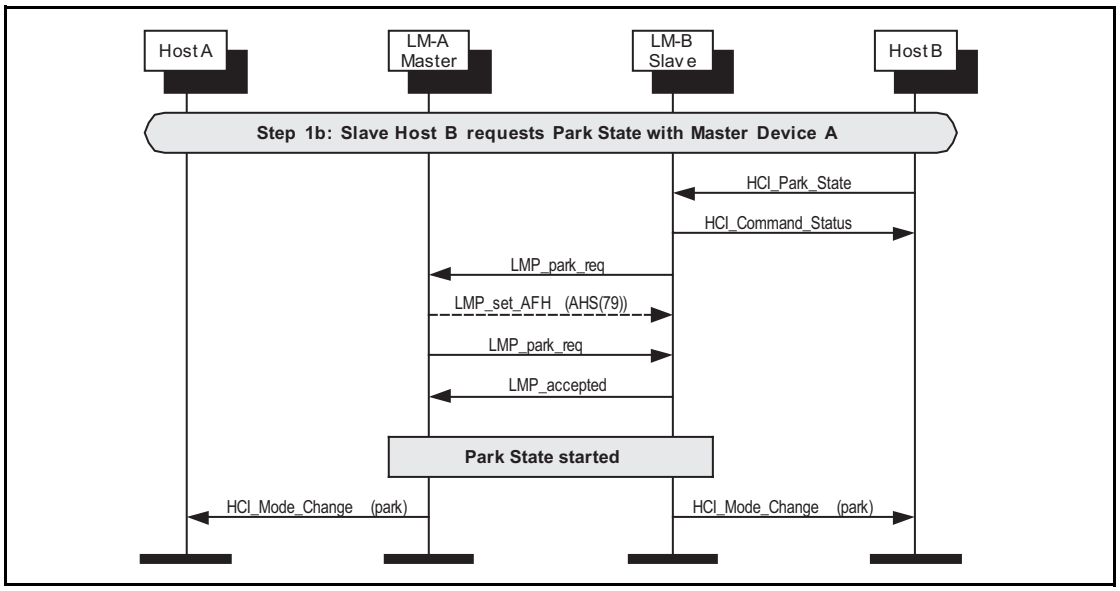


Figure 6.8: Park state request from slave.

Step 2: When in the park state, a slave still needs to unparked for link supervision purposes. The master sends an LMP_unpark_PM_ADDR_req PDU or an LMP_unpark_BD_ADDR_req PDU to the slave during the beacon. Only the PM_ADDR version is illustrated in the figure. (See Figure 6.9 on page 659)

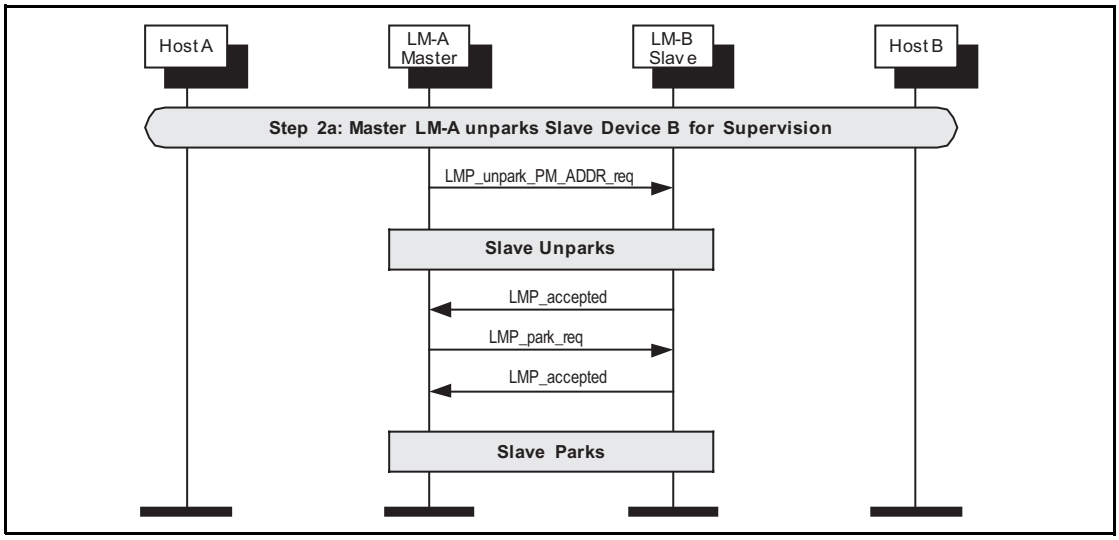


Figure 6.9: Master unparks slave for supervision.

Step 3a: A master may unpark a slave to exit park state. The master should re-enable AFH by setting the current AFH channel map to the unparked slave. (See Figure 6.10 on page 659)

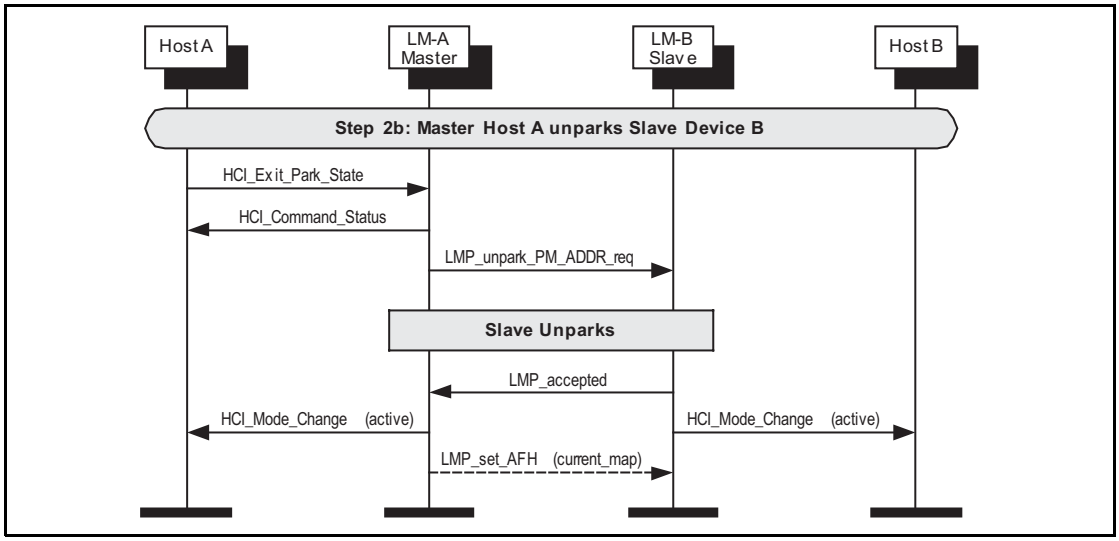


Figure 6.10: Master exits park state with slave.

Step 3b: A slave may request to be unparked by sending a message in an access window. It will then receive instructions from the master to unpark. The master should re-enable AFH by setting the current AFH channel map to the unparked slave. (See Figure 6.11 on page 660)

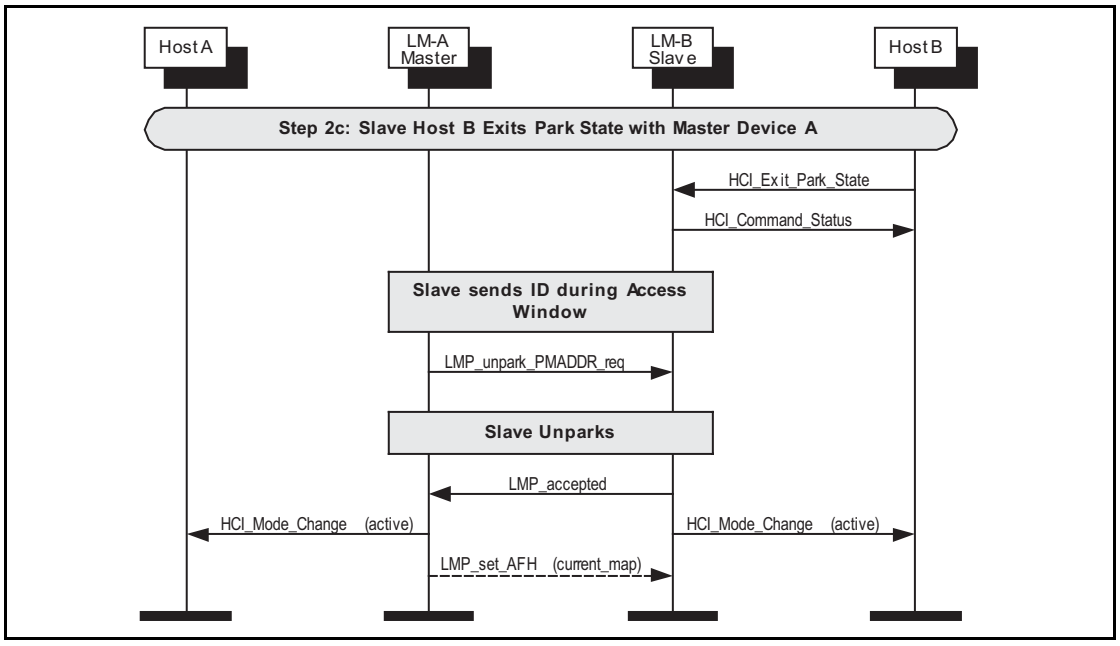


Figure 6.11: Slave exits park state with master.

7 BUFFER MANAGEMENT, FLOW CONTROL

Buffer management is very important for resource limited devices. This can be achieved on the Host Controller Interface using the HCI_Read_Buffer_Size command, and the HCI_Number_Of_Completed_Packets event, and the HCI_Set_Host_Controller_To_Host_Flow_Control, HCI_Host_Buffer_Size and HCI_Host_Number_Of_Completed_Packets commands.

Step 1: During initialization, the host reads the buffer sizes available in the Controller. When an HCI Data Packet has been transferred to the remote device, and a Baseband acknowledgement has been received for this data, then an HCI_Number_Of_Completed_Packets event will be generated. (See Figure 7.1 on page 661)

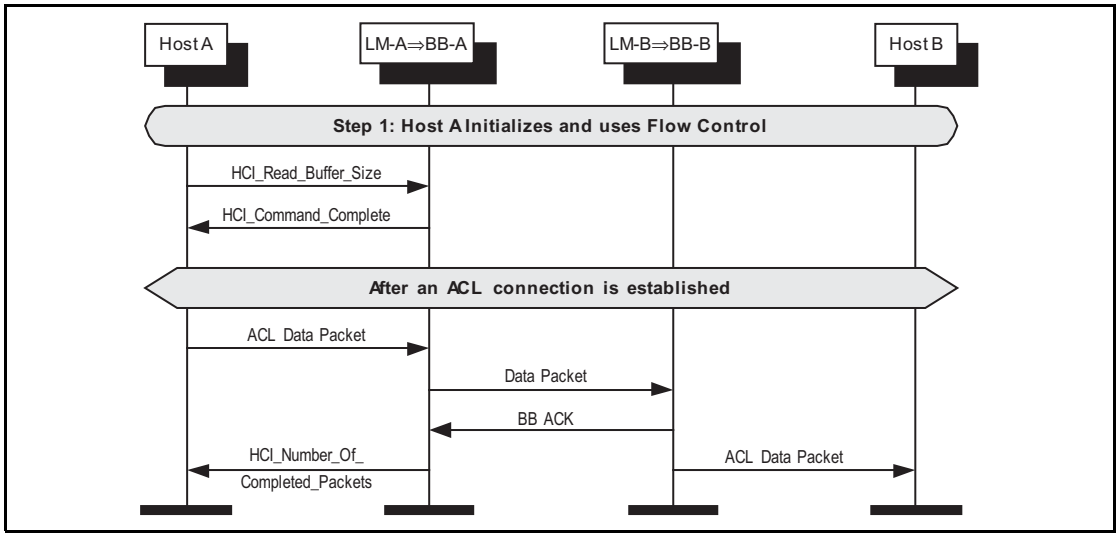


Figure 7.1: Host to Controller flow control.

Step 2: During initialization, the host notifies the Controller that host flow control shall be used, and then the host buffer sizes available. When a data packet has been received from a remote device, an HCI Data Packet is sent to the host from the Controller, and the host shall acknowledge its receipt by sending HCI_Host_Number_Of_Completed_Packets. (See Figure 7.2 on page 662)

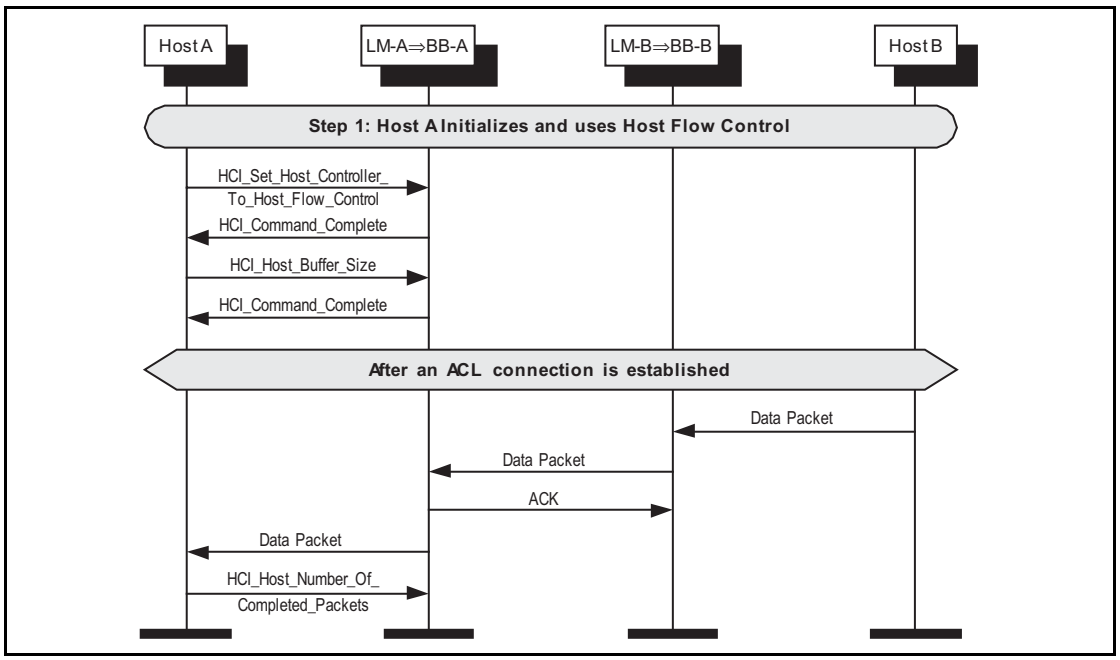


Figure 7.2: Controller to Host flow control.

8 LOOPBACK MODE

The loopback modes are used for testing of a device only.

8.1 LOCAL LOOPBACK MODE

The local loopback mode is used to loopback received HCI Commands, and HCI ACL and HCI Synchronous packets sent from the Host to the Controller.

Step 1: The host enters local loopback mode. Four connection complete events are generated and then a command complete event.
(See Figure 8.1 on page 663)

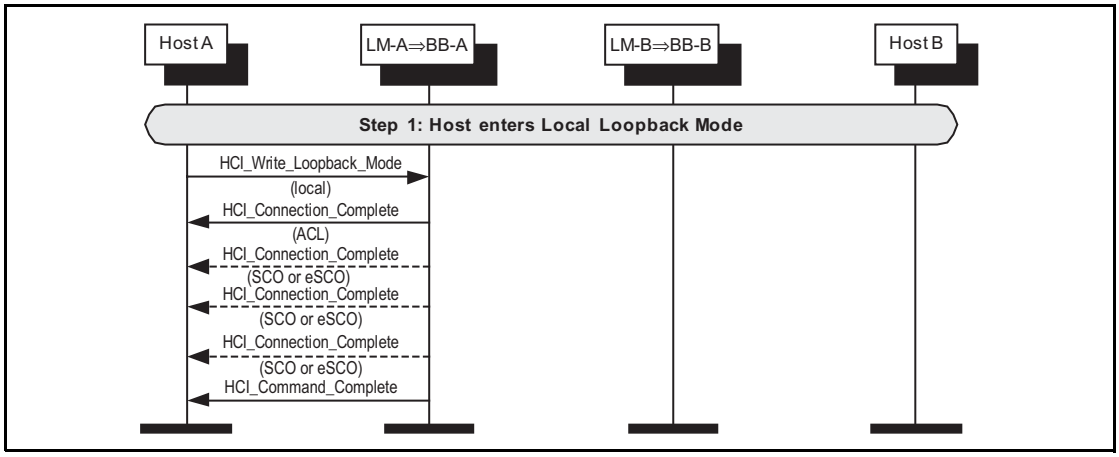


Figure 8.1: Entering local loopback mode.

Step 2a: The host sending HCI Data Packet will receive the exact same data back in HCI Data Packets from the Controller. (See Figure 8.2 on page 663)

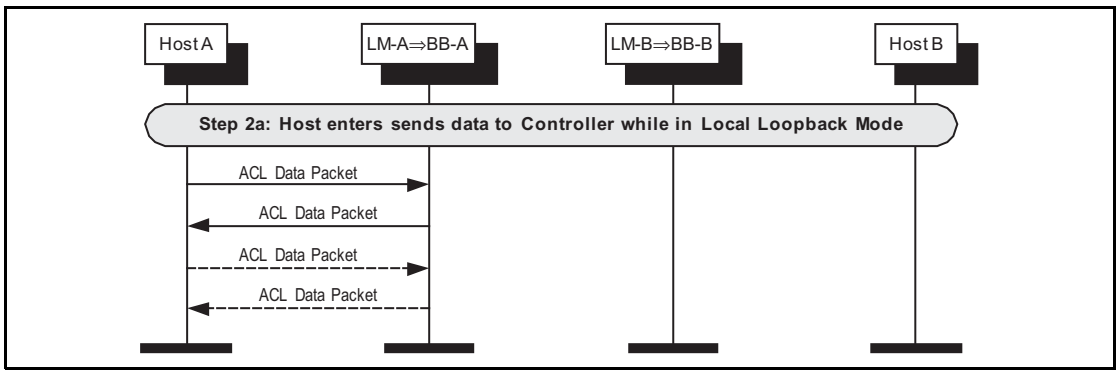


Figure 8.2: Looping back data in local loopback mode.

Step 2b: The host sending most HCI Command Packets to the Controller will receive an HCI_Loopback_Command event with the contents of the HCI Command Packet in the payload. (See Figure 8.3 on page 664)

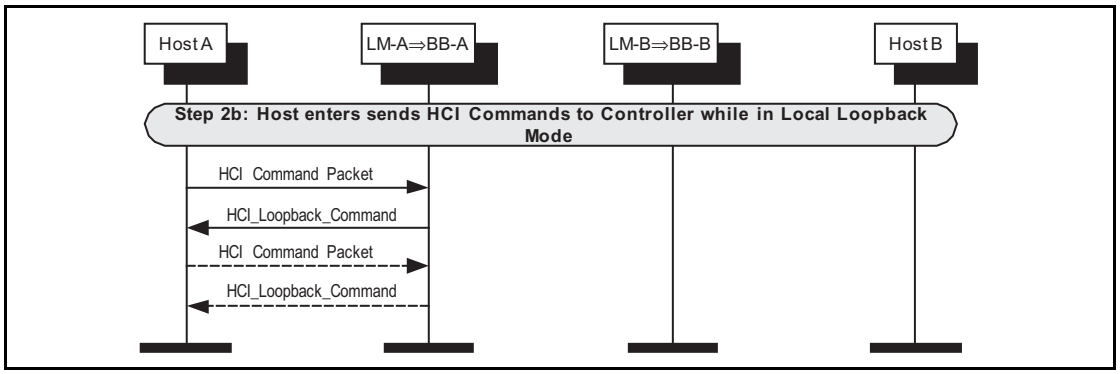


Figure 8.3: Looping back commands in local loopback mode.

Step 3: The host exits local loopback mode. Multiple disconnection complete events are generated before the command complete event.
(See Figure 8.4 on page 664)

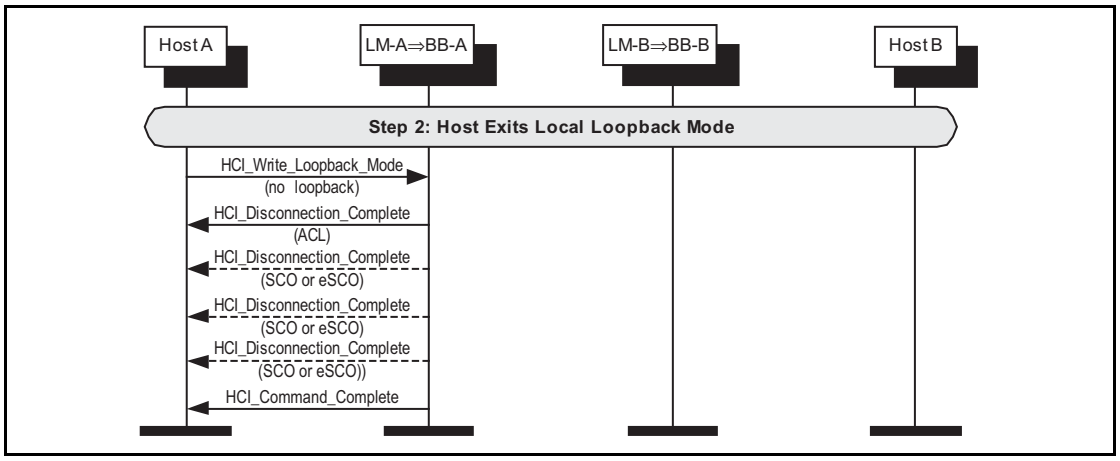


Figure 8.4: Exiting local loopback mode.

8.2 REMOTE LOOPBACK MODE

The remote loopback mode is used to lookback data to a remote device over the air.

Step 1: The remote host first sets up an connection to the local device. The local device then enables remote loopback. (See Figure 8.5 on page 665)

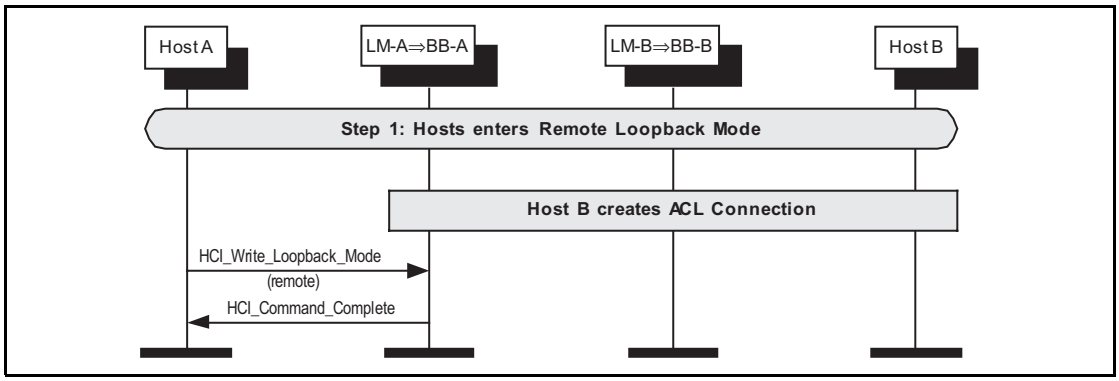


Figure 8.5: Entering remote loopback mode.

Step 2: Any data received from the remote host will be loopbacked in the Controller of the local device. (See Figure 8.6 on page 665)

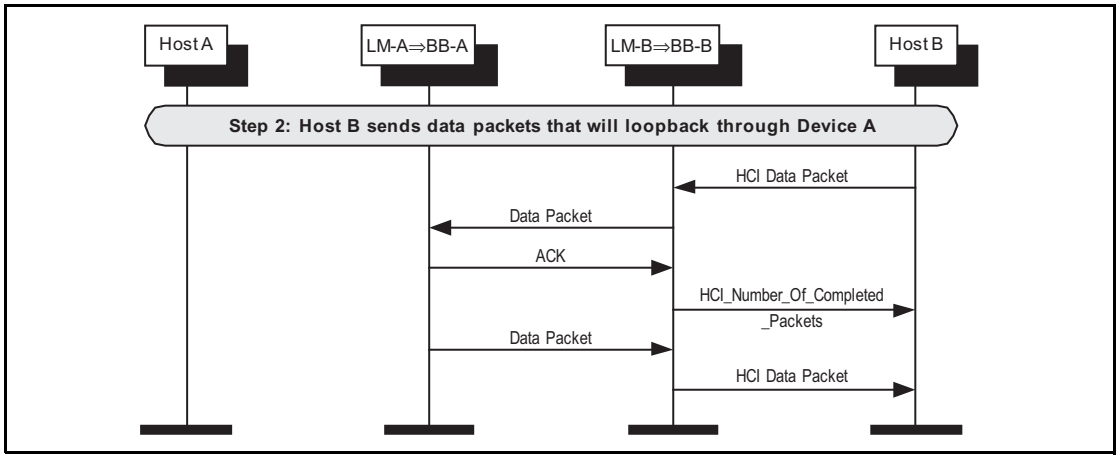


Figure 8.6: Looping back data in Remote Loopback Mode.

Step 3: The local host exits remote loopback mode. Any connections can then be disconnected by the remote device. (See Figure 8.7 on page 666)

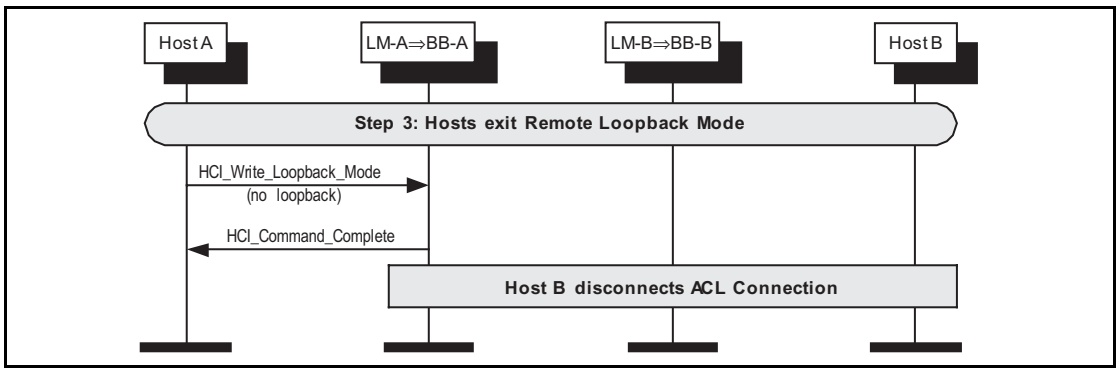


Figure 8.7: Exiting remote loopback mode.

9 LIST OF FIGURES

SAMPLE DATA

This appendix contains sample data for various parts of the Bluetooth baseband specification. All sample data are provided for reference purpose only; they are intended as a complement to the definitions provided elsewhere in the specification. They can be used to check the behavior of an implementation and avoid misunderstandings. Fulfilling these sample data is a necessary but not sufficient condition for an implementation to be fully Bluetooth compliant.

CONTENTS

1 ENCRYPTION SAMPLE DATA

This section contains four sets of sample data for the encryption process.

With respect to the functional description of the encryption engine in the Bluetooth baseband specification, the contents of registers and resulting concurrent values are listed as well. This by no means excludes different implementations (as far as they produce the same encryption stream) but is intended to describe the functional behavior.

In case of misunderstandings or inconsistencies, these sample data form the normative reference.

1.1 GENERATING KC' FROM KC,

where Kc'(x) = g2(x)(Kc(x) mod g1(x)).

```
Note: All polynomials are in hexadecimal notation.
      'L' is the effective key length in bytes.
      The notation 'p: [m]' implies that deg(p(x)) = m.

                              MSB                                        LSB
L = 1
g1:                [8]        00000000 00000000 00000000 0000011d
g2:                [119]      00e275a0 abd218d4 cf928b9b bf6cb08f
Kc:                           a2b230a4 93f281bb 61a85b82 a9d4a30e
Kc mod g1:         [7]        00000000 00000000 00000000 0000009f
g2(Kc mod g1):     [126]      7aa16f39 59836ba3 22049a7b 87f1d8a5
-------------------------------------------------------------
L = 2
g1:                [16]       00000000 00000000 00000000 0001003f
g2:                [112]      0001e3f6 3d7659b3 7f18c258 cff6efef
Kc:                           64e7df78 bb7ccaa4 61433123 5b3222ad
Kc mod g1:         [12]       00000000 00000000 00000000 00001ff0
g2(Kc mod g1):     [124]      142057bb 0bceac4c 58bd142e 1e710a50
-------------------------------------------------------------
L = 3
g1:                [24]       00000000 00000000 00000000 010000db
g2:                [104]      000001be f66c6c3a b1030a5a 1919808b
Kc:                           575e5156 ba685dc6 112124ac edb2c179
Kc mod g1:         [23]       00000000 00000000 00000000 008ddbc8
g2(Kc mod g1):     [127]      d56d0adb 8216cb39 7fe3c591 1ff95618
-------------------------------------------------------------
L = 4
g1:                [32]       00000000 00000000 00000001 000000af
g2:                [96]       00000001 6ab89969 de17467f d3736ad9
Kc:                           8917b4fc 403b6db2 1596b86d 1cb8adab
Kc mod g1:         [31]       00000000 00000000 00000000 aa1e78aa
g2(Kc mod g1):     [127]      91910128 b0e2f5ed a132a03e af3d8cda
-------------------------------------------------------------
```

Sample Data

```
L = 5
g1:                [40]        00000000 00000000 00000100 00000039
g2:                [88]        00000000 01630632 91da50ec 55715247
Kc:                            785c915b dd25b9c6 0102ab00 b6cd2a68
Kc mod g1:         [38]        00000000 00000000 0000007f 13d44436
g2(Kc mod g1):     [126]       6fb5651c cb80c8d7 ea1ee56d f1ec5d02
------------------------------------------------------------
L = 6
g1:                [48]        00000000 00000000 00010000 00000291
g2:                [77]        00000000 00002c93 52aa6cc0 54468311
Kc:                            5e77d19f 55ccd7d5 798f9a32 3b83e5d8
Kc mod g1:         [47]        00000000 00000000 000082eb 4af213ed
g2(Kc mod g1):     [124]       16096bcb afcf8def 1d226a1b 4d3f9a3d
------------------------------------------------------------
```

Sample Data

```
L = 7
g1:                [56]        00000000 00000000 01000000 00000095
g2:                [71]        00000000 000000b3 f7fffce2 79f3a073
Kc:                            05454e03 8ddcfbe3 ed024b2d 92b7f54c
Kc mod g1:         [55]        00000000 00000000 0095b8a4 8eb816da
g2(Kc mod g1):     [126]       50f9c0d4 e3178da9 4a09fe0d 34f67b0e
------------------------------------------------------------
L = 8
g1:                [64]        00000000 00000001 00000000 0000001b
g2:                [63]        00000000 00000000 a1ab815b c7ec8025
Kc:                            7ce149fc f4b38ad7 2a5d8a41 eb15ba31
Kc mod g1:         [63]        00000000 00000000 8660806c 1865deec
g2(Kc mod g1):     [126]       532c36d4 5d0954e0 922989b6 826f78dc
------------------------------------------------------------
L = 9
g1:                [72]        00000000 00000100 00000000 00000609
g2:                [49]        00000000 00000000 0002c980 11d8b04d
Kc:                            5eeff7ca 84fc2782 9c051726 3df6f36e
Kc mod g1:         [71]        00000000 00000083 58ccb7d0 b95d3c71
g2(Kc mod g1):     [120]       016313f6 0d3771cf 7f8e4bb9 4aa6827d
------------------------------------------------------------
L = 10
g1:                [80]        00000000 00010000 00000000 00000215
g2:                [42]        00000000 00000000 0000058e 24f9a4bb
Kc:                            7b13846e 88beb4de 34e7160a fd44dc65
Kc mod g1:         [79]        00000000 0000b4de 34171767 f36981c3
g2(Kc mod g1):     [121]       023bc1ec 34a0029e f798dcfb 618ba58d
------------------------------------------------------------
L = 11
g1:                [88]        00000000 01000000 00000000 0000013b
g2:                [35]        00000000 00000000 0000000c a76024d7
Kc:                            bda6de6c 6e7d757e 8dfe2d49 9a181193
Kc mod g1:         [86]        00000000 007d757e 8dfe88aa 2fcee371
g2(Kc mod g1):     [121]       022e08a9 3aa51d8d 2f93fa78 85cc1f87
------------------------------------------------------------
L = 12
g1:                [96]        00000001 00000000 00000000 000000dd
g2:                [28]        00000000 00000000 00000000 1c9c26b9
Kc:                            e6483b1c 2cdb1040 9a658f97 c4efd90d
Kc mod g1:         [93]        00000000 2cdb1040 9a658fd7 5b562e41
g2(Kc mod g1):     [121]       030d752b 216fe29b b880275c d7e6f6f9
------------------------------------------------------------
L = 13
g1:                [104]       00000100 00000000 00000000 0000049d
g2:                [21]        00000000 00000000 00000000 0026d9e3
Kc:                            d79d281d a2266847 6b223c46 dc0ab9ee
Kc mod g1:         [100]       0000001d a2266847 6b223c45 e1fc5fa6
g2(Kc mod g1):     [121]       03f11138 9cebf919 00b93808 4ac158aa
------------------------------------------------------------
```

Sample Data

```
L = 14
g1:                 [112]       00010000 00000000 00000000 0000014f
g2:                 [14]        00000000 00000000 00000000 00004377
Kc:                             cad9a65b 9fca1c1d a2320fcf 7c4ae48e
Kc mod g1:          [111]       0000a65b 9fca1c1d a2320fcf 7cb6a909
g2(Kc mod g1):      [125]       284840fd f1305f3c 529f5703 76adf7cf
------------------------------------------------------------
L = 15
g1:                 [120]       01000000 00000000 00000000 000000e7
g2:                 [7]         00000000 00000000 00000000 00000089
Kc:                             21f0cc31 049b7163 d375e9e1 06029809
Kc mod g1:          [119]       00f0cc31 049b7163 d375e9e1 0602840e
g2(Kc mod g1):      [126]       7f10b53b 6df84b94 f22e566a 3754a37e
------------------------------------------------------------
L = 16
g1:                 [128]       00000001 00000000 00000000 00000000 00000000
g2:                 [0]         00000000 00000000 00000000 00000001
Kc:                             35ec8fc3 d50ccd32 5f2fd907 bde206de
Kc mod g1:          [125]       35ec8fc3 d50ccd32 5f2fd907 bde206de
g2(Kc mod g1):      [125]       35ec8fc3 d50ccd32 5f2fd907 bde206de
------------------------------------------------------------
```

1.2 FIRST SET OF SAMPLE DATA

```
Initial values for the key, pan address and clock

K'c1[0]   = 00   K'c1[1]   = 00    K'c1[2]   = 00    K'c1[3]  = 00
K'c1[4]   = 00   K'c1[5]   = 00    K'c1[6]   = 00    K'c1[7]  = 00
K'c1[8]   = 00   K'c1[9]   = 00    K'c1[10]  = 00    K'c1[11] = 00
K'c1[12]  = 00   K'c1[13]  = 00    K'c1[14]  = 00    K'c1[15] = 00

Addr1[0] = 00   Addr1[1] = 00    Addr1[2] = 00
Addr1[3] = 00   Addr1[4] = 00    Addr1[5] = 00

CL1[0]  = 00   CL1[1]  = 00    CL1[2]  = 00    CL1[3]  = 00


=============================================================================================
         Fill LFSRs with initial data
=============================================================================================

  t  clk#    LFSR1    LFSR2     LFSR3      LFSR4           X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

  0     0    0000000* 00000000* 000000000* 0000000000*     0  0  0  0    0  00     00   00
  1     1    0000000* 00000001* 000000000* 0000000001*     0  0  0  0    0  00     00   00
  2     2    0000000* 00000002* 000000000* 0000000003*     0  0  0  0    0  00     00   00
  3     3    0000000* 00000004* 000000000* 0000000007*     0  0  0  0    0  00     00   00
  4     4    0000000* 00000008* 000000000* 000000000E*     0  0  0  0    0  00     00   00
  5     5    0000000* 00000010* 000000000* 000000001C*     0  0  0  0    0  00     00   00
  6     6    0000000* 00000020* 000000000* 0000000038*     0  0  0  0    0  00     00   00
```

Sample Data

7	7	0000000*	00000040*	000000000*	0000000070*	0	0	0	0	0	00	00	00
8	8	0000000*	00000080*	000000000*	00000000E0*	0	0	0	0	0	00	00	00
9	9	0000000*	00000100*	000000000*	00000001C0*	0	0	0	0	0	00	00	00
10	10	0000000*	00000200*	000000000*	0000000380*	0	0	0	0	0	00	00	00
11	11	0000000*	00000400*	000000000*	0000000700*	0	0	0	0	0	00	00	00
12	12	0000000*	00000800*	000000000*	0000000E00*	0	0	0	0	0	00	00	00
13	13	0000000*	00001000*	000000000*	0000001C00*	0	0	0	0	0	00	00	00
14	14	0000000*	00002000*	000000000*	0000003800*	0	0	0	0	0	00	00	00
15	15	0000000*	00004000*	000000000*	0000007000*	0	0	0	0	0	00	00	00
16	16	0000000*	00008000*	000000000*	000000E000*	0	0	0	0	0	00	00	00
17	17	0000000*	00010000*	000000000*	000001C000*	0	0	0	0	0	00	00	00
18	18	0000000*	00020000*	000000000*	0000038000*	0	0	0	0	0	00	00	00
19	19	0000000*	00040000*	000000000*	0000070000*	0	0	0	0	0	00	00	00
20	20	0000000*	00080000*	000000000*	00000E0000*	0	0	0	0	0	00	00	00
21	21	0000000*	00100000*	000000000*	00001C0000*	0	0	0	0	0	00	00	00
22	22	0000000*	00200000*	000000000*	0000380000*	0	0	0	0	0	00	00	00
23	23	0000000*	00400000*	000000000*	0000700000*	0	0	0	0	0	00	00	00
24	24	0000000*	00800000*	000000000*	0000E00000*	0	1	0	0	1	00	00	00
25	25	0000000*	01000000*	000000000*	0001C00000*	0	0	0	0	0	00	00	00
26	26	0000000	02000000*	000000000*	0003800000*	0	0	0	0	0	00	00	00
27	27	0000000	04000000*	000000000*	0007000000*	0	0	0	0	0	00	00	00
28	28	0000000	08000000*	000000000*	000E000000*	0	0	0	0	0	00	00	00
29	29	0000000	10000000*	000000000*	001C000000*	0	0	0	0	0	00	00	00
30	30	0000000	20000000*	000000000*	0038000000*	0	0	0	0	0	00	00	00
31	31	0000000	40000000*	000000000*	0070000000*	0	0	0	0	0	00	00	00
32	32	0000000	00000001	000000000*	00E0000000*	0	0	0	1	1	00	00	00
33	33	0000000	00000002	000000000*	01C0000000*	0	0	0	1	1	00	00	00
34	34	0000000	00000004	000000000	0380000000*	0	0	0	1	1	00	00	00
35	35	0000000	00000008	000000000	0700000000*	0	0	0	0	0	00	00	00
36	36	0000000	00000010	000000000	0E00000000*	0	0	0	0	0	00	00	00
37	37	0000000	00000020	000000000	1C00000000*	0	0	0	0	0	00	00	00
38	38	0000000	00000040	000000000	3800000000*	0	0	0	0	0	00	00	00
39	39	0000000	00000080	000000000	7000000000*	0	0	0	0	0	00	00	00

Start clocking Summation Combiner

40	1	0000000	00000100	000000000	6000000001	0	0	0	0	0	00	00	00
41	2	0000000	00000200	000000000	4000000003	0	0	0	0	0	00	00	00
42	3	0000000	00000400	000000000	0000000007	0	0	0	0	0	00	00	00
43	4	0000000	00000800	000000000	000000000E	0	0	0	0	0	00	00	00
44	5	0000000	00001001	000000000	000000001D	0	0	0	0	0	00	00	00
45	6	0000000	00002002	000000000	000000003B	0	0	0	0	0	00	00	00
46	7	0000000	00004004	000000000	0000000077	0	0	0	0	0	00	00	00
47	8	0000000	00008008	000000000	00000000EE	0	0	0	0	0	00	00	00
48	9	0000000	00010011	000000000	00000001DD	0	0	0	0	0	00	00	00
49	10	0000000	00020022	000000000	00000003BB	0	0	0	0	0	00	00	00
50	11	0000000	00040044	000000000	0000000777	0	0	0	0	0	00	00	00
51	12	0000000	00080088	000000000	0000000EEE	0	0	0	0	0	00	00	00
52	13	0000000	00100110	000000000	0000001DDD	0	0	0	0	0	00	00	00
53	14	0000000	00200220	000000000	0000003BBB	0	0	0	0	0	00	00	00
54	15	0000000	00400440	000000000	0000007777	0	0	0	0	0	00	00	00
55	16	0000000	00800880	000000000	000000EEEE	0	1	0	0	1	00	00	00
56	17	0000000	01001100	000000000	000001DDDD	0	0	0	0	0	00	00	00
57	18	0000000	02002200	000000000	000003BBBB	0	0	0	0	0	00	00	00
58	19	0000000	04004400	000000000	0000077777	0	0	0	0	0	00	00	00
59	20	0000000	08008800	000000000	00000EEEEE	0	0	0	0	0	00	00	00
60	21	0000000	10011000	000000000	00001DDDDD	0	0	0	0	0	00	00	00

Sample Data

61	22	0000000	20022000	000000000	00003BBBBB	0	0	0	0	0	00	00	00
62	23	0000000	40044000	000000000	0000777777	0	0	0	0	0	00	00	00
63	24	0000000	00088001	000000000	0000EEEEEE	0	0	0	0	0	00	00	00
64	25	0000000	00110003	000000000	0001DDDDDD	0	0	0	0	0	00	00	00
65	26	0000000	00220006	000000000	0003BBBBBB	0	0	0	0	0	00	00	00
66	27	0000000	0044000C	000000000	0007777777	0	0	0	0	0	00	00	00
67	28	0000000	00880018	000000000	000EEEEEEE	0	1	0	0	1	00	00	00
68	29	0000000	01100031	000000000	001DDDDDDC	0	0	0	0	0	00	00	00
69	30	0000000	02200062	000000000	003BBBBBB8	0	0	0	0	0	00	00	00
70	31	0000000	044000C4	000000000	0077777770	0	0	0	0	0	00	00	00
71	32	0000000	08800188	000000000	00EEEEEEE0	0	1	0	1	0	01	00	00
72	33	0000000	11000311	000000000	01DDDDDDC1	0	0	0	1	0	00	01	00
73	34	0000000	22000622	000000000	03BBBBBB83	0	0	0	1	1	11	00	01
74	35	0000000	44000C44	000000000	0777777707	0	0	0	0	1	10	11	00
75	36	0000000	08001888	000000000	0EEEEEEE0E	0	0	0	1	1	01	10	11
76	37	0000000	10003111	000000000	1DDDDDDC1D	0	0	0	1	0	01	01	10
77	38	0000000	20006222	000000000	3BBBBBB83B	0	0	0	1	0	11	01	01
78	39	0000000	4000C444	000000000	7777777077	0	0	0	0	1	01	11	01
79	40	0000000	00018888	000000000	6EEEEEE0EF	0	0	0	1	0	10	01	11
80	41	0000000	00031110	000000000	5DDDDDC1DE	0	0	0	1	1	00	10	01
81	42	0000000	00062220	000000000	3BBBBB83BC	0	0	0	1	1	01	00	10
82	43	0000000	000C4440	000000000	7777770779	0	0	0	0	1	01	01	00
83	44	0000000	00188880	000000000	6EEEEE0EF2	0	0	0	1	0	11	01	01
84	45	0000000	00311100	000000000	5DDDDC1DE5	0	0	0	1	0	10	11	01
85	46	0000000	00622200	000000000	3BBBB83BCB	0	0	0	1	1	01	10	11
86	47	0000000	00C44400	000000000	7777707797	0	1	0	0	0	01	01	10
87	48	0000000	01888801	000000000	6EEEE0EF2F	0	1	0	1	1	11	01	01
88	49	0000000	03111003	000000000	5DDDC1DE5E	0	0	0	1	0	10	11	01
89	50	0000000	06222006	000000000	3BBB83BCBC	0	0	0	1	1	01	10	11
90	51	0000000	0C44400C	000000000	7777077979	0	0	0	0	1	00	01	10
91	52	0000000	18888018	000000000	6EEE0EF2F2	0	1	0	1	0	10	00	01
92	53	0000000	31110030	000000000	5DDC1DE5E5	0	0	0	1	1	11	10	00
93	54	0000000	62220060	000000000	3BB83BCBCB	0	0	0	1	0	00	11	10
94	55	0000000	444400C1	000000000	7770779797	0	0	0	0	0	10	00	11
95	56	0000000	08880183	000000000	6EE0EF2F2F	0	1	0	1	0	00	10	00
96	57	0000000	11100307	000000000	5DC1DE5E5F	0	0	0	1	1	01	00	10
97	58	0000000	2220060E	000000000	3B83BCBCBF	0	0	0	1	0	00	01	00
98	59	0000000	44400C1C	000000000	770779797E	0	0	0	0	0	11	00	01
99	60	0000000	08801838	000000000	6E0EF2F2FC	0	1	0	0	0	01	11	00
100	61	0000000	11003070	000000000	5C1DE5E5F8	0	0	0	0	1	11	01	11
101	62	0000000	220060E0	000000000	383BCBCBF0	0	0	0	0	1	01	11	01
102	63	0000000	4400C1C0	000000000	70779797E0	0	0	0	0	1	11	01	11
103	64	0000000	08018380	000000000	60EF2F2FC1	0	0	0	1	0	10	11	01
104	65	0000000	10030701	000000000	41DE5E5F82	0	0	0	1	1	01	10	11
105	66	0000000	20060E02	000000000	03BCBCBF04	0	0	0	1	0	01	01	10
106	67	0000000	400C1C05	000000000	0779797E09	0	0	0	0	1	10	01	01
107	68	0000000	0018380A	000000000	0EF2F2FC12	0	0	0	1	1	00	10	01
108	69	0000000	00307015	000000000	1DE5E5F825	0	0	0	1	1	01	00	10
109	70	0000000	0060E02A	000000000	3BCBCBF04B	0	0	0	1	0	00	01	00
110	71	0000000	00C1C055	000000000	779797E097	0	1	0	1	0	10	00	01
111	72	0000000	018380AA	000000000	6F2F2FC12F	0	1	0	0	1	11	10	00
112	73	0000000	03070154	000000000	5E5E5F825E	0	0	0	0	1	11	11	10
113	74	0000000	060E02A8	000000000	3CBCBF04BC	0	0	0	1	0	11	11	11
114	75	0000000	0C1C0550	000000000	79797E0979	0	0	0	0	1	00	11	11
115	76	0000000	18380AA0	000000000	72F2FC12F2	0	0	0	1	1	10	00	11
116	77	0000000	30701541	000000000	65E5F825E5	0	0	0	1	1	11	10	00
117	78	0000000	60E02A82	000000000	4BCBF04BCB	0	1	0	1	1	00	11	10

Sample Data

118	79	0000000	41C05505	000000000	1797E09796	0	1	0	1	0	11	00	11
119	80	0000000	0380AA0A	000000000	2F2FC12F2C	0	1	0	0	0	01	11	00
120	81	0000000	07015415	000000000	5E5F825E59	0	0	0	0	1	11	01	11
121	82	0000000	0E02A82A	000000000	3CBF04BCB2	0	0	0	1	0	10	11	01
122	83	0000000	1C055054	000000000	797E097964	0	0	0	0	0	01	10	11
123	84	0000000	380AA0A8	000000000	72FC12F2C9	0	0	0	1	0	01	01	10
124	85	0000000	70154151	000000000	65F825E593	0	0	0	1	0	11	01	01
125	86	0000000	602A82A3	000000000	4BF04BCB26	0	0	0	1	0	10	11	01
126	87	0000000	40550546	000000000	17E097964C	0	0	0	1	1	01	10	11
127	88	0000000	00AA0A8D	000000000	2FC12F2C99	0	1	0	1	1	01	01	10
128	89	0000000	0154151A	000000000	5F825E5932	0	0	0	1	0	11	01	01
129	90	0000000	02A82A34	000000000	3F04BCB264	0	1	0	0	0	10	11	01
130	91	0000000	05505468	000000000	7E097964C9	0	0	0	0	0	01	10	11
131	92	0000000	0AA0A8D0	000000000	7C12F2C992	0	1	0	0	0	01	01	10
132	93	0000000	154151A1	000000000	7825E59324	0	0	0	0	1	10	01	01
133	94	0000000	2A82A342	000000000	704BCB2648	0	1	0	0	1	00	10	01
134	95	0000000	55054684	000000000	6097964C91	0	0	0	1	1	01	00	10
135	96	0000000	2A0A8D09	000000000	412F2C9923	0	0	0	0	1	01	01	00
136	97	0000000	54151A12	000000000	025E593246	0	0	0	0	1	10	01	01
137	98	0000000	282A3424	000000000	04BCB2648D	0	0	0	1	1	00	10	01
138	99	0000000	50546848	000000000	097964C91A	0	0	0	0	0	01	00	10
139	100	0000000	20A8D090	000000000	12F2C99235	0	1	0	1	1	00	01	00
140	101	0000000	4151A120	000000000	25E593246A	0	0	0	1	1	11	00	01
141	102	0000000	02A34240	000000000	4BCB2648D5	0	1	0	1	1	01	11	00
142	103	0000000	05468481	000000000	17964C91AB	0	0	0	1	0	10	01	11
143	104	0000000	0A8D0903	000000000	2F2C992357	0	1	0	0	1	00	10	01
144	105	0000000	151A1206	000000000	5E593246AE	0	0	0	0	0	01	00	10
145	106	0000000	2A34240C	000000000	3CB2648D5C	0	0	0	1	0	00	01	00
146	107	0000000	54684818	000000000	7964C91AB8	0	0	0	0	0	11	00	01
147	108	0000000	28D09030	000000000	72C9923571	0	1	0	1	1	01	11	00
148	109	0000000	51A12060	000000000	6593246AE2	0	1	0	1	1	10	01	11
149	110	0000000	234240C0	000000000	4B2648D5C5	0	0	0	0	0	00	10	01
150	111	0000000	46848180	000000000	164C91AB8A	0	1	0	0	1	01	00	10
151	112	0000000	0D090301	000000000	2C99235714	0	0	0	1	0	00	01	00
152	113	0000000	1A120602	000000000	593246AE28	0	0	0	0	0	11	00	01
153	114	0000000	34240C04	000000000	32648D5C51	0	0	0	0	1	10	11	00
154	115	0000000	68481809	000000000	64C91AB8A2	0	0	0	1	1	01	10	11
155	116	0000000	50903012	000000000	4992357144	0	1	0	1	1	01	01	10
156	117	0000000	21206024	000000000	13246AE288	0	0	0	0	1	10	01	01
157	118	0000000	4240C048	000000000	2648D5C511	0	0	0	0	0	00	10	01
158	119	0000000	04818090	000000000	4C91AB8A23	0	1	0	1	0	00	00	10
159	120	0000000	09030120	000000000	1923571446	0	0	0	0	0	00	00	00
160	121	0000000	12060240	000000000	3246AE288D	0	0	0	0	0	00	00	00
161	122	0000000	240C0480	000000000	648D5C511B	0	0	0	1	1	00	00	00
162	123	0000000	48180900	000000000	491AB8A237	0	0	0	0	0	00	00	00
163	124	0000000	10301200	000000000	123571446F	0	0	0	0	0	00	00	00
164	125	0000000	20602400	000000000	246AE288DF	0	0	0	0	0	00	00	00
165	126	0000000	40C04800	000000000	48D5C511BE	0	1	0	1	0	01	00	00
166	127	0000000	01809001	000000000	11AB8A237D	0	1	0	1	1	00	01	00
167	128	0000000	03012002	000000000	23571446FA	0	0	0	0	0	11	00	01
168	129	0000000	06024004	000000000	46AE288DF5	0	0	0	1	0	01	11	00
169	130	0000000	0C048008	000000000	0D5C511BEA	0	0	0	0	1	11	01	11
170	131	0000000	18090011	000000000	1AB8A237D5	0	0	0	1	0	10	11	01
171	132	0000000	30120022	000000000	3571446FAA	0	0	0	0	0	01	10	11
172	133	0000000	60240044	000000000	6AE288DF55	0	0	0	1	0	01	01	10
173	134	0000000	40480089	000000000	55C511BEAA	0	0	0	1	0	11	01	01
174	135	0000000	00900113	000000000	2B8A237D54	0	1	0	1	1	10	11	01

Sample Data

175	136	0000000	01200227	000000000	571446FAA8	0	0	0	0	0	01	10	11
176	137	0000000	0240044E	000000000	2E288DF550	0	0	0	0	1	00	01	10
177	138	0000000	0480089C	000000000	5C511BEAA0	0	1	0	0	1	11	00	01
178	139	0000000	09001138	000000000	38A237D540	0	0	0	1	0	01	11	00
179	140	0000000	12002270	000000000	71446FAA81	0	0	0	0	1	11	01	11
180	141	0000000	240044E0	000000000	6288DF5503	0	0	0	1	0	10	11	01
181	142	0000000	480089C0	000000000	4511BEAA06	0	0	0	0	0	01	10	11
182	143	0000000	10011381	000000000	0A237D540D	0	0	0	0	1	00	01	10
183	144	0000000	20022702	000000000	1446FAA81A	0	0	0	0	0	11	00	01
184	145	0000000	40044E04	000000000	288DF55035	0	0	0	1	0	01	11	00
185	146	0000000	00089C08	000000000	511BEAA06A	0	0	0	0	1	11	01	11
186	147	0000000	00113810	000000000	2237D540D5	0	0	0	0	1	01	11	01
187	148	0000000	00227021	000000000	446FAA81AA	0	0	0	0	1	11	01	11
188	149	0000000	0044E042	000000000	08DF550355	0	0	0	1	0	10	11	01
189	150	0000000	0089C085	000000000	11BEAA06AA	0	1	0	1	0	10	10	11
190	151	0000000	0113810A	000000000	237D540D54	0	0	0	0	0	10	10	10
191	152	0000000	02270215	000000000	46FAA81AA9	0	0	0	1	1	10	10	10
192	153	0000000	044E042A	000000000	0DF5503553	0	0	0	1	1	10	10	10
193	154	0000000	089C0854	000000000	1BEAA06AA7	0	1	0	1	0	01	10	10
194	155	0000000	113810A8	000000000	37D540D54E	0	0	0	1	0	01	01	10
195	156	0000000	22702150	000000000	6FAA81AA9D	0	0	0	1	0	11	01	01
196	157	0000000	44E042A0	000000000	5F5503553A	0	1	0	0	0	10	11	01
197	158	0000000	09C08540	000000000	3EAA06AA75	0	1	0	1	0	10	10	11
198	159	0000000	13810A80	000000000	7D540D54EA	0	1	0	0	1	10	10	10
199	160	0000000	27021500	000000000	7AA81AA9D5	0	0	0	1	1	10	10	10
200	161	0000000	4E042A00	000000000	75503553AB	0	0	0	0	0	10	10	10
201	162	0000000	1C085400	000000000	6AA06AA756	0	0	0	1	1	10	10	10
202	163	0000000	3810A800	000000000	5540D54EAC	0	0	0	0	0	10	10	10
203	164	0000000	70215000	000000000	2A81AA9D58	0	0	0	1	1	10	10	10
204	165	0000000	6042A001	000000000	5503553AB0	0	0	0	0	0	10	10	10
205	166	0000000	40854002	000000000	2A06AA7561	0	1	0	0	1	10	10	10
206	167	0000000	010A8004	000000000	540D54EAC3	0	0	0	0	0	10	10	10
207	168	0000000	02150009	000000000	281AA9D586	0	0	0	0	0	10	10	10
208	169	0000000	042A0012	000000000	503553AB0C	0	0	0	0	0	10	10	10
209	170	0000000	08540024	000000000	206AA75618	0	0	0	0	0	10	10	10
210	171	0000000	10A80048	000000000	40D54EAC30	0	1	0	1	0	01	10	10
211	172	0000000	21500091	000000000	01AA9D5861	0	0	0	1	0	01	01	10
212	173	0000000	42A00122	000000000	03553AB0C3	0	1	0	0	0	11	01	01
213	174	0000000	05400244	000000000	06AA756186	0	0	0	1	0	10	11	01
214	175	0000000	0A800488	000000000	0D54EAC30D	0	1	0	0	1	01	10	11
215	176	0000000	15000911	000000000	1AA9D5861A	0	0	0	1	0	01	01	10
216	177	0000000	2A001223	000000000	3553AB0C35	0	0	0	0	1	10	01	01
217	178	0000000	54002446	000000000	6AA756186A	0	0	0	1	1	00	10	01
218	179	0000000	2800488D	000000000	554EAC30D5	0	0	0	0	0	01	00	10
219	180	0000000	5000911B	000000000	2A9D5861AA	0	0	0	1	0	00	01	00
220	181	0000000	20012236	000000000	553AB0C355	0	0	0	0	0	11	00	01
221	182	0000000	4002446C	000000000	2A756186AA	0	0	0	0	1	10	11	00
222	183	0000000	000488D9	000000000	54EAC30D54	0	0	0	1	1	01	10	11
223	184	0000000	000911B2	000000000	29D5861AA8	0	0	0	1	0	01	01	10
224	185	0000000	00122364	000000000	53AB0C3550	0	0	0	1	0	11	01	01
225	186	0000000	002446C8	000000000	2756186AA0	0	0	0	0	1	01	11	01
226	187	0000000	00488D90	000000000	4EAC30D540	0	0	0	1	0	10	01	11
227	188	0000000	00911B20	000000000	1D5861AA81	0	1	0	0	1	00	10	01
228	189	0000000	01223640	000000000	3AB0C35502	0	0	0	1	1	01	00	10
229	190	0000000	02446C80	000000000	756186AA05	0	0	0	0	1	01	01	00
230	191	0000000	0488D901	000000000	6AC30D540B	0	1	0	1	1	11	01	01
231	192	0000000	0911B203	000000000	55861AA817	0	0	0	1	0	10	11	01

Sample Data

```
232   193    0000000  12236407  000000000  2B0C35502F      0  0  0  0    0  01     10   11
233   194    0000000  2446C80E  000000000  56186AA05F      0  0  0  0    1  00     01   10
234   195    0000000  488D901C  000000000  2C30D540BF      0  1  0  0    1  11     00   01
235   196    0000000  111B2039  000000000  5861AA817E      0  0  0  0    1  10     11   00
236   197    0000000  22364072  000000000  30C35502FD      0  0  0  1    1  01     10   11
237   198    0000000  446C80E4  000000000  6186AA05FB      0  0  0  1    0  01     01   10
238   199    0000000  08D901C8  000000000  430D540BF6      0  1  0  0    0  11     01   01
239   200    0000000  11B20391  000000000  061AA817EC      0  1  0  0    0  10     11   01

Z[0]  = 3D
Z[1]  = C1
Z[2]  = F0
Z[3]  = BB
Z[4]  = 58
Z[5]  = 1E
Z[6]  = 42
Z[7]  = 42
Z[8]  = 4B
Z[9]  = 8E
Z[10] = C1
Z[11] = 2A
Z[12] = 40
Z[13] = 63
Z[14] = 7A
Z[15] = 1E

================================================================================================
         Reload this pattern into the LFSRs
         Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 04B583D
LFSR2  <= 208E1EC1
LFSR3  <= 063C142F0
LFSR4  <= 0F7A2A42BB
C[t+1] <= 10

================================================================================================
         Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240     1    04B583D  208E1EC1  063C142F0  0F7A2A42BB      0  1  0  0    0  10     11   01
241     2    096B07A  411C3D82  0C78285E1  1EF4548577      1  0  1  1    1  10     10   11
242     3    12D60F4  02387B04  18F050BC3  3DE8A90AEF      0  0  1  1    0  01     10   10
243     4    05AC1E9  0470F609  11E0A1786  7BD15215DF      0  0  0  1    0  01     01   10
244     5    0B583D2  08E1EC13  03C142F0C  77A2A42BBF      1  1  0  1    0  00     01   01
245     6    16B07A5  11C3D827  078285E18  6F4548577E      0  1  0  0    1  11     00   01
246     7    0D60F4B  2387B04F  0F050BC30  5E8A90AEFD      1  1  1  1    1  00     11   00
247     8    1AC1E97  470F609E  1E0A17860  3D15215DFA      1  0  1  0    0  11     00   11
248     9    1583D2E  0E1EC13D  1C142F0C0  7A2A42BBF4      0  0  1  0    0  01     11   00
249    10    0B07A5D  1C3D827B  18285E181  74548577E9      1  0  1  0    1  10     01   11
250    11    160F4BB  387B04F7  1050BC302  68A90AEFD2      0  0  0  1    1  00     10   01
251    12    0C1E976  70F609EE  00A178605  515215DFA5      1  1  0  0    0  00     00   10
252    13    183D2ED  61EC13DD  0142F0C0B  22A42BBF4B      1  1  0  1    1  01     00   00
253    14    107A5DA  43D827BA  0285E1817  4548577E97      0  1  0  0    0  00     01   00
254    15    00F4BB4  07B04F74  050BC302F  0A90AEFD2E      0  1  0  1    0  10     00   01
255    16    01E9769  0F609EE8  0A178605E  15215DFA5C      0  0  1  0    1  11     10   00
256    17    03D2ED3  1EC13DD0  142F0C0BD  2A42BBF4B9      0  1  0  0    0  00     11   10
257    18    07A5DA7  3D827BA0  085E1817B  548577E972      0  1  1  1    1  11     00   11
```

Sample Data

258	19	0F4BB4F	7B04F740	10BC302F6	290AEFD2E5	1	0	0	0	0	01	11	00
259	20	1E9769F	7609EE80	0178605ED	5215DFA5CA	1	0	0	0	0	10	01	11
260	21	1D2ED3F	6C13DD01	02F0C0BDA	242BBF4B94	1	0	0	0	1	00	10	01
261	22	1A5DA7E	5827BA03	05E1817B4	48577E9729	1	0	0	0	1	01	00	10
262	23	14BB4FC	304F7407	0BC302F69	10AEFD2E53	0	0	1	1	1	00	01	00
263	24	09769F9	609EE80E	178605ED2	215DFA5CA7	1	1	0	0	0	10	00	01
264	25	12ED3F2	413DD01C	0F0C0BDA4	42BBF4B94F	0	0	1	1	0	00	10	00
265	26	05DA7E5	027BA038	1E1817B49	0577E9729F	0	0	1	0	1	01	00	10
266	27	0BB4FCA	04F74071	1C302F693	0AEFD2E53F	1	1	1	1	1	11	01	00
267	28	1769F95	09EE80E3	18605ED27	15DFA5CA7F	0	1	1	1	0	11	11	01
268	29	0ED3F2B	13DD01C6	10C0BDA4F	2BBF4B94FE	1	1	0	1	0	10	11	11
269	30	1DA7E56	27BA038D	01817B49F	577E9729FD	1	1	0	0	0	10	10	11
270	31	1B4FCAD	4F74071B	0302F693E	2EFD2E53FB	1	0	0	1	0	01	10	10
271	32	169F95B	1EE80E37	0605ED27D	5DFA5CA7F7	0	1	0	1	1	01	01	10
272	33	0D3F2B7	3DD01C6E	0C0BDA4FB	3BF4B94FEF	1	1	1	1	1	00	01	01
273	34	1A7E56F	7BA038DC	1817B49F6	77E9729FDE	1	1	1	1	0	01	00	01
274	35	14FCADF	774071B9	102F693ED	6FD2E53FBD	0	0	0	1	0	00	01	00
275	36	09F95BE	6E80E373	005ED27DB	5FA5CA7F7B	1	1	0	1	1	10	00	01
276	37	13F2B7C	5D01C6E7	00BDA4FB6	3F4B94FEF7	0	0	0	0	0	11	10	00
277	38	07E56F9	3A038DCE	017B49F6C	7E9729FDEE	0	0	0	1	0	00	11	10
278	39	0FCADF2	74071B9C	02F693ED8	7D2E53FBDD	1	0	0	0	1	10	00	11
279	40	1F95BE5	680E3738	05ED27DB0	7A5CA7F7BA	1	0	0	0	1	11	10	00
280	41	1F2B7CA	501C6E71	0BDA4FB60	74B94FEF74	1	0	1	1	0	01	11	10
281	42	1E56F94	2038DCE2	17B49F6C0	69729FDEE8	1	0	0	0	0	10	01	11
282	43	1CADF29	4071B9C4	0F693ED80	52E53FBDD1	1	0	1	1	1	11	10	01
283	44	195BE53	00E37389	1ED27DB01	25CA7F7BA3	1	1	1	1	1	01	11	10
284	45	12B7CA6	01C6E713	1DA4FB602	4B94FEF747	0	1	1	1	0	01	01	11
285	46	056F94C	038DCE26	1B49F6C04	1729FDEE8E	0	1	1	0	1	11	01	01
286	47	0ADF299	071B9C4D	1693ED808	2E53FBDD1C	1	0	0	0	0	10	11	01
287	48	15BE532	0E37389A	0D27DB011	5CA7F7BA38	0	0	1	1	0	10	10	11
288	49	0B7CA64	1C6E7135	1A4FB6022	394FEF7471	1	0	1	0	0	01	10	10
289	50	16F94C9	38DCE26A	149F6C044	729FDEE8E2	0	1	0	1	1	01	01	10
290	51	0DF2993	71B9C4D4	093ED8089	653FBDD1C4	1	1	1	0	0	00	01	01
291	52	1BE5327	637389A9	127DB0112	4A7F7BA388	1	0	0	0	1	11	00	01
292	53	17CA64E	46E71353	04FB60224	14FEF74710	0	1	0	1	1	01	11	00
293	54	0F94C9C	0DCE26A6	09F6C0448	29FDEE8E21	1	1	1	1	1	01	01	11
294	55	1F29939	1B9C4D4D	13ED80890	53FBDD1C42	1	1	0	1	0	00	01	01
295	56	1E53272	37389A9A	07DB01121	27F7BA3884	1	0	0	1	0	10	00	01
296	57	1CA64E5	6E713534	0FB602242	4FEF747108	1	0	1	1	1	00	10	00
297	58	194C9CB	5CE26A69	1F6C04485	1FDEE8E210	1	1	1	1	0	11	00	10
298	59	1299397	39C4D4D3	1ED80890A	3FBDD1C420	0	1	1	1	0	00	11	00
299	60	053272F	7389A9A6	1DB011214	7F7BA38840	0	1	1	0	0	11	00	11
300	61	0A64E5E	6713534C	1B6022428	7EF7471081	1	0	1	1	0	00	11	00
301	62	14C9CBD	4E26A699	16C044850	7DEE8E2102	0	0	0	1	1	10	00	11
302	63	099397A	1C4D4D32	0D80890A0	7BDD1C4205	1	0	1	1	1	00	10	00
303	64	13272F4	389A9A65	1B0112141	77BA38840B	0	1	1	1	1	00	00	10
304	65	064E5E8	713534CB	160224283	6F74710817	0	0	0	0	0	00	00	00
305	66	0C9CBD1	626A6997	0C0448507	5EE8E2102E	1	0	1	1	1	01	00	00
306	67	19397A3	44D4D32E	180890A0E	3DD1C4205C	1	1	1	1	1	11	01	00
307	68	1272F46	09A9A65D	10112141D	7BA38840B8	0	1	0	1	1	10	11	01
308	69	04E5E8C	13534CBA	00224283A	7747108171	0	0	0	0	0	01	10	11
309	70	09CBD19	26A69975	004485075	6E8E2102E3	1	1	0	1	0	10	01	10
310	71	1397A32	4D4D32EB	00890A0EA	5D1C4205C7	0	0	0	0	0	00	10	01
311	72	072F465	1A9A65D7	0112141D5	3A38840B8F	0	1	0	0	1	01	00	10
312	73	0E5E8CA	3534CBAF	0224283AA	747108171F	1	0	0	0	0	00	01	00
313	74	1CBD194	6A69975E	044850755	68E2102E3E	1	0	0	1	0	10	00	01
314	75	197A329	54D32EBC	0890A0EAB	51C4205C7D	1	1	1	1	0	01	10	00

Sample Data

315	76	12F4653	29A65D79	112141D56	238840B8FA	0	1	0	1	1	01	01	10
316	77	05E8CA6	534CBAF2	024283AAD	47108171F4	0	0	0	0	1	10	01	01
317	78	0BD194D	269975E5	04850755B	0E2102E3E9	1	1	0	0	0	11	10	01
318	79	17A329A	4D32EBCB	090A0EAB6	1C4205C7D2	0	0	1	0	0	00	11	10
319	80	0F46535	1A65D797	12141D56D	38840B8FA5	1	0	0	1	0	11	00	11
320	81	1E8CA6A	34CBAF2F	04283AADA	7108171F4B	1	1	0	0	1	01	11	00
321	82	1D194D5	69975E5F	0850755B4	62102E3E97	1	1	1	0	0	01	01	11
322	83	1A329AA	532EBCBF	10A0EAB68	44205C7D2F	1	0	0	0	0	11	01	01
323	84	1465355	265D797F	0141D56D1	0840B8FA5E	0	0	0	0	1	01	11	01
324	85	08CA6AB	4CBAF2FF	0283AADA2	108171F4BC	1	1	0	1	0	01	01	11
325	86	1194D56	1975E5FF	050755B45	2102E3E979	0	0	0	0	1	10	01	01
326	87	0329AAD	32EBCBFF	0A0EAB68A	4205C7D2F3	0	1	1	0	0	11	10	01
327	88	065355A	65D797FF	141D56D14	040B8FA5E7	0	1	0	0	0	00	11	10
328	89	0CA6AB4	4BAF2FFF	083AADA28	08171F4BCF	1	1	1	0	1	11	00	11
329	90	194D569	175E5FFF	10755B450	102E3E979E	1	0	0	0	0	01	11	00
330	91	129AAD3	2EBCBFFF	00EAB68A1	205C7D2F3C	0	1	0	0	0	10	01	11
331	92	05355A6	5D797FFF	01D56D142	40B8FA5E78	0	0	0	1	1	00	10	01
332	93	0A6AB4D	3AF2FFFE	03AADA285	0171F4BCF1	1	1	0	0	0	00	00	10
333	94	14D569B	75E5FFFD	0755B450A	02E3E979E2	0	1	0	1	0	01	00	00
334	95	09AAD37	6BCBFFFA	0EAB68A15	05C7D2F3C4	1	1	1	1	1	11	01	00
335	96	1355A6E	5797FFF4	1D56D142A	0B8FA5E788	0	1	1	1	0	11	11	01
336	97	06AB4DC	2F2FFFE8	1AADA2854	171F4BCF11	0	0	1	0	0	11	11	11
337	98	0D569B8	5E5FFFD0	155B450A9	2E3E979E23	1	0	0	0	0	11	11	11
338	99	1AAD370	3CBFFFA1	0AB68A153	5C7D2F3C46	1	1	1	0	0	10	11	11
339	100	155A6E0	797FFF43	156D142A7	38FA5E788D	0	0	0	1	1	01	10	11
340	101	0AB4DC0	72FFFE87	0ADA2854E	71F4BCF11B	1	1	1	1	1	10	01	10
341	102	1569B81	65FFFD0E	15B450A9D	63E979E236	0	1	0	1	0	11	10	01
342	103	0AD3703	4BFFFA1C	0B68A153B	47D2F3C46C	1	1	1	1	1	01	11	10
343	104	15A6E07	17FFF438	16D142A76	0FA5E788D8	0	1	0	1	1	10	01	11
344	105	0B4DC0F	2FFFE870	0DA2854EC	1F4BCF11B0	1	1	1	0	1	11	10	01
345	106	169B81F	5FFFD0E1	1B450A9D8	3E979E2360	0	1	1	1	0	01	11	10
346	107	0D3703F	3FFFA1C3	168A153B0	7D2F3C46C1	1	1	0	0	1	10	01	11
347	108	1A6E07E	7FFF4386	0D142A761	7A5E788D83	1	1	1	0	1	11	10	01
348	109	14DC0FD	7FFE870C	1A2854EC2	74BCF11B07	0	1	1	1	0	01	11	10
349	110	09B81FB	7FFD0E19	1450A9D84	6979E2360E	1	1	0	0	1	10	01	11
350	111	13703F6	7FFA1C33	08A153B09	52F3C46C1C	0	1	1	1	1	11	10	01
351	112	06E07EC	7FF43867	1142A7612	25E788D838	0	1	0	1	1	00	11	10
352	113	0DC0FD8	7FE870CF	02854EC25	4BCF11B071	1	1	0	1	1	11	00	11
353	114	1B81FB1	7FD0E19E	050A9D84B	179E2360E3	1	1	0	1	0	00	11	00
354	115	1703F62	7FA1C33D	0A153B096	2F3C46C1C7	0	1	1	0	0	11	00	11
355	116	0E07EC4	7F43867B	142A7612C	5E788D838E	1	0	0	0	0	01	11	00
356	117	1C0FD88	7E870CF6	0854EC259	3CF11B071C	1	1	1	1	1	01	01	11
357	118	181FB11	7D0E19ED	10A9D84B3	79E2360E38	1	0	0	1	1	11	01	01
358	119	103F622	7A1C33DA	0153B0967	73C46C1C71	0	0	0	1	0	10	11	01
359	120	007EC45	743867B5	02A7612CE	6788D838E3	0	0	0	1	1	01	10	11
360	121	00FD88B	6870CF6B	054EC259C	4F11B071C6	0	0	0	0	1	00	01	10
361	122	01FB117	50E19ED7	0A9D84B38	1E2360E38C	0	1	1	0	0	10	00	01
362	123	03F622F	21C33DAE	153B09671	3C46C1C718	0	1	0	0	1	11	10	00
363	124	07EC45F	43867B5C	0A7612CE2	788D838E30	0	1	1	1	0	01	11	10
364	125	0FD88BF	070CF6B9	14EC259C4	711B071C61	1	0	0	0	0	10	01	11

1.3 SECOND SET OF SAMPLE DATA

```
Initial values for the key, BD_ADDR and clock

K'c2[0]   = 00   K'c2[1]   = 00    K'c2[2]   = 00    K'c2[3]  = 00
K'c2[4]   = 00   K'c2[5]   = 00    K'c2[6]   = 00    K'c2[7]  = 00
K'c2[8]   = 00   K'c2[9]   = 00    K'c2[10]  = 00    K'c2[11] = 00
K'c2[12]  = 00   K'c2[13]  = 00    K'c2[14]  = 00    K'c2[15] = 00

Addr2[0] = 00   Addr2[1] = 00    Addr2[2] = 00
Addr2[3] = 00   Addr2[4] = 00    Addr2[5] = 00

CL2[0]  = 00   CL2[1]  = 00    CL2[2]  = 00    CL2[3]  = 03


=============================================================================================
         Fill LFSRs with initial data
=============================================================================================

  t  clk#    LFSR1    LFSR2     LFSR3      LFSR4           X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

  0     0    0000000* 00000000* 000000000* 0000000000*     0  0  0  0    0  00     00   00
  1     1    0000001* 00000001* 000000001* 0000000001*     0  0  0  0    0  00     00   00
  2     2    0000002* 00000002* 000000002* 0000000003*     0  0  0  0    0  00     00   00
  3     3    0000004* 00000004* 000000004* 0000000007*     0  0  0  0    0  00     00   00
  4     4    0000008* 00000008* 000000008* 000000000E*     0  0  0  0    0  00     00   00
  5     5    0000010* 00000010* 000000010* 000000001C*     0  0  0  0    0  00     00   00
  6     6    0000020* 00000020* 000000020* 0000000038*     0  0  0  0    0  00     00   00
  7     7    0000040* 00000040* 000000040* 0000000070*     0  0  0  0    0  00     00   00
  8     8    0000080* 00000080* 000000080* 00000000E0*     0  0  0  0    0  00     00   00
  9     9    0000100* 00000100* 000000100* 00000001C0*     0  0  0  0    0  00     00   00
 10    10    0000200* 00000200* 000000200* 0000000380*     0  0  0  0    0  00     00   00
 11    11    0000400* 00000400* 000000400* 0000000700*     0  0  0  0    0  00     00   00
 12    12    0000800* 00000800* 000000800* 0000000E00*     0  0  0  0    0  00     00   00
 13    13    0001000* 00001000* 000001000* 0000001C00*     0  0  0  0    0  00     00   00
 14    14    0002000* 00002000* 000002000* 0000003800*     0  0  0  0    0  00     00   00
 15    15    0004000* 00004000* 000004000* 0000007000*     0  0  0  0    0  00     00   00
 16    16    0008000* 00008000* 000008000* 000000E000*     0  0  0  0    0  00     00   00
 17    17    0010000* 00010000* 000010000* 000001C000*     0  0  0  0    0  00     00   00
 18    18    0020000* 00020000* 000020000* 0000038000*     0  0  0  0    0  00     00   00
 19    19    0040000* 00040000* 000040000* 0000070000*     0  0  0  0    0  00     00   00
 20    20    0080000* 00080000* 000080000* 00000E0000*     0  0  0  0    0  00     00   00
 21    21    0100000* 00100000* 000100000* 00001C0000*     0  0  0  0    0  00     00   00
 22    22    0200000* 00200000* 000200000* 0000380000*     0  0  0  0    0  00     00   00
 23    23    0400000* 00400000* 000400000* 0000700000*     0  0  0  0    0  00     00   00
 24    24    0800000* 00800000* 000800000* 0000E00000*     1  1  0  0    0  01     00   00
 25    25    1000000* 01000000* 001000000* 0001C00000*     0  0  0  0    0  00     00   00
 26    26    0000001  02000000* 002000000* 0003800000*     0  0  0  0    0  00     00   00
 27    27    0000002  04000000* 004000000* 0007000000*     0  0  0  0    0  00     00   00
 28    28    0000004  08000000* 008000000* 000E000000*     0  0  0  0    0  00     00   00
 29    29    0000008  10000000* 010000000* 001C000000*     0  0  0  0    0  00     00   00
 30    30    0000010  20000000* 020000000* 0038000000*     0  0  0  0    0  00     00   00
 31    31    0000020  40000000* 040000000* 0070000000*     0  0  0  0    0  00     00   00
 32    32    0000040  00000001  080000000* 00E0000000*     0  0  1  1    0  01     00   00
 33    33    0000080  00000002  100000000* 01C0000000*     0  0  0  1    1  00     00   00
 34    34    0000101  00000004  000000001  0380000000*     0  0  0  1    1  00     00   00
```

Sample Data

35	35	0000202	00000008	000000002	0700000000*	0	0	0	0	0	00	00	00
36	36	0000404	00000010	000000004	0E00000000*	0	0	0	0	0	00	00	00
37	37	0000808	00000020	000000008	1C00000000*	0	0	0	0	0	00	00	00
38	38	0001011	00000040	000000011	3800000000*	0	0	0	0	0	00	00	00
39	39	0002022	00000080	000000022	7000000000*	0	0	0	0	0	00	00	00

Start clocking Summation Combiner

40	1	0004044	00000100	000000044	6000000001	0	0	0	0	0	00	00	00
41	2	0008088	00000200	000000088	4000000003	0	0	0	0	0	00	00	00
42	3	0010111	00000400	000000111	0000000007	0	0	0	0	0	00	00	00
43	4	0020222	00000800	000000222	000000000E	0	0	0	0	0	00	00	00
44	5	0040444	00001001	000000444	000000001D	0	0	0	0	0	00	00	00
45	6	0080888	00002002	000000888	000000003B	0	0	0	0	0	00	00	00
46	7	0101111	00004004	000001111	0000000077	0	0	0	0	0	00	00	00
47	8	0202222	00008008	000002222	00000000EE	0	0	0	0	0	00	00	00
48	9	0404444	00010011	000004444	00000001DD	0	0	0	0	0	00	00	00
49	10	0808888	00020022	000008888	00000003BB	1	0	0	0	1	00	00	00
50	11	1011110	00040044	000011111	0000000777	0	0	0	0	0	00	00	00
51	12	0022221	00080088	000022222	0000000EEE	0	0	0	0	0	00	00	00
52	13	0044442	00100110	000044444	0000001DDD	0	0	0	0	0	00	00	00
53	14	0088884	00200220	000088888	0000003BBB	0	0	0	0	0	00	00	00
54	15	0111109	00400440	000111111	0000007777	0	0	0	0	0	00	00	00
55	16	0222212	00800880	000222222	000000EEEE	0	1	0	0	1	00	00	00
56	17	0444424	01001100	000444444	000001DDDD	0	0	0	0	0	00	00	00
57	18	0888848	02002200	000888888	000003BBBB	1	0	0	0	1	00	00	00
58	19	1111090	04004400	001111110	0000077777	0	0	0	0	0	00	00	00
59	20	0222120	08008800	002222220	00000EEEEE	0	0	0	0	0	00	00	00
60	21	0444240	10011000	004444440	00001DDDDD	0	0	0	0	0	00	00	00
61	22	0888480	20022000	008888880	00003BBBBB	1	0	0	0	1	00	00	00
62	23	1110900	40044000	011111100	0000777777	0	0	0	0	0	00	00	00
63	24	0221200	00088001	022222200	0000EEEEEE	0	0	0	0	0	00	00	00
64	25	0442400	00110003	044444400	0001DDDDDD	0	0	0	0	0	00	00	00
65	26	0884800	00220006	088888800	0003BBBBBB	1	0	1	0	0	01	00	00
66	27	1109000	0044000C	111111000	0007777777	0	0	0	0	1	01	01	00
67	28	0212001	00880018	022222001	000EEEEEEE	0	1	0	0	0	11	01	01
68	29	0424002	01100031	044444002	001DDDDDDC	0	0	0	0	1	01	11	01
69	30	0848004	02200062	088888004	003BBBBBB8	1	0	1	0	1	10	01	11
70	31	1090008	044000C4	111110008	0077777770	0	0	0	0	0	00	10	01
71	32	0120010	08800188	022220010	00EEEEEEE0	0	1	0	1	0	00	00	10
72	33	0240020	11000311	044440020	01DDDDDDC1	0	0	0	1	1	00	00	00
73	34	0480040	22000622	088880040	03BBBBBB83	0	0	1	1	0	01	00	00
74	35	0900081	44000C44	111100080	0777777707	1	0	0	0	0	00	01	00
75	36	1200103	08001888	022200101	0EEEEEEE0E	0	0	0	1	1	11	00	01
76	37	0400207	10003111	044400202	1DDDDDDC1D	0	0	0	1	0	01	11	00
77	38	080040E	20006222	088800404	3BBBBBB83B	1	0	1	1	0	01	01	11
78	39	100081C	4000C444	111000808	7777777077	0	0	0	0	1	10	01	01
79	40	0001038	00018888	022001010	6EEEEEE0EF	0	0	0	1	1	00	10	01
80	41	0002070	00031110	044002020	5DDDDDC1DE	0	0	0	1	1	01	00	10
81	42	00040E0	00062220	088004040	3BBBBB83BC	0	0	1	1	1	00	01	00
82	43	00081C1	000C4440	110008081	7777770779	0	0	0	0	0	11	00	01
83	44	0010383	00188880	020010103	6EEEEE0EF2	0	0	0	1	0	01	11	00
84	45	0020707	00311100	040020206	5DDDDC1DE5	0	0	0	1	0	10	01	11
85	46	0040E0E	00622200	08004040C	3BBBB83BCB	0	0	1	1	0	11	10	01
86	47	0081C1D	00C44400	100080819	7777707797	0	1	0	0	0	00	11	10
87	48	010383A	01888801	000101032	6EEEE0EF2F	0	1	0	1	0	11	00	11
88	49	0207075	03111003	000202064	5DDDC1DE5E	0	0	0	1	0	01	11	00

Sample Data

89	50	040E0EA	06222006	0004040C8	3BBB83BCBC	0	0	0	1	0	10	01	11
90	51	081C1D5	0C44400C	000808191	7777077979	1	0	0	0	1	00	10	01
91	52	10383AB	18888018	001010323	6EEE0EF2F2	0	1	0	1	0	00	00	10
92	53	0070756	31110030	002020646	5DDC1DE5E5	0	0	0	1	1	00	00	00
93	54	00E0EAC	62220060	004040C8C	3BB83BCBCB	0	0	0	1	1	00	00	00
94	55	01C1D59	444400C1	008081919	7770779797	0	0	0	0	0	00	00	00
95	56	0383AB2	08880183	010103232	6EE0EF2F2F	0	1	0	1	0	01	00	00
96	57	0707565	11100307	020206464	5DC1DE5E5F	0	0	0	1	0	00	01	00
97	58	0E0EACA	2220060E	04040C8C8	3B83BCBCBF	1	0	0	1	0	10	00	01
98	59	1C1D594	44400C1C	080819191	770779797E	1	0	1	0	0	00	10	00
99	60	183AB28	08801838	101032323	6E0EF2F2FC	1	1	0	0	0	00	00	10
100	61	1075650	11003070	002064647	5C1DE5E5F8	0	0	0	0	0	00	00	00
101	62	00EACA1	220060E0	0040C8C8E	383BCBCBF0	0	0	0	0	0	00	00	00
102	63	01D5943	4400C1C0	00819191D	70779797E0	0	0	0	0	0	00	00	00
103	64	03AB286	08018380	01032323A	60EF2F2FC1	0	0	0	1	1	00	00	00
104	65	075650C	10030701	020646475	41DE5E5F82	0	0	0	1	1	00	00	00
105	66	0EACA18	20060E02	040C8C8EA	03BCBCBF04	1	0	0	1	0	01	00	00
106	67	1D59430	400C1C05	0819191D4	0779797E09	1	0	1	0	1	00	01	00
107	68	1AB2861	0018380A	1032323A9	0EF2F2FC12	1	0	0	1	0	10	00	01
108	69	15650C3	00307015	006464752	1DE5E5F825	0	0	0	1	1	11	10	00
109	70	0ACA186	0060E02A	00C8C8EA4	3BCBCBF04B	1	0	0	1	1	00	11	10
110	71	159430C	00C1C055	019191D48	779797E097	0	1	0	1	0	11	00	11
111	72	0B28618	018380AA	032323A90	6F2F2FC12F	1	1	0	0	1	01	11	00
112	73	1650C30	03070154	064647520	5E5E5F825E	0	0	0	0	1	11	01	11
113	74	0CA1860	060E02A8	0C8C8EA40	3CBCBF04BC	1	0	1	1	0	11	11	01
114	75	19430C0	0C1C0550	19191D480	79797E0979	1	0	1	0	1	11	11	11
115	76	1286180	18380AA0	12323A900	72F2FC12F2	0	0	0	1	0	11	11	11
116	77	050C301	30701541	046475201	65E5F825E5	0	0	0	1	0	11	11	11
117	78	0A18602	60E02A82	08C8EA402	4BCBF04BCB	1	1	1	1	1	10	11	11
118	79	1430C04	41C05505	1191D4804	1797E09796	0	1	0	1	0	10	10	11
119	80	0861808	0380AA0A	0323A9008	2F2FC12F2C	1	1	0	0	0	01	10	10
120	81	10C3011	07015415	064752011	5E5F825E59	0	0	0	0	1	00	01	10
121	82	0186022	0E02A82A	0C8EA4022	3CBF04BCB2	0	0	1	1	0	10	00	01
122	83	030C045	1C055054	191D48044	797E097964	0	0	1	0	1	11	10	00
123	84	061808A	380AA0A8	123A90088	72FC12F2C9	0	0	0	1	0	00	11	10
124	85	0C30115	70154151	047520111	65F825E593	1	0	0	1	0	11	00	11
125	86	186022A	602A82A3	08EA40222	4BF04BCB26	1	0	1	1	0	00	11	00
126	87	10C0455	40550546	11D480444	17E097964C	0	0	0	1	1	10	00	11
127	88	01808AA	00AA0A8D	03A900888	2FC12F2C99	0	1	0	1	0	00	10	00
128	89	0301155	0154151A	075201111	5F825E5932	0	0	0	1	1	01	00	10
129	90	06022AA	02A82A34	0EA402222	3F04BCB264	0	1	1	0	1	00	01	00
130	91	0C04555	05505468	1D4804445	7E097964C9	1	0	1	0	0	10	00	01
131	92	1808AAA	0AA0A8D0	1A900888A	7C12F2C992	1	1	1	0	1	00	10	00
132	93	1011555	154151A1	152011115	7825E59324	0	0	0	0	0	01	00	10
133	94	0022AAB	2A82A342	0A402222B	704BCB2648	0	1	1	0	1	00	01	00
134	95	0045556	55054684	148044457	6097964C91	0	0	0	1	1	11	00	01
135	96	008AAAC	2A0A8D09	0900888AE	412F2C9923	0	0	1	0	0	01	11	00
136	97	0115559	54151A12	12011115D	025E593246	0	0	0	0	1	11	01	11
137	98	022AAB2	282A3424	0402222BA	04BCB2648D	0	0	0	1	0	10	11	01
138	99	0455564	50546848	080444575	097964C91A	0	0	1	0	1	01	10	11
139	100	08AAAC8	20A8D090	100888AEA	12F2C99235	1	1	0	1	0	10	01	10
140	101	1155591	4151A120	0011115D5	25E593246A	0	0	0	1	1	00	10	01
141	102	02AAB22	02A34240	002222BAA	4BCB2648D5	0	1	0	1	0	00	00	10
142	103	0555644	05468481	004445755	17964C91AB	0	0	0	1	1	00	00	00
143	104	0AAAC88	0A8D0903	00888AEAA	2F2C992357	1	1	0	0	0	01	00	00
144	105	1555911	151A1206	011115D55	5E593246AE	0	0	0	0	1	01	01	00
145	106	0AAB222	2A34240C	02222BAAA	3CB2648D5C	1	0	0	1	1	11	01	01

Sample Data

146	107	1556445	54684818	044457555	7964C91AB8	0	0	0	0	1	01	11	01
147	108	0AAC88B	28D09030	0888AEAAA	72C9923571	1	1	1	1	1	01	01	11
148	109	1559117	51A12060	11115D555	6593246AE2	0	1	0	1	1	11	01	01
149	110	0AB222F	234240C0	0222BAAAB	4B2648D5C5	1	0	0	0	0	10	11	01
150	111	156445F	46848180	044575557	164C91AB8A	0	1	0	0	1	01	10	11
151	112	0AC88BF	0D090301	088AEAAAE	2C99235714	1	0	1	1	0	10	01	10
152	113	159117F	1A120602	1115D555D	593246AE28	0	0	0	0	0	00	10	01
153	114	0B222FE	34240C04	022BAAABA	32648D5C51	1	0	0	0	1	01	00	10
154	115	16445FD	68481809	045755574	64C91AB8A2	0	0	0	1	0	00	01	00
155	116	0C88BFA	50903012	08AEAAAE8	4992357144	1	1	1	1	0	01	00	01
156	117	19117F5	21206024	115D555D1	13246AE288	1	0	0	0	0	00	01	00
157	118	1222FEA	4240C048	02BAAABA2	2648D5C511	0	0	0	0	0	11	00	01
158	119	0445FD5	04818090	057555744	4C91AB8A23	0	1	0	1	1	01	11	00
159	120	088BFAA	09030120	0AEAAAE88	1923571446	1	0	1	0	1	10	01	11
160	121	1117F55	12060240	15D555D11	3246AE288D	0	0	0	0	0	00	10	01
161	122	022FEAA	240C0480	0BAAABA22	648D5C511B	0	0	1	1	0	00	00	10
162	123	045FD54	48180900	175557444	491AB8A237	0	0	0	0	0	00	00	00
163	124	08BFAA9	10301200	0EAAAE889	123571446F	1	0	1	0	0	01	00	00
164	125	117F553	20602400	1D555D113	246AE288DF	0	0	1	0	0	00	01	00
165	126	02FEAA7	40C04800	1AAABA227	48D5C511BE	0	1	1	1	1	10	00	01
166	127	05FD54F	01809001	15557444F	11AB8A237D	0	1	0	1	0	00	10	00
167	128	0BFAA9F	03012002	0AAAE889E	23571446FA	1	0	1	0	0	00	00	10
168	129	17F553F	06024004	1555D113D	46AE288DF5	0	0	0	1	1	00	00	00
169	130	0FEAA7E	0C048008	0AABA227A	0D5C511BEA	1	0	1	0	0	01	00	00
170	131	1FD54FC	18090011	1557444F5	1AB8A237D5	1	0	0	1	1	00	01	00
171	132	1FAA9F9	30120022	0AAE889EB	3571446FAA	1	0	1	0	0	10	00	01
172	133	1F553F2	60240044	155D113D7	6AE288DF55	1	0	0	1	0	00	10	00
173	134	1EAA7E4	40480089	0ABA227AE	55C511BEAA	1	0	1	1	1	00	00	10
174	135	1D54FC9	00900113	157444F5D	2B8A237D54	1	1	0	1	1	01	00	00
175	136	1AA9F93	01200227	0AE889EBA	571446FAA8	1	0	1	0	1	00	01	00
176	137	1553F26	0240044E	15D113D75	2E288DF550	0	0	0	0	0	11	00	01
177	138	0AA7E4C	0480089C	0BA227AEA	5C511BEAA0	1	1	1	0	0	00	11	00
178	139	154FC98	09001138	17444F5D4	38A237D540	0	0	0	1	1	10	00	11
179	140	0A9F931	12002270	0E889EBA9	71446FAA81	1	0	1	0	0	00	10	00
180	141	153F262	240044E0	1D113D753	6288DF5503	0	0	1	1	0	00	00	10
181	142	0A7E4C5	480089C0	1A227AEA7	4511BEAA06	1	0	1	0	0	01	00	00
182	143	14FC98B	10011381	1444F5D4F	0A237D540D	0	0	0	0	1	01	01	00
183	144	09F9316	20022702	0889EBA9E	1446FAA81A	1	0	1	0	1	11	01	01
184	145	13F262D	40044E04	1113D753D	288DF55035	0	0	0	1	0	10	11	01
185	146	07E4C5A	00089C08	0227AEA7A	511BEAA06A	0	0	0	0	0	01	10	11
186	147	0FC98B4	00113810	044F5D4F5	2237D540D5	1	0	0	0	0	01	01	10
187	148	1F93169	00227021	089EBA9EB	446FAA81AA	1	0	1	0	1	11	01	01
188	149	1F262D2	0044E042	113D753D7	08DF550355	1	0	0	1	1	10	11	01
189	150	1E4C5A4	0089C085	027AEA7AE	11BEAA06AA	1	1	0	1	1	10	10	11
190	151	1C98B48	0113810A	04F5D4F5C	237D540D54	1	0	0	0	1	10	10	10
191	152	1931691	02270215	09EBA9EB8	46FAA81AA9	1	0	1	1	1	01	10	10
192	153	1262D22	044E042A	13D753D71	0DF5503553	0	0	0	1	0	01	01	10
193	154	04C5A44	089C0854	07AEA7AE2	1BEAA06AA7	0	1	0	1	1	11	01	01
194	155	098B488	113810A8	0F5D4F5C4	37D540D54E	1	0	1	1	0	11	11	01
195	156	1316910	22702150	1EBA9EB89	6FAA81AA9D	0	0	1	1	1	11	11	11
196	157	062D220	44E042A0	1D753D712	5F5503553A	0	1	1	0	1	11	11	11
197	158	0C5A440	09C08540	1AEA7AE25	3EAA06AA75	1	1	1	1	1	10	11	11
198	159	18B4880	13810A80	15D4F5C4B	7D540D54EA	1	1	0	0	0	10	10	11
199	160	1169100	27021500	0BA9EB897	7AA81AA9D5	0	0	1	1	0	01	10	10
200	161	02D2201	4E042A00	1753D712E	75503553AB	0	0	0	0	1	00	01	10
201	162	05A4403	1C085400	0EA7AE25C	6AA06AA756	0	0	1	1	0	10	00	01
202	163	0B48807	3810A800	1D4F5C4B8	5540D54EAC	1	0	1	0	0	00	10	00

Sample Data

203	164	169100F	70215000	1A9EB8971	2A81AA9D58	0	0	1	1	0	00	00	10
204	165	0D2201E	6042A001	153D712E3	5503553AB0	1	0	0	0	1	00	00	00
205	166	1A4403C	40854002	0A7AE25C6	2A06AA7561	1	1	1	0	1	01	00	00
206	167	1488079	010A8004	14F5C4B8D	540D54EAC3	0	0	0	0	1	01	01	00
207	168	09100F2	02150009	09EB8971B	281AA9D586	1	0	1	0	1	11	01	01
208	169	12201E5	042A0012	13D712E37	503553AB0C	0	0	0	0	1	01	11	01
209	170	04403CA	08540024	07AE25C6E	206AA75618	0	0	0	0	1	11	01	11
210	171	0880795	10A80048	0F5C4B8DD	40D54EAC30	1	1	1	1	1	11	11	01
211	172	1100F2A	21500091	1EB8971BA	01AA9D5861	0	0	1	1	1	11	11	11
212	173	0201E54	42A00122	1D712E374	03553AB0C3	0	1	1	0	1	11	11	11
213	174	0403CA9	05400244	1AE25C6E9	06AA756186	0	0	1	1	1	11	11	11
214	175	0807952	0A800488	15C4B8DD3	0D54EAC30D	1	1	0	0	1	11	11	11
215	176	100F2A5	15000911	0B8971BA6	1AA9D5861A	0	0	1	1	1	11	11	11
216	177	001E54A	2A001223	1712E374C	3553AB0C35	0	0	0	0	1	00	11	11
217	178	003CA94	54002446	0E25C6E98	6AA756186A	0	0	1	1	0	11	00	11
218	179	0079528	2800488D	1C4B8DD31	554EAC30D5	0	0	1	0	0	01	11	00
219	180	00F2A50	5000911B	18971BA62	2A9D5861AA	0	0	1	1	1	10	01	11
220	181	01E54A0	20012236	112E374C4	553AB0C355	0	0	0	0	0	00	10	01
221	182	03CA940	4002446C	025C6E988	2A756186AA	0	0	0	0	0	01	00	10
222	183	0795280	000488D9	04B8DD310	54EAC30D54	0	0	0	1	0	00	01	00
223	184	0F2A500	000911B2	0971BA620	29D5861AA8	1	0	1	1	1	10	00	01
224	185	1E54A00	00122364	12E374C40	53AB0C3550	1	0	0	1	0	00	10	00
225	186	1CA9400	002446C8	05C6E9880	2756186AA0	1	0	0	0	1	01	00	10
226	187	1952800	00488D90	0B8DD3101	4EAC30D540	1	0	1	1	0	11	01	00
227	188	12A5000	00911B20	171BA6202	1D5861AA81	0	1	0	0	0	10	11	01
228	189	054A000	01223640	0E374C404	3AB0C35502	0	0	1	1	0	10	10	11
229	190	0A94000	02446C80	1C6E98808	756186AA05	1	0	1	0	0	01	10	10
230	191	1528001	0488D901	18DD31011	6AC30D540B	0	1	1	1	0	10	01	10
231	192	0A50003	0911B203	11BA62023	55861AA817	1	0	0	1	0	11	10	01
232	193	14A0006	12236407	0374C4047	2B0C35502F	0	0	0	0	1	11	11	10
233	194	094000C	2446C80E	06E98808E	56186AA05F	1	0	0	0	0	11	11	11
234	195	1280018	488D901C	0DD31011D	2C30D540BF	0	1	1	0	1	11	11	11
235	196	0500030	111B2039	1BA62023A	5861AA817E	0	0	1	0	0	11	11	11
236	197	0A00060	22364072	174C40475	30C35502FD	1	0	0	1	1	11	11	11
237	198	14000C0	446C80E4	0E98808EA	6186AA05FB	0	0	1	1	1	11	11	11
238	199	0800180	08D901C8	1D31011D5	430D540BF6	1	1	1	0	0	10	11	11
239	200	1000301	11B20391	1A62023AB	061AA817EC	0	1	1	0	0	10	10	11

```
Z[0]  = 25
Z[1]  = 45
Z[2]  = 6B
Z[3]  = 55
Z[4]  = 5F
Z[5]  = C2
Z[6]  = 20
Z[7]  = E5
Z[8]  = C4
Z[9]  = F8
Z[10] = 3A
Z[11] = F1
Z[12] = FF
Z[13] = 89
Z[14] = 02
Z[15] = 35

================================================================================================
         Reload this pattern into the LFSRs
```

Sample Data

```
         Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 1C45F25
LFSR2  <= 7FF8C245
LFSR3  <= 1893A206B
LFSR4  <= 1A02F1E555
C[t+1] <= 10

================================================================================================
         Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240     1    1C45F25  7FF8C245  1893A206B  1A02F1E555      1  1  1  0    1  10     10   11
241     2    188BE4A  7FF1848B  1127440D7  3405E3CAAB      1  1  0  0    0  01     10   10
242     3    1117C95  7FE30917  024E881AF  680BC79557      0  1  0  0    0  01     01   10
243     4    022F92B  7FC6122F  049D1035E  50178F2AAF      0  1  0  0    0  11     01   01
244     5    045F257  7F8C245E  093A206BD  202F1E555E      0  1  1  0    1  10     11   01
245     6    08BE4AE  7F1848BC  127440D7A  405E3CAABC      1  0  0  0    1  01     10   11
246     7    117C95C  7E309178  04E881AF4  00BC795579      0  0  0  1    0  01     01   10
247     8    02F92B8  7C6122F0  09D1035E8  0178F2AAF2      0  0  1  0    0  11     01   01
248     9    05F2570  78C245E1  13A206BD0  02F1E555E5      0  1  0  1    1  10     11   01
249    10    0BE4AE1  71848BC2  07440D7A0  05E3CAABCA      1  1  0  1    1  10     10   11
250    11    17C95C3  63091784  0E881AF40  0BC7955795      0  0  1  1    0  01     10   10
251    12    0F92B87  46122F09  1D1035E80  178F2AAF2B      1  0  1  1    0  10     01   10
252    13    1F2570F  0C245E12  1A206BD01  2F1E555E56      1  0  1  0    0  11     10   01
253    14    1E4AE1F  1848BC25  1440D7A03  5E3CAABCAC      1  0  0  0    0  00     11   10
254    15    1C95C3E  3091784A  0881AF407  3C79557958      1  1  1  0    1  11     00   11
255    16    192B87D  6122F094  11035E80F  78F2AAF2B1      1  0  0  1    1  01     11   00
256    17    12570FA  4245E128  0206BD01E  71E555E562      0  0  0  1    0  10     01   11
257    18    04AE1F4  048BC250  040D7A03D  63CAABCAC5      0  1  0  1    0  11     10   01
258    19    095C3E8  091784A0  081AF407A  479557958A      1  0  1  1    0  01     11   10
259    20    12B87D1  122F0941  1035E80F4  0F2AAF2B14      0  0  0  0    1  11     01   11
260    21    0570FA3  245E1283  006BD01E9  1E555E5628      0  0  0  0    1  01     11   01
261    22    0AE1F46  48BC2506  00D7A03D2  3CAABCAC50      1  1  0  1    0  01     01   11
262    23    15C3E8C  11784A0C  01AF407A5  79557958A0      0  0  0  0    1  10     01   01
263    24    0B87D18  22F09419  035E80F4A  72AAF2B140      1  1  0  1    1  11     10   01
264    25    170FA30  45E12832  06BD01E94  6555E56280      0  1  0  0    0  00     11   10
265    26    0E1F460  0BC25065  0D7A03D28  4AABCAC501      1  1  1  1    0  00     00   11
266    27    1C3E8C0  1784A0CB  1AF407A50  1557958A03      1  1  1  0    1  01     00   00
267    28    187D181  2F094196  15E80F4A0  2AAF2B1406      1  0  0  1    1  00     01   00
268    29    10FA302  5E12832C  0BD01E941  555E56280C      0  0  1  0    1  11     00   01
269    30    01F4604  3C250658  17A03D283  2ABCAC5019      0  0  0  1    0  01     11   00
270    31    03E8C09  784A0CB0  0F407A506  557958A033      0  0  1  0    0  10     01   11
271    32    07D1812  70941960  1E80F4A0C  2AF2B14066      0  1  1  1    1  11     10   01
272    33    0FA3024  612832C1  1D01E9419  55E56280CD      1  0  1  1    0  01     11   10
273    34    1F46049  42506583  1A03D2832  2BCAC5019A      1  0  1  1    0  01     01   11
274    35    1E8C093  04A0CB07  1407A5065  57958A0335      1  1  0  1    0  00     01   01
275    36    1D18127  0941960F  080F4A0CB  2F2B14066B      1  0  1  0    0  10     00   01
276    37    1A3024F  12832C1F  101E94196  5E56280CD7      1  1  0  0    0  00     10   00
277    38    146049F  2506583E  003D2832C  3CAC5019AE      0  0  0  1    1  01     00   10
278    39    08C093E  4A0CB07D  007A50658  7958A0335D      1  0  0  0    0  00     01   00
279    40    118127C  141960FA  00F4A0CB0  72B14066BA      0  0  0  1    1  11     00   01
280    41    03024F8  2832C1F4  01E941961  656280CD74      0  0  0  0    1  10     11   00
281    42    06049F1  506583E9  03D2832C2  4AC5019AE9      0  0  0  1    1  01     10   11
282    43    0C093E2  20CB07D2  07A506585  158A0335D3      1  1  0  1    0  10     01   10
283    44    18127C5  41960FA5  0F4A0CB0B  2B14066BA7      1  1  1  0    1  11     10   01
284    45    1024F8A  032C1F4B  1E9419616  56280CD74F      0  0  1  0    0  00     11   10
285    46    0049F15  06583E97  1D2832C2C  2C5019AE9F      0  0  1  0    1  10     00   11
```

Sample Data

286	47	0093E2B	0CB07D2F	1A5065859	58A0335D3E	0	1	1	1	1	00	10	00
287	48	0127C56	1960FA5E	14A0CB0B2	314066BA7D	0	0	0	0	0	01	00	10
288	49	024F8AD	32C1F4BC	094196164	6280CD74FB	0	1	1	1	0	11	01	00
289	50	049F15A	6583E978	12832C2C8	45019AE9F6	0	1	0	0	0	10	11	01
290	51	093E2B5	4B07D2F0	050658591	0A0335D3ED	1	0	0	0	1	01	10	11
291	52	127C56B	160FA5E0	0A0CB0B22	14066BA7DA	0	0	1	0	0	01	01	10
292	53	04F8AD7	2C1F4BC1	141961645	280CD74FB5	0	0	0	0	1	10	01	01
293	54	09F15AF	583E9783	0832C2C8A	5019AE9F6A	1	0	1	0	0	11	10	01
294	55	13E2B5E	307D2F06	106585915	20335D3ED5	0	0	0	0	1	11	11	10
295	56	07C56BD	60FA5E0D	00CB0B22B	4066BA7DAA	0	1	0	0	0	11	11	11
296	57	0F8AD7A	41F4BC1B	019616457	00CD74FB54	1	1	0	1	0	10	11	11
297	58	1F15AF4	03E97836	032C2C8AF	019AE9F6A9	1	1	0	1	1	10	10	11
298	59	1E2B5E9	07D2F06C	06585915E	0335D3ED52	1	1	0	0	0	01	10	10
299	60	1C56BD2	0FA5E0D8	0CB0B22BC	066BA7DAA4	1	1	1	0	0	10	01	10
300	61	18AD7A5	1F4BC1B0	196164578	0CD74FB549	1	0	1	1	1	11	10	01
301	62	115AF4B	3E978361	12C2C8AF0	19AE9F6A92	0	1	0	1	1	00	11	10
302	63	02B5E96	7D2F06C2	0585915E0	335D3ED524	0	0	0	0	0	10	00	11
303	64	056BD2D	7A5E0D85	0B0B22BC1	66BA7DAA49	0	0	1	1	0	00	10	00
304	65	0AD7A5B	74BC1B0A	161645783	4D74FB5493	1	1	0	0	0	00	00	10
305	66	15AF4B6	69783615	0C2C8AF07	1AE9F6A926	0	0	1	1	0	01	00	00
306	67	0B5E96D	52F06C2B	185915E0F	35D3ED524C	1	1	1	1	1	11	01	00
307	68	16BD2DB	25E0D857	10B22BC1F	6BA7DAA499	0	1	0	1	1	10	11	01
308	69	0D7A5B7	4BC1B0AF	01645783F	574FB54933	1	1	0	0	0	10	10	11
309	70	1AF4B6F	1783615F	02C8AF07F	2E9F6A9266	1	1	0	1	1	01	10	10
310	71	15E96DF	2F06C2BF	05915E0FF	5D3ED524CC	0	0	0	0	1	00	01	10
311	72	0BD2DBF	5E0D857F	0B22BC1FE	3A7DAA4998	1	0	1	0	0	10	00	01
312	73	17A5B7F	3C1B0AFE	1645783FD	74FB549331	0	0	0	1	1	11	10	00
313	74	0F4B6FF	783615FD	0C8AF07FA	69F6A92662	1	0	1	1	0	01	11	10
314	75	1E96DFF	706C2BFB	1915E0FF5	53ED524CC4	1	0	1	1	0	01	01	11
315	76	1D2DBFE	60D857F6	122BC1FEB	27DAA49988	1	1	0	1	0	00	01	01
316	77	1A5B7FD	41B0AFEC	045783FD7	4FB5493310	1	1	0	1	1	10	00	01
317	78	14B6FFA	03615FD8	08AF07FAE	1F6A926620	0	0	1	0	1	11	10	00
318	79	096DFF4	06C2BFB1	115E0FF5D	3ED524CC40	1	1	0	1	0	01	11	10
319	80	12DBFE8	0D857F63	02BC1FEBB	7DAA499881	0	1	0	1	1	10	01	11
320	81	05B7FD0	1B0AFEC6	05783FD77	7B54933103	0	0	0	0	0	00	10	01
321	82	0B6FFA1	3615FD8C	0AF07FAEF	76A9266206	1	0	1	1	1	00	00	10
322	83	16DFF42	6C2BFB18	15E0FF5DE	6D524CC40C	0	0	0	0	0	00	00	00
323	84	0DBFE85	5857F631	0BC1FEBBD	5AA4998819	1	0	1	1	1	01	00	00
324	85	1B7FD0B	30AFEC62	1783FD77A	3549331033	1	1	0	0	1	00	01	00
325	86	16FFA16	615FD8C5	0F07FAEF5	6A92662067	0	0	1	1	0	10	00	01
326	87	0DFF42D	42BFB18B	1E0FF5DEA	5524CC40CE	1	1	1	0	1	00	10	00
327	88	1BFE85B	057F6317	1C1FEBBD5	2A4998819C	1	0	1	0	0	00	00	10
328	89	17FD0B7	0AFEC62E	183FD77AA	5493310339	0	1	1	1	1	01	00	00
329	90	0FFA16F	15FD8C5C	107FAEF55	2926620672	1	1	0	0	1	00	01	00
330	91	1FF42DF	2BFB18B9	00FF5DEAA	524CC40CE5	1	1	0	0	0	10	00	01
331	92	1FE85BF	57F63172	01FEBBD55	24998819CA	1	1	0	1	1	00	10	00
332	93	1FD0B7F	2FEC62E4	03FD77AAA	4933103394	1	1	0	0	0	00	00	10
333	94	1FA16FF	5FD8C5C9	07FAEF555	1266206728	1	1	0	0	0	01	00	00
334	95	1F42DFF	3FB18B93	0FF5DEAAA	24CC40CE51	1	1	1	1	1	11	01	00
335	96	1E85BFF	7F631727	1FEBBD554	4998819CA3	1	0	1	1	0	11	11	01
336	97	1D0B7FE	7EC62E4F	1FD77AAA9	1331033947	1	1	1	0	0	10	11	11
337	98	1A16FFC	7D8C5C9F	1FAEF5553	266206728E	1	1	1	0	1	10	10	11
338	99	142DFF9	7B18B93F	1F5DEAAA7	4CC40CE51D	0	0	1	1	0	01	10	10
339	100	085BFF3	7631727F	1EBBD554E	198819CA3B	1	0	1	1	0	10	01	10
340	101	10B7FE6	6C62E4FF	1D77AAA9C	3310339477	0	0	1	0	1	00	10	01
341	102	016FFCC	58C5C9FE	1AEF55538	66206728EE	0	1	1	0	0	00	00	10
342	103	02DFF98	318B93FC	15DEAAA70	4C40CE51DC	0	1	0	0	1	00	00	00

Sample Data

343	104	05BFF31	631727F8	0BBD554E1	18819CA3B9	0	0	1	1	0	01	00	00
344	105	0B7FE62	462E4FF1	177AAA9C2	3103394772	1	0	0	0	0	00	01	00
345	106	16FFCC5	0C5C9FE2	0EF555384	6206728EE4	0	0	1	0	1	11	00	01
346	107	0DFF98A	18B93FC4	1DEAAA709	440CE51DC9	1	1	1	0	0	00	11	00
347	108	1BFF315	31727F88	1BD554E12	0819CA3B93	1	0	1	0	0	11	00	11
348	109	17FE62A	62E4FF11	17AAA9C24	1033947726	0	1	0	0	0	01	11	00
349	110	0FFCC54	45C9FE22	0F5553849	206728EE4C	1	1	1	0	0	01	01	11
350	111	1FF98A8	0B93FC44	1EAAA7093	40CE51DC99	1	1	1	1	1	00	01	01
351	112	1FF3150	1727F889	1D554E127	019CA3B933	1	0	1	1	1	10	00	01
352	113	1FE62A0	2E4FF112	1AAA9C24F	0339477267	1	0	1	0	0	00	10	00
353	114	1FCC541	5C9FE225	15553849E	06728EE4CF	1	1	0	0	0	00	00	10
354	115	1F98A82	393FC44B	0AAA7093C	0CE51DC99F	1	0	1	1	1	01	00	00
355	116	1F31504	727F8897	1554E1279	19CA3B933E	1	0	0	1	1	00	01	00
356	117	1E62A09	64FF112F	0AA9C24F2	339477267D	1	1	1	1	0	01	00	01
357	118	1CC5412	49FE225E	1553849E4	6728EE4CFB	1	1	0	0	1	00	01	00
358	119	198A824	13FC44BC	0AA7093C9	4E51DC99F7	1	1	1	0	1	10	00	01
359	120	1315049	27F88979	154E12792	1CA3B933EE	0	1	0	1	0	00	10	00
360	121	062A093	4FF112F3	0A9C24F24	39477267DC	0	1	1	0	0	00	00	10
361	122	0C54127	1FE225E6	153849E48	728EE4CFB8	1	1	0	1	1	01	00	00
362	123	18A824E	3FC44BCD	0A7093C91	651DC99F71	1	1	1	0	0	11	01	00
363	124	115049C	7F88979A	14E127922	4A3B933EE2	0	1	0	0	0	10	11	01
364	125	02A0938	7F112F35	09C24F244	1477267DC5	0	0	1	0	1	01	10	11

1.4 THIRD SET OF SAMPLES

```
Initial values for the key, pan address and clock

K'c3[0]   = FF   K'c3[1]   = FF    K'c3[2]   = FF    K'c3[3]  = FF
K'c3[4]   = FF   K'c3[5]   = FF    K'c3[6]   = FF    K'c3[7]  = FF
K'c3[8]   = FF   K'c3[9]   = FF    K'c3[10]  = FF    K'c3[11] = FF
K'c3[12]  = FF   K'c3[13]  = FF    K'c3[14]  = FF    K'c3[15] = FF

Addr3[0] = FF   Addr3[1] = FF    Addr3[2] = FF
Addr3[3] = FF   Addr3[4] = FF    Addr3[5] = FF

CL3[0]  = FF   CL3[1]  = FF    CL3[2]  = FF    CL3[3]  = 03


================================================================================================
         Fill LFSRs with initial data
================================================================================================

  t  clk#    LFSR1    LFSR2     LFSR3      LFSR4          X1 X2 X3 X4   Z  C[t+1] C[t] C[t-1]

  0     0    0000000* 00000000* 000000000* 0000000000*    0  0  0  0    0  00     00   00
  1     1    0000001* 00000001* 000000001* 0000000001*    0  0  0  0    0  00     00   00
  2     2    0000003* 00000002* 000000003* 0000000003*    0  0  0  0    0  00     00   00
  3     3    0000007* 00000004* 000000007* 0000000007*    0  0  0  0    0  00     00   00
  4     4    000000F* 00000009* 00000000F* 000000000F*    0  0  0  0    0  00     00   00
  5     5    000001F* 00000013* 00000001F* 000000001F*    0  0  0  0    0  00     00   00
  6     6    000003F* 00000027* 00000003F* 000000003F*    0  0  0  0    0  00     00   00
  7     7    000007F* 0000004F* 00000007F* 000000007F*    0  0  0  0    0  00     00   00
  8     8    00000FF* 0000009F* 0000000FF* 00000000FF*    0  0  0  0    0  00     00   00
  9     9    00001FF* 0000013F* 0000001FF* 00000001FF*    0  0  0  0    0  00     00   00
 10    10    00003FF* 0000027F* 0000003FF* 00000003FF*    0  0  0  0    0  00     00   00
 11    11    00007FF* 000004FF* 0000007FF* 00000007FF*    0  0  0  0    0  00     00   00
 12    12    0000FFF* 000009FF* 000000FFF* 0000000FFF*    0  0  0  0    0  00     00   00
 13    13    0001FFF* 000013FF* 000001FFF* 0000001FFF*    0  0  0  0    0  00     00   00
 14    14    0003FFF* 000027FF* 000003FFF* 0000003FFF*    0  0  0  0    0  00     00   00
 15    15    0007FFF* 00004FFF* 000007FFF* 0000007FFF*    0  0  0  0    0  00     00   00
 16    16    000FFFF* 00009FFF* 00000FFFF* 000000FFFF*    0  0  0  0    0  00     00   00
 17    17    001FFFF* 00013FFF* 00001FFFF* 000001FFFF*    0  0  0  0    0  00     00   00
 18    18    003FFFF* 00027FFF* 00003FFFF* 000003FFFF*    0  0  0  0    0  00     00   00
 19    19    007FFFF* 0004FFFF* 00007FFFF* 000007FFFF*    0  0  0  0    0  00     00   00
 20    20    00FFFFF* 0009FFFF* 0000FFFFF* 00000FFFFF*    0  0  0  0    0  00     00   00
 21    21    01FFFFF* 0013FFFF* 0001FFFFF* 00001FFFFF*    0  0  0  0    0  00     00   00
 22    22    03FFFFF* 0027FFFF* 0003FFFFF* 00003FFFFF*    0  0  0  0    0  00     00   00
 23    23    07FFFFF* 004FFFFF* 0007FFFFF* 00007FFFFF*    0  0  0  0    0  00     00   00
 24    24    0FFFFFF* 009FFFFF* 000FFFFFF* 0000FFFFFF*    1  1  0  0    0  01     00   00
 25    25    1FFFFFF* 013FFFFF* 001FFFFFF* 0001FFFFFF*    1  0  0  0    1  00     00   00
 26    26    1FFFFFF  027FFFFF* 003FFFFFF* 0003FFFFFF*    1  0  0  0    1  00     00   00
 27    27    1FFFFFF  04FFFFFF* 007FFFFFF* 0007FFFFFF*    1  1  0  0    0  01     00   00
 28    28    1FFFFFF  09FFFFFF* 00FFFFFFF* 000FFFFFFF*    1  1  0  0    0  01     00   00
 29    29    1FFFFFF  13FFFFFF* 01FFFFFFF* 001FFFFFFF*    1  1  0  0    0  01     00   00
 30    30    1FFFFFF  27FFFFFF* 03FFFFFFF* 003FFFFFFF*    1  1  0  0    0  01     00   00
 31    31    1FFFFFF  4FFFFFFF* 07FFFFFFF* 007FFFFFFF*    1  1  0  0    0  01     00   00
 32    32    1FFFFFF  1FFFFFFF  0FFFFFFFF* 00FFFFFFFF*    1  1  1  1    0  10     00   00
 33    33    1FFFFFF  3FFFFFFE  1FFFFFFFF* 01FFFFFFFF*    1  1  1  1    0  10     00   00
 34    34    1FFFFFF  7FFFFFFC  1FFFFFFFF  03FFFFFFFF*    1  1  1  1    0  10     00   00
```

Sample Data

35	35	1FFFFFF	7FFFFFF9	1FFFFFFFF	07FFFFFFFF*	1	1	1	1	0	10	00	00
36	36	1FFFFFF	7FFFFFF3	1FFFFFFFF	0FFFFFFFFF*	1	1	1	1	0	10	00	00
37	37	1FFFFFF	7FFFFFE7	1FFFFFFFF	1FFFFFFFFF*	1	1	1	1	0	10	00	00
38	38	1FFFFFF	7FFFFFCF	1FFFFFFFF	3FFFFFFFFF*	1	1	1	1	0	10	00	00
39	39	1FFFFFF	7FFFFF9F	1FFFFFFFF	7FFFFFFFFF*	1	1	1	1	0	10	00	00

Start clocking Summation Combiner

40	1	1FFFFFF	7FFFFF3F	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	0	01	10	00
41	2	1FFFFFF	7FFFFE7F	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	1	10	01	10
42	3	1FFFFFF	7FFFFCFF	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	0	10	10	01
43	4	1FFFFFF	7FFFF9FF	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	0	00	10	10
44	5	1FFFFFF	7FFFF3FF	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	0	11	00	10
45	6	1FFFFFF	7FFFE7FE	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	1	00	11	00
46	7	1FFFFFF	7FFFCFFC	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	0	00	00	11
47	8	1FFFFFF	7FFF9FF9	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	0	10	00	00
48	9	1FFFFFF	7FFF3FF3	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	0	01	10	00
49	10	1FFFFFF	7FFE7FE6	1FFFFFFFF	7FFFFFFFFF	1	1	1	1	1	10	01	10
50	11	1FFFFFE	7FFCFFCC	1FFFFFFFE	7FFFFFFFFF	1	1	1	1	0	10	10	01
51	12	1FFFFFC	7FF9FF99	1FFFFFFFC	7FFFFFFFFF	1	1	1	1	0	00	10	10
52	13	1FFFFF8	7FF3FF33	1FFFFFFF8	7FFFFFFFFF	1	1	1	1	0	11	00	10
53	14	1FFFFF0	7FE7FE67	1FFFFFFF0	7FFFFFFFFF	1	1	1	1	1	00	11	00
54	15	1FFFFE0	7FCFFCCF	1FFFFFFE1	7FFFFFFFFF	1	1	1	1	0	00	00	11
55	16	1FFFFC0	7F9FF99F	1FFFFFFC3	7FFFFFFFFF	1	1	1	1	0	10	00	00
56	17	1FFFF80	7F3FF33E	1FFFFFF87	7FFFFFFFFE	1	0	1	1	1	00	10	00
57	18	1FFFF00	7E7FE67C	1FFFFFF0F	7FFFFFFFFC	1	0	1	1	1	00	00	10
58	19	1FFFE01	7CFFCCF8	1FFFFFE1E	7FFFFFFFF8	1	1	1	1	0	10	00	00
59	20	1FFFC03	79FF99F0	1FFFFFC3C	7FFFFFFFF0	1	1	1	1	0	01	10	00
60	21	1FFF807	73FF33E0	1FFFFF878	7FFFFFFFE1	1	1	1	1	1	10	01	10
61	22	1FFF00F	67FE67C0	1FFFFF0F0	7FFFFFFFC3	1	1	1	1	0	10	10	01
62	23	1FFE01E	4FFCCF80	1FFFFE1E1	7FFFFFFF87	1	1	1	1	0	00	10	10
63	24	1FFC03C	1FF99F00	1FFFFC3C3	7FFFFFFF0F	1	1	1	1	0	11	00	10
64	25	1FF8078	3FF33E01	1FFFF8787	7FFFFFFE1E	1	1	1	1	1	00	11	00
65	26	1FF00F0	7FE67C02	1FFFF0F0F	7FFFFFFC3C	1	1	1	1	0	00	00	11
66	27	1FE01E1	7FCCF805	1FFFE1E1E	7FFFFFF878	1	1	1	1	0	10	00	00
67	28	1FC03C3	7F99F00A	1FFFC3C3C	7FFFFFF0F0	1	1	1	1	0	01	10	00
68	29	1F80787	7F33E015	1FFF87878	7FFFFFE1E1	1	0	1	1	0	10	01	10
69	30	1F00F0F	7E67C02A	1FFF0F0F0	7FFFFFC3C3	1	0	1	1	1	11	10	01
70	31	1E01E1E	7CCF8054	1FFE1E1E1	7FFFFF8787	1	1	1	1	1	01	11	10
71	32	1C03C3C	799F00A9	1FFC3C3C3	7FFFFF0F0F	1	1	1	1	1	01	01	11
72	33	1807878	733E0152	1FF878787	7FFFFE1E1E	1	0	1	1	0	00	01	01
73	34	100F0F0	667C02A5	1FF0F0F0F	7FFFFC3C3C	0	0	1	1	0	10	00	01
74	35	001E1E0	4CF8054B	1FE1E1E1F	7FFFF87878	0	1	1	1	1	00	10	00
75	36	003C3C1	19F00A96	1FC3C3C3F	7FFFF0F0F0	0	1	1	1	1	00	00	10
76	37	0078783	33E0152C	1F878787F	7FFFE1E1E1	0	1	1	1	1	01	00	00
77	38	00F0F07	67C02A59	1F0F0F0FF	7FFFC3C3C3	0	1	1	1	0	11	01	00
78	39	01E1E0E	4F8054B3	1E1E1E1FF	7FFF878787	0	1	1	1	0	11	11	01
79	40	03C3C1C	1F00A966	1C3C3C3FF	7FFF0F0F0F	0	0	1	1	1	11	11	11
80	41	0787838	3E0152CC	1878787FF	7FFE1E1E1E	0	0	1	1	1	11	11	11
81	42	0F0F070	7C02A598	10F0F0FFF	7FFC3C3C3C	1	0	0	1	1	11	11	11
82	43	1E1E0E0	78054B30	01E1E1FFF	7FF8787878	1	0	0	1	1	11	11	11
83	44	1C3C1C0	700A9660	03C3C3FFE	7FF0F0F0F0	1	0	0	1	1	11	11	11
84	45	1878380	60152CC0	078787FFC	7FE1E1E1E0	1	0	0	1	1	11	11	11
85	46	10F0700	402A5980	0F0F0FFF8	7FC3C3C3C0	0	0	1	1	1	11	11	11
86	47	01E0E00	0054B300	1E1E1FFF0	7F87878780	0	0	1	1	1	11	11	11
87	48	03C1C00	00A96601	1C3C3FFE0	7F0F0F0F00	0	1	1	0	1	11	11	11
88	49	0783800	0152CC03	18787FFC0	7E1E1E1E01	0	0	1	0	0	11	11	11

Sample Data

89	50	0F07000	02A59806	10F0FFF80	7C3C3C3C03	1	1	0	0	1	11	11	11
90	51	1E0E000	054B300D	01E1FFF00	7878787807	1	0	0	0	0	11	11	11
91	52	1C1C001	0A96601A	03C3FFE01	70F0F0F00F	1	1	0	1	0	10	11	11
92	53	1838003	152CC035	0787FFC03	61E1E1E01E	1	0	0	1	0	10	10	11
93	54	1070007	2A59806B	0F0FFF807	43C3C3C03C	0	0	1	1	0	01	10	10
94	55	00E000F	54B300D7	1E1FFF00F	0787878078	0	1	1	1	0	10	01	10
95	56	01C001F	296601AE	1C3FFE01F	0F0F0F00F1	0	0	1	0	1	00	10	01
96	57	038003F	52CC035C	187FFC03F	1E1E1E01E2	0	1	1	0	0	00	00	10
97	58	070007F	259806B8	10FFF807F	3C3C3C03C4	0	1	0	0	1	00	00	00
98	59	0E000FE	4B300D71	01FFF00FE	7878780788	1	0	0	0	1	00	00	00
99	60	1C001FD	16601AE2	03FFE01FD	70F0F00F10	1	0	0	1	0	01	00	00
100	61	18003FA	2CC035C5	07FFC03FB	61E1E01E21	1	1	0	1	0	11	01	00
101	62	10007F4	59806B8B	0FFF807F7	43C3C03C43	0	1	1	1	0	11	11	01
102	63	0000FE8	3300D717	1FFF00FEE	0787807887	0	0	1	1	1	11	11	11
103	64	0001FD0	6601AE2F	1FFE01FDC	0F0F00F10E	0	0	1	0	0	11	11	11
104	65	0003FA0	4C035C5F	1FFC03FB8	1E1E01E21D	0	0	1	0	0	11	11	11
105	66	0007F40	1806B8BE	1FF807F70	3C3C03C43B	0	0	1	0	0	11	11	11
106	67	000FE81	300D717C	1FF00FEE1	7878078877	0	0	1	0	0	11	11	11
107	68	001FD02	601AE2F8	1FE01FDC2	70F00F10EF	0	0	1	1	1	11	11	11
108	69	003FA05	4035C5F0	1FC03FB84	61E01E21DE	0	0	1	1	1	11	11	11
109	70	007F40B	006B8BE0	1F807F708	43C03C43BC	0	0	1	1	1	11	11	11
110	71	00FE816	00D717C0	1F00FEE11	0780788778	0	1	1	1	0	10	11	11
111	72	01FD02C	01AE2F81	1E01FDC23	0F00F10EF1	0	1	1	0	0	10	10	11
112	73	03FA059	035C5F02	1C03FB847	1E01E21DE3	0	0	1	0	1	10	10	10
113	74	07F40B3	06B8BE05	1807F708F	3C03C43BC7	0	1	1	0	0	01	10	10
114	75	0FE8166	0D717C0B	100FEE11E	780788778F	1	0	0	0	0	01	01	10
115	76	1FD02CD	1AE2F817	001FDC23D	700F10EF1F	1	1	0	0	1	11	01	01
116	77	1FA059B	35C5F02F	003FB847A	601E21DE3F	1	1	0	0	1	10	11	01
117	78	1F40B37	6B8BE05E	007F708F4	403C43BC7F	1	1	0	0	0	10	10	11
118	79	1E8166E	5717C0BD	00FEE11E9	00788778FF	1	0	0	0	1	10	10	10
119	80	1D02CDC	2E2F817A	01FDC23D3	00F10EF1FE	1	0	0	1	0	01	10	10
120	81	1A059B9	5C5F02F5	03FB847A6	01E21DE3FD	1	0	0	1	1	01	01	10
121	82	140B373	38BE05EB	07F708F4C	03C43BC7FB	0	1	0	1	1	11	01	01
122	83	08166E7	717C0BD7	0FEE11E98	0788778FF7	1	0	1	1	0	11	11	01
123	84	102CDCF	62F817AE	1FDC23D31	0F10EF1FEF	0	1	1	0	1	11	11	11
124	85	0059B9F	45F02F5C	1FB847A63	1E21DE3FDE	0	1	1	0	1	11	11	11
125	86	00B373E	0BE05EB9	1F708F4C7	3C43BC7FBC	0	1	1	0	1	11	11	11
126	87	0166E7D	17C0BD72	1EE11E98F	788778FF78	0	1	1	1	0	10	11	11
127	88	02CDCFB	2F817AE5	1DC23D31F	710EF1FEF1	0	1	1	0	0	10	10	11
128	89	059B9F7	5F02F5CA	1B847A63F	621DE3FDE2	0	0	1	0	1	10	10	10
129	90	0B373EF	3E05EB94	1708F4C7F	443BC7FBC4	1	0	0	0	1	10	10	10
130	91	166E7DF	7C0BD728	0E11E98FF	08778FF788	0	0	1	0	1	10	10	10
131	92	0CDCFBE	7817AE50	1C23D31FF	10EF1FEF10	1	0	1	1	1	01	10	10
132	93	19B9F7D	702F5CA1	1847A63FE	21DE3FDE21	1	0	1	1	0	10	01	10
133	94	1373EFB	605EB942	108F4C7FC	43BC7FBC43	0	0	0	1	1	00	10	01
134	95	06E7DF7	40BD7285	011E98FF8	0778FF7886	0	1	0	0	1	01	00	10
135	96	0DCFBEF	017AE50A	023D31FF0	0EF1FEF10D	1	0	0	1	1	00	01	00
136	97	1B9F7DF	02F5CA15	047A63FE1	1DE3FDE21A	1	1	0	1	1	10	00	01
137	98	173EFBF	05EB942B	08F4C7FC3	3BC7FBC434	0	1	1	1	1	00	10	00
138	99	0E7DF7F	0BD72856	11E98FF87	778FF78869	1	1	0	1	1	00	00	10
139	100	1CFBEFF	17AE50AC	03D31FF0F	6F1FEF10D3	1	1	0	0	0	01	00	00
140	101	19F7DFE	2F5CA159	07A63FE1E	5E3FDE21A7	1	0	0	0	0	00	01	00
141	102	13EFBFC	5EB942B3	0F4C7FC3C	3C7FBC434F	0	1	1	0	0	10	00	01
142	103	07DF7F8	3D728566	1E98FF878	78FF78869F	0	0	1	1	0	00	10	00
143	104	0FBEFF0	7AE50ACD	1D31FF0F0	71FEF10D3E	1	1	1	1	0	11	00	10
144	105	1F7DFE1	75CA159B	1A63FE1E1	63FDE21A7D	1	1	1	1	1	00	11	00
145	106	1EFBFC3	6B942B36	14C7FC3C3	47FBC434FB	1	1	0	1	1	11	00	11

Sample Data

146	107	1DF7F86	5728566D	098FF8786	0FF78869F7	1	0	1	1	0	00	11	00
147	108	1BEFF0C	2E50ACDB	131FF0F0C	1FEF10D3EF	1	0	0	1	0	11	00	11
148	109	17DFE19	5CA159B6	063FE1E19	3FDE21A7DF	0	1	0	1	1	01	11	00
149	110	0FBFC33	3942B36D	0C7FC3C32	7FBC434FBF	1	0	1	1	0	01	01	11
150	111	1F7F866	728566DB	18FF87865	7F78869F7E	1	1	1	0	0	00	01	01
151	112	1EFF0CC	650ACDB6	11FF0F0CB	7EF10D3EFC	1	0	0	1	0	10	00	01
152	113	1DFE199	4A159B6D	03FE1E196	7DE21A7DF9	1	0	0	1	0	00	10	00
153	114	1BFC333	142B36DB	07FC3C32C	7BC434FBF3	1	0	0	1	0	00	00	10
154	115	17F8666	28566DB6	0FF878659	778869F7E6	0	0	1	1	0	01	00	00
155	116	0FF0CCC	50ACDB6D	1FF0F0CB3	6F10D3EFCC	1	1	1	0	0	11	01	00
156	117	1FE1999	2159B6DA	1FE1E1966	5E21A7DF99	1	0	1	0	1	10	11	01
157	118	1FC3332	42B36DB5	1FC3C32CC	3C434FBF33	1	1	1	0	1	10	10	11
158	119	1F86664	0566DB6B	1F8786599	78869F7E67	1	0	1	1	1	01	10	10
159	120	1F0CCC8	0ACDB6D6	1F0F0CB33	710D3EFCCE	1	1	1	0	0	10	01	10
160	121	1E19991	159B6DAC	1E1E19666	621A7DF99D	1	1	1	0	1	11	10	01
161	122	1C33323	2B36DB58	1C3C32CCC	4434FBF33B	1	0	1	0	1	00	11	10
162	123	1866647	566DB6B0	187865999	0869F7E676	1	0	1	0	0	11	00	11
163	124	10CCC8F	2CDB6D60	10F0CB333	10D3EFCCEC	0	1	0	1	1	01	11	00
164	125	019991E	59B6DAC0	01E196666	21A7DF99D9	0	1	0	1	1	10	01	11
165	126	033323C	336DB580	03C32CCCD	434FBF33B3	0	0	0	0	0	00	10	01
166	127	0666478	66DB6B01	07865999A	069F7E6766	0	1	0	1	0	00	00	10
167	128	0CCC8F0	4DB6D603	0F0CB3334	0D3EFCCECD	1	1	1	0	1	01	00	00
168	129	19991E1	1B6DAC07	1E1966669	1A7DF99D9B	1	0	1	0	1	00	01	00
169	130	13323C3	36DB580E	1C32CCCD3	34FBF33B37	0	1	1	1	1	10	00	01
170	131	0664786	6DB6B01C	1865999A7	69F7E6766F	0	1	1	1	1	00	10	00
171	132	0CC8F0D	5B6D6039	10CB3334F	53EFCCECDF	1	0	0	1	0	00	00	10
172	133	1991E1A	36DAC073	01966669E	27DF99D9BF	1	1	0	1	1	01	00	00
173	134	1323C35	6DB580E6	032CCCD3C	4FBF33B37E	0	1	0	1	1	00	01	00
174	135	064786A	5B6B01CD	065999A78	1F7E6766FC	0	0	0	0	0	11	00	01
175	136	0C8F0D5	36D6039B	0CB3334F0	3EFCCECDF9	1	1	1	1	1	00	11	00
176	137	191E1AA	6DAC0737	1966669E1	7DF99D9BF3	1	1	1	1	0	00	00	11
177	138	123C354	5B580E6E	12CCCD3C3	7BF33B37E7	0	0	0	1	1	00	00	00
178	139	04786A9	36B01CDC	05999A787	77E6766FCE	0	1	0	1	0	01	00	00
179	140	08F0D53	6D6039B8	0B3334F0E	6FCCECDF9C	1	0	1	1	0	11	01	00
180	141	11E1AA6	5AC07370	166669E1D	5F99D9BF38	0	1	0	1	1	10	11	01
181	142	03C354C	3580E6E0	0CCCD3C3A	3F33B37E70	0	1	1	0	0	10	10	11
182	143	0786A99	6B01CDC0	1999A7875	7E6766FCE1	0	0	1	0	1	10	10	10
183	144	0F0D533	56039B81	13334F0EB	7CCECDF9C2	1	0	0	1	0	01	10	10
184	145	1E1AA66	2C073703	06669E1D6	799D9BF385	1	0	0	1	1	01	01	10
185	146	1C354CC	580E6E06	0CCD3C3AC	733B37E70B	1	0	1	0	1	11	01	01
186	147	186A998	301CDC0C	199A78759	66766FCE17	1	0	1	0	1	10	11	01
187	148	10D5331	6039B818	1334F0EB2	4CECDF9C2F	0	0	0	1	1	01	10	11
188	149	01AA662	40737031	0669E1D65	19D9BF385E	0	0	0	1	0	01	01	10
189	150	0354CC5	00E6E063	0CD3C3ACB	33B37E70BD	0	1	1	1	0	00	01	01
190	151	06A998A	01CDC0C6	19A787596	6766FCE17B	0	1	1	0	0	10	00	01
191	152	0D53315	039B818C	134F0EB2C	4ECDF9C2F6	1	1	0	1	1	00	10	00
192	153	1AA662A	07370318	069E1D659	1D9BF385ED	1	0	0	1	0	00	00	10
193	154	154CC54	0E6E0630	0D3C3ACB3	3B37E70BDB	0	0	1	0	1	00	00	00
194	155	0A998A8	1CDC0C60	1A7875967	766FCE17B6	1	1	1	0	1	01	00	00
195	156	1533151	39B818C0	14F0EB2CE	6CDF9C2F6C	0	1	0	1	1	00	01	00
196	157	0A662A3	73703180	09E1D659D	59BF385ED8	1	0	1	1	1	10	00	01
197	158	14CC547	66E06301	13C3ACB3A	337E70BDB0	0	1	0	0	1	11	10	00
198	159	0998A8E	4DC0C602	078759675	66FCE17B61	1	1	0	1	0	01	11	10
199	160	133151D	1B818C05	0F0EB2CEB	4DF9C2F6C2	0	1	1	1	0	01	01	11
200	161	0662A3B	3703180B	1E1D659D6	1BF385ED85	0	0	1	1	1	11	01	01
201	162	0CC5477	6E063017	1C3ACB3AC	37E70BDB0B	1	0	1	1	0	11	11	01
202	163	198A8EF	5C0C602F	187596759	6FCE17B617	1	0	1	1	0	10	11	11

Sample Data

203	164	13151DE	3818C05F	10EB2CEB2	5F9C2F6C2F	0	0	0	1	1	01	10	11
204	165	062A3BC	703180BF	01D659D65	3F385ED85E	0	0	0	0	1	00	01	10
205	166	0C54779	6063017E	03ACB3ACB	7E70BDB0BD	1	0	0	0	1	11	00	01
206	167	18A8EF2	40C602FD	075967597	7CE17B617B	1	1	0	1	0	00	11	00
207	168	1151DE4	018C05FA	0EB2CEB2F	79C2F6C2F7	0	1	1	1	1	11	00	11
208	169	02A3BC9	03180BF5	1D659D65E	7385ED85EE	0	0	1	1	1	01	11	00
209	170	0547793	063017EB	1ACB3ACBC	670BDB0BDC	0	0	1	0	0	10	01	11
210	171	0A8EF27	0C602FD6	159675978	4E17B617B9	1	0	0	0	1	00	10	01
211	172	151DE4E	18C05FAD	0B2CEB2F1	1C2F6C2F73	0	1	1	0	0	00	00	10
212	173	0A3BC9C	3180BF5A	1659D65E3	385ED85EE6	1	1	0	0	0	01	00	00
213	174	1477938	63017EB5	0CB3ACBC6	70BDB0BDCC	0	0	1	1	1	00	01	00
214	175	08EF270	4602FD6A	19675978D	617B617B99	1	0	1	0	0	10	00	01
215	176	11DE4E1	0C05FAD5	12CEB2F1A	42F6C2F733	0	0	0	1	1	11	10	00
216	177	03BC9C3	180BF5AA	059D65E34	05ED85EE67	0	0	0	1	0	00	11	10
217	178	0779387	3017EB55	0B3ACBC68	0BDB0BDCCF	0	0	1	1	0	11	00	11
218	179	0EF270F	602FD6AA	1675978D0	17B617B99F	1	0	0	1	1	01	11	00
219	180	1DE4E1F	405FAD54	0CEB2F1A1	2F6C2F733F	1	0	1	0	1	10	01	11
220	181	1BC9C3F	00BF5AA9	19D65E342	5ED85EE67F	1	1	1	1	0	10	10	01
221	182	179387F	017EB552	13ACBC684	3DB0BDCCFE	0	0	0	1	1	10	10	10
222	183	0F270FF	02FD6AA5	075978D09	7B617B99FC	1	1	0	0	0	01	10	10
223	184	1E4E1FF	05FAD54A	0EB2F1A12	76C2F733F9	1	1	1	1	1	10	01	10
224	185	1C9C3FE	0BF5AA94	1D65E3425	6D85EE67F2	1	1	1	1	0	10	10	01
225	186	19387FD	17EB5529	1ACBC684B	5B0BDCCFE4	1	1	1	0	1	01	10	10
226	187	1270FFA	2FD6AA53	15978D096	3617B99FC9	0	1	0	0	0	01	01	10
227	188	04E1FF5	5FAD54A7	0B2F1A12C	6C2F733F93	0	1	1	0	1	11	01	01
228	189	09C3FEB	3F5AA94E	165E34258	585EE67F27	1	0	0	0	0	10	11	01
229	190	1387FD7	7EB5529C	0CBC684B1	30BDCCFE4F	0	1	1	1	1	10	10	11
230	191	070FFAE	7D6AA538	1978D0962	617B99FC9E	0	0	1	0	1	10	10	10
231	192	0E1FF5C	7AD54A70	12F1A12C4	42F733F93D	1	1	0	1	1	01	10	10
232	193	1C3FEB9	75AA94E1	05E342588	05EE67F27A	1	1	0	1	0	10	01	10
233	194	187FD73	6B5529C3	0BC684B10	0BDCCFE4F4	1	0	1	1	1	11	10	01
234	195	10FFAE6	56AA5386	178D09621	17B99FC9E8	0	1	0	1	1	00	11	10
235	196	01FF5CC	2D54A70C	0F1A12C43	2F733F93D0	0	0	1	0	1	10	00	11
236	197	03FEB98	5AA94E19	1E3425887	5EE67F27A1	0	1	1	1	1	00	10	00
237	198	07FD731	35529C33	1C684B10F	3DCCFE4F42	0	0	1	1	0	00	00	10
238	199	0FFAE63	6AA53866	18D09621F	7B99FC9E84	1	1	1	1	0	10	00	00
239	200	1FF5CC6	554A70CD	11A12C43F	7733F93D09	1	0	0	0	1	11	10	00

```
Z[0]  = 59
Z[1]  = 3B
Z[2]  = EF
Z[3]  = 07
Z[4]  = 13
Z[5]  = 70
Z[6]  = 9B
Z[7]  = B7
Z[8]  = 52
Z[9]  = 8F
Z[10] = 3E
Z[11] = B9
Z[12] = A5
Z[13] = AC
Z[14] = EA
Z[15] = 9E
```

Sample Data

```
================================================================================================
         Reload this pattern into the LFSRs
         Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 1521359
LFSR2  <= 528F703B
LFSR3  <= 0AC3E9BEF
LFSR4  <= 4FEAB9B707
C[t+1] <= 00

================================================================================================
         Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
```

240	1	1521359	528F703B	0AC3E9BEF	4FEAB9B707	0	1	1	1	1	00	10	00
241	2	0A426B3	251EE076	1587D37DE	1FD5736E0F	1	0	0	1	0	00	00	10
242	3	1484D67	4A3DC0ED	0B0FA6FBD	3FAAE6DC1E	0	0	1	1	0	01	00	00
243	4	0909ACF	147B81DA	161F4DF7A	7F55CDB83D	1	0	0	0	0	00	01	00
244	5	121359E	28F703B5	0C3E9BEF5	7EAB9B707B	0	1	1	1	1	10	00	01
245	6	0426B3C	51EE076B	187D37DEB	7D5736E0F6	0	1	1	0	0	00	10	00
246	7	084D679	23DC0ED6	10FA6FBD7	7AAE6DC1EC	1	1	0	1	1	00	00	10
247	8	109ACF2	47B81DAC	01F4DF7AF	755CDB83D8	0	1	0	0	1	00	00	00
248	9	01359E4	0F703B59	03E9BEF5E	6AB9B707B1	0	0	0	1	1	00	00	00
249	10	026B3C8	1EE076B3	07D37DEBD	55736E0F63	0	1	0	0	1	00	00	00
250	11	04D6791	3DC0ED67	0FA6FBD7A	2AE6DC1EC7	0	1	1	1	1	01	00	00
251	12	09ACF22	7B81DACF	1F4DF7AF4	55CDB83D8F	1	1	1	1	1	11	01	00
252	13	1359E44	7703B59E	1E9BEF5E8	2B9B707B1F	0	0	1	1	1	10	11	01
253	14	06B3C88	6E076B3C	1D37DEBD0	5736E0F63F	0	0	1	0	1	01	10	11
254	15	0D67911	5C0ED678	1A6FBD7A1	2E6DC1EC7E	1	0	1	0	1	01	01	10
255	16	1ACF223	381DACF0	14DF7AF42	5CDB83D8FD	1	0	0	1	1	11	01	01
256	17	159E446	703B59E0	09BEF5E85	39B707B1FA	0	0	1	1	1	10	11	01
257	18	0B3C88C	6076B3C0	137DEBD0A	736E0F63F4	1	0	0	0	1	01	10	11
258	19	1679118	40ED6780	06FBD7A15	66DC1EC7E8	0	1	0	1	1	01	01	10
259	20	0CF2231	01DACF00	0DF7AF42A	4DB83D8FD1	1	1	1	1	1	00	01	01
260	21	19E4463	03B59E01	1BEF5E854	1B707B1FA3	1	1	1	0	1	10	00	01
261	22	13C88C6	076B3C03	17DEBD0A9	36E0F63F47	0	0	0	1	1	11	10	00
262	23	079118C	0ED67807	0FBD7A152	6DC1EC7E8E	0	1	1	1	0	01	11	10
263	24	0F22318	1DACF00E	1F7AF42A4	5B83D8FD1D	1	1	1	1	1	01	01	11
264	25	1E44630	3B59E01C	1EF5E8548	3707B1FA3B	1	0	1	0	1	11	01	01
265	26	1C88C61	76B3C039	1DEBD0A91	6E0F63F477	1	1	1	0	0	11	11	01
266	27	19118C3	6D678073	1BD7A1523	5C1EC7E8EF	1	0	1	0	1	11	11	11
267	28	1223187	5ACF00E6	17AF42A46	383D8FD1DE	0	1	0	0	0	11	11	11
268	29	044630E	359E01CC	0F5E8548D	707B1FA3BD	0	1	1	0	1	11	11	11
269	30	088C61C	6B3C0399	1EBD0A91A	60F63F477B	1	0	1	1	0	10	11	11
270	31	1118C39	56780733	1D7A15234	41EC7E8EF6	0	0	1	1	0	10	10	11
271	32	0231872	2CF00E67	1AF42A468	03D8FD1DEC	0	1	1	1	1	01	10	10
272	33	04630E5	59E01CCE	15E8548D1	07B1FA3BD8	0	1	0	1	1	01	01	10
273	34	08C61CB	33C0399D	0BD0A91A3	0F63F477B1	1	1	1	0	0	00	01	01
274	35	118C396	6780733A	17A152347	1EC7E8EF63	0	1	0	1	0	10	00	01
275	36	031872D	4F00E674	0F42A468E	3D8FD1DEC7	0	0	1	1	0	00	10	00
276	37	0630E5A	1E01CCE8	1E8548D1D	7B1FA3BD8E	0	0	1	0	1	01	00	10
277	38	0C61CB5	3C0399D0	1D0A91A3B	763F477B1C	1	0	1	0	1	00	01	00
278	39	18C396A	780733A0	1A1523477	6C7E8EF639	1	0	1	0	0	10	00	01
279	40	11872D5	700E6741	142A468EF	58FD1DEC72	0	0	0	1	1	11	10	00
280	41	030E5AB	601CCE83	08548D1DF	31FA3BD8E5	0	0	1	1	1	00	11	10
281	42	061CB57	40399D07	10A91A3BF	63F477B1CB	0	0	0	1	1	10	00	11
282	43	0C396AF	00733A0F	01523477E	47E8EF6396	1	0	0	1	0	00	10	00
283	44	1872D5F	00E6741F	02A468EFD	0FD1DEC72C	1	1	0	1	1	00	00	10

Sample Data

284	45	10E5ABE	01CCE83F	0548D1DFA	1FA3BD8E58	0	1	0	1	0	01	00	00
285	46	01CB57C	0399D07F	0A91A3BF4	3F477B1CB0	0	1	1	0	1	00	01	00
286	47	0396AF9	0733A0FE	1523477E9	7E8EF63961	0	0	0	1	1	11	00	01
287	48	072D5F3	0E6741FD	0A468EFD2	7D1DEC72C3	0	0	1	0	0	01	11	00
288	49	0E5ABE7	1CCE83FA	148D1DFA4	7A3BD8E587	1	1	0	0	1	10	01	11
289	50	1CB57CE	399D07F4	091A3BF49	7477B1CB0F	1	1	1	0	1	11	10	01
290	51	196AF9D	733A0FE9	123477E92	68EF63961E	1	0	0	1	1	00	11	10
291	52	12D5F3B	66741FD2	0468EFD25	51DEC72C3C	0	0	0	1	1	10	00	11
292	53	05ABE77	4CE83FA4	08D1DFA4B	23BD8E5879	0	1	1	1	1	00	10	00
293	54	0B57CEE	19D07F49	11A3BF496	477B1CB0F2	1	1	0	0	0	00	00	10
294	55	16AF9DC	33A0FE92	03477E92C	0EF63961E4	0	1	0	1	0	01	00	00
295	56	0D5F3B8	6741FD25	068EFD259	1DEC72C3C9	1	0	0	1	1	00	01	00
296	57	1ABE771	4E83FA4B	0D1DFA4B3	3BD8E58793	1	1	1	1	0	01	00	01
297	58	157CEE2	1D07F496	1A3BF4967	77B1CB0F26	0	0	1	1	1	00	01	00
298	59	0AF9DC5	3A0FE92D	1477E92CE	6F63961E4D	1	0	0	0	1	11	00	01
299	60	15F3B8B	741FD25A	08EFD259C	5EC72C3C9B	0	0	1	1	1	01	11	00
300	61	0BE7716	683FA4B4	11DFA4B39	3D8E587937	1	0	0	1	1	10	01	11
301	62	17CEE2D	507F4968	03BF49672	7B1CB0F26E	0	0	0	0	0	00	10	01
302	63	0F9DC5B	20FE92D0	077E92CE4	763961E4DC	1	1	0	0	0	00	00	10
303	64	1F3B8B6	41FD25A0	0EFD259C9	6C72C3C9B9	1	1	1	0	1	01	00	00
304	65	1E7716D	03FA4B40	1DFA4B393	58E5879373	1	1	1	1	1	11	01	00
305	66	1CEE2DB	07F49680	1BF496727	31CB0F26E6	1	1	1	1	1	11	11	01
306	67	19DC5B7	0FE92D00	17E92CE4E	63961E4DCD	1	1	0	1	0	10	11	11
307	68	13B8B6F	1FD25A00	0FD259C9C	472C3C9B9A	0	1	1	0	0	10	10	11
308	69	07716DF	3FA4B400	1FA4B3938	0E58793735	0	1	1	0	0	01	10	10
309	70	0EE2DBF	7F496800	1F4967271	1CB0F26E6A	1	0	1	1	0	10	01	10
310	71	1DC5B7F	7E92D000	1E92CE4E2	3961E4DCD4	1	1	1	0	1	11	10	01
311	72	1B8B6FF	7D25A001	1D259C9C4	72C3C9B9A9	1	0	1	1	0	01	11	10
312	73	1716DFF	7A4B4002	1A4B39389	6587937352	0	0	1	1	1	10	01	11
313	74	0E2DBFF	74968005	149672713	4B0F26E6A5	1	1	0	0	0	11	10	01
314	75	1C5B7FE	692D000B	092CE4E26	161E4DCD4B	1	0	1	0	1	00	11	10
315	76	18B6FFC	525A0017	1259C9C4D	2C3C9B9A96	1	0	0	0	1	10	00	11
316	77	116DFF8	24B4002F	04B39389B	587937352C	0	1	0	0	1	11	10	00
317	78	02DBFF1	4968005F	096727136	30F26E6A58	0	0	1	1	1	00	11	10
318	79	05B7FE3	12D000BF	12CE4E26C	61E4DCD4B1	0	1	0	1	0	11	00	11
319	80	0B6FFC7	25A0017F	059C9C4D8	43C9B9A963	1	1	0	1	0	00	11	00
320	81	16DFF8E	4B4002FF	0B39389B1	07937352C6	0	0	1	1	0	11	00	11
321	82	0DBFF1C	168005FF	167271363	0F26E6A58C	1	1	0	0	1	01	11	00
322	83	1B7FE38	2D000BFF	0CE4E26C7	1E4DCD4B18	1	0	1	0	1	10	01	11
323	84	16FFC70	5A0017FF	19C9C4D8F	3C9B9A9631	0	0	1	1	0	11	10	01
324	85	0DFF8E1	34002FFF	139389B1E	7937352C62	1	0	0	0	0	00	11	10
325	86	1BFF1C3	68005FFF	07271363D	726E6A58C4	1	0	0	0	1	10	00	11
326	87	17FE387	5000BFFE	0E4E26C7B	64DCD4B188	0	0	1	1	0	00	10	00
327	88	0FFC70F	20017FFD	1C9C4D8F6	49B9A96311	1	0	1	1	1	00	00	10
328	89	1FF8E1F	4002FFFB	19389B1ED	137352C623	1	0	1	0	0	01	00	00
329	90	1FF1C3F	0005FFF7	1271363DB	26E6A58C46	1	0	0	1	1	00	01	00
330	91	1FE387F	000BFFEE	04E26C7B6	4DCD4B188C	1	0	0	1	0	10	00	01
331	92	1FC70FF	0017FFDC	09C4D8F6D	1B9A963118	1	0	1	1	1	00	10	00
332	93	1F8E1FF	002FFFB8	1389B1EDA	37352C6231	1	0	0	0	1	01	00	10
333	94	1F1C3FF	005FFF70	071363DB4	6E6A58C462	1	0	0	0	0	00	01	00
334	95	1E387FE	00BFFEE0	0E26C7B68	5CD4B188C5	1	1	1	1	0	01	00	01
335	96	1C70FFC	017FFDC1	1C4D8F6D1	39A963118A	1	0	1	1	0	11	01	00
336	97	18E1FF9	02FFFB82	189B1EDA2	7352C62315	1	1	1	0	0	11	11	01
337	98	11C3FF2	05FFF705	11363DB45	66A58C462B	0	1	0	1	1	11	11	11
338	99	0387FE4	0BFFEE0A	026C7B68B	4D4B188C56	0	1	0	0	0	11	11	11
339	100	070FFC9	17FFDC15	04D8F6D16	1A963118AD	0	1	0	1	1	11	11	11
340	101	0E1FF92	2FFFB82B	09B1EDA2C	352C62315A	1	1	1	0	0	10	11	11

Sample Data

341	102	1C3FF24	5FFF7057	1363DB458	6A58C462B4	1	1	0	0	0	10	10	11
342	103	187FE48	3FFEE0AE	06C7B68B0	54B188C569	1	1	0	1	1	01	10	10
343	104	10FFC90	7FFDC15C	0D8F6D161	2963118AD2	0	1	1	0	1	01	01	10
344	105	01FF920	7FFB82B9	1B1EDA2C2	52C62315A5	0	1	1	1	0	00	01	01
345	106	03FF240	7FF70573	163DB4584	258C462B4B	0	1	0	1	0	10	00	01
346	107	07FE481	7FEE0AE6	0C7B68B08	4B188C5696	0	1	1	0	0	00	10	00
347	108	0FFC902	7FDC15CD	18F6D1610	163118AD2D	1	1	1	0	1	00	00	10
348	109	1FF9204	7FB82B9A	11EDA2C20	2C62315A5B	1	1	0	0	0	01	00	00
349	110	1FF2408	7F705735	03DB45841	58C462B4B6	1	0	0	1	1	00	01	00
350	111	1FE4810	7EE0AE6B	07B68B082	3188C5696C	1	1	0	1	1	10	00	01
351	112	1FC9021	7DC15CD6	0F6D16105	63118AD2D8	1	1	1	0	1	00	10	00
352	113	1F92042	7B82B9AD	1EDA2C20B	462315A5B0	1	1	1	0	1	00	00	10
353	114	1F24084	7705735A	1DB458416	0C462B4B61	1	0	1	0	0	01	00	00
354	115	1E48108	6E0AE6B5	1B68B082C	188C5696C3	1	0	1	1	0	11	01	00
355	116	1C90211	5C15CD6A	16D161059	3118AD2D86	1	0	0	0	0	10	11	01
356	117	1920422	382B9AD5	0DA2C20B3	62315A5B0D	1	0	1	0	0	10	10	11
357	118	1240845	705735AA	1B4584167	4462B4B61A	0	0	1	0	1	10	10	10
358	119	048108A	60AE6B55	168B082CF	08C5696C34	0	1	0	1	0	01	10	10
359	120	0902114	415CD6AB	0D161059E	118AD2D869	1	0	1	1	0	10	01	10
360	121	1204228	02B9AD56	1A2C20B3D	2315A5B0D2	0	1	1	0	0	11	10	01
361	122	0408451	05735AAD	14584167B	462B4B61A4	0	0	0	0	1	11	11	10
362	123	08108A2	0AE6B55B	08B082CF7	0C5696C348	1	1	1	0	0	10	11	11
363	124	1021144	15CD6AB6	1161059EF	18AD2D8690	0	1	0	1	0	10	10	11
364	125	0042289	2B9AD56C	02C20B3DE	315A5B0D20	0	1	0	0	1	10	10	10

Sample Data

1.5 FOURTH SET OF SAMPLES

Initial values for the key, pan address and clock

K'c4[0] = 21 K'c4[1] = 87 K'c4[2] = F0 K'c4[3] = 4A
K'c4[4] = BA K'c4[5] = 90 K'c4[6] = 31 K'c4[7] = D0
K'c4[8] = 78 K'c4[9] = 0D K'c4[10] = 4C K'c4[11] = 53
K'c4[12] = E0 K'c4[13] = 15 K'c4[14] = 3A K'c4[15] = 63

Addr4[0] = 2C Addr4[1] = 7F Addr4[2] = 94
Addr4[3] = 56 Addr4[4] = 0F Addr4[5] = 1B

CL4[0] = 5F CL4[1] = 1A CL4[2] = 00 CL4[3] = 02

Fill LFSRs with initial data

t	clk#	LFSR1	LFSR2	LFSR3	LFSR4	X1	X2	X3	X4	Z	C[t+1]	C[t]	C[t-1]
0	0	0000000*	00000000*	000000000*	0000000000*	0	0	0	0	0	00	00	00
1	1	0000000*	00000001*	000000001*	0000000001*	0	0	0	0	0	00	00	00
2	2	0000001*	00000002*	000000002*	0000000003*	0	0	0	0	0	00	00	00
3	3	0000002*	00000004*	000000004*	0000000007*	0	0	0	0	0	00	00	00
4	4	0000004*	00000009*	000000008*	000000000F*	0	0	0	0	0	00	00	00
5	5	0000008*	00000013*	000000010*	000000001E*	0	0	0	0	0	00	00	00
6	6	0000010*	00000027*	000000021*	000000003D*	0	0	0	0	0	00	00	00
7	7	0000021*	0000004F*	000000043*	000000007A*	0	0	0	0	0	00	00	00
8	8	0000042*	0000009F*	000000087*	00000000F4*	0	0	0	0	0	00	00	00
9	9	0000084*	0000013F*	00000010F*	00000001E9*	0	0	0	0	0	00	00	00
10	10	0000108*	0000027F*	00000021F*	00000003D2*	0	0	0	0	0	00	00	00
11	11	0000211*	000004FE*	00000043E*	00000007A5*	0	0	0	0	0	00	00	00
12	12	0000422*	000009FC*	00000087C*	0000000F4A*	0	0	0	0	0	00	00	00
13	13	0000845*	000013F8*	0000010F8*	0000001E94*	0	0	0	0	0	00	00	00
14	14	000108B*	000027F0*	0000021F1*	0000003D29*	0	0	0	0	0	00	00	00
15	15	0002117*	00004FE1*	0000043E3*	0000007A52*	0	0	0	0	0	00	00	00
16	16	000422E*	00009FC2*	0000087C6*	000000F4A4*	0	0	0	0	0	00	00	00
17	17	000845D*	00013F84*	000010F8C*	000001E948*	0	0	0	0	0	00	00	00
18	18	00108BA*	00027F08*	000021F18*	000003D290*	0	0	0	0	0	00	00	00
19	19	0021174*	0004FE10*	000043E30*	000007A520*	0	0	0	0	0	00	00	00
20	20	00422E8*	0009FC21*	000087C61*	00000F4A41*	0	0	0	0	0	00	00	00
21	21	00845D1*	0013F842*	00010F8C3*	00001E9482*	0	0	0	0	0	00	00	00
22	22	0108BA3*	0027F084*	00021F186*	00003D2905*	0	0	0	0	0	00	00	00
23	23	0211747*	004FE109*	00043E30C*	00007A520B*	0	0	0	0	0	00	00	00
24	24	0422E8F*	009FC213*	00087C619*	0000F4A417*	0	1	0	0	1	00	00	00
25	25	0845D1E*	013F8426*	0010F8C32*	0001E9482F*	1	0	0	0	1	00	00	00
26	26	108BA3D	027F084D*	0021F1864*	0003D2905E*	0	0	0	0	0	00	00	00
27	27	011747B	04FE109B*	0043E30C9*	0007A520BC*	0	1	0	0	1	00	00	00
28	28	022E8F6	09FC2136*	0087C6192*	000F4A4179*	0	1	0	0	1	00	00	00
29	29	045D1EC	13F8426C*	010F8C325*	001E9482F2*	0	1	0	0	1	00	00	00
30	30	08BA3D9	27F084D8*	021F1864B*	003D2905E5*	1	1	0	0	0	01	00	00
31	31	11747B3	4FE109B0*	043E30C97*	007A520BCA*	0	1	0	0	1	00	00	00
32	32	02E8F67	1FC21360	087C6192E*	00F4A41795*	0	1	1	1	1	01	00	00
33	33	05D1ECF	3F8426C1	10F8C325C*	01E9482F2B*	0	1	0	1	0	01	00	00
34	34	0BA3D9F	7F084D82	01F1864B8	03D2905E56*	1	0	0	1	0	01	00	00

Sample Data

35	35	1747B3E	7E109B04	03E30C970	07A520BCAC*	0	0	0	1	1	00	00	00
36	36	0E8F67C	7C213608	07C6192E1	0F4A417958*	1	0	0	0	1	00	00	00
37	37	1D1ECF8	78426C11	0F8C325C3	1E9482F2B1*	1	0	1	1	1	01	00	00
38	38	1A3D9F0	7084D822	1F1864B86	3D2905E563*	1	1	1	0	1	01	00	00
39	39	147B3E1	6109B044	1E30C970C	7A520BCAC6*	0	0	1	0	1	00	00	00

Start clocking Summation Combiner

40	1	08F67C2	42136088	1C6192E18	74A417958D	1	0	1	1	1	01	00	00
41	2	11ECF84	0426C111	18C325C30	69482F2B1B	0	0	1	0	0	00	01	00
42	3	03D9F08	084D8222	11864B861	52905E5637	0	0	0	1	1	11	00	01
43	4	07B3E10	109B0444	030C970C3	2520BCAC6E	0	1	0	0	0	01	11	00
44	5	0F67C21	21360889	06192E186	4A417958DC	1	0	0	0	0	10	01	11
45	6	1ECF843	426C1112	0C325C30C	1482F2B1B8	1	0	1	1	1	11	10	01
46	7	1D9F086	04D82225	1864B8619	2905E56370	1	1	1	0	0	01	11	10
47	8	1B3E10D	09B0444B	10C970C32	520BCAC6E1	1	1	0	0	1	10	01	11
48	9	167C21B	13608897	0192E1865	2417958DC3	0	0	0	0	0	00	10	01
49	10	0CF8436	26C1112F	0325C30CB	482F2B1B87	1	1	0	0	0	00	00	10
50	11	19F086D	4D82225E	064B86197	105E56370F	1	1	0	0	0	01	00	00
51	12	13E10DB	1B0444BC	0C970C32F	20BCAC6E1F	0	0	1	1	1	00	01	00
52	13	07C21B7	36088979	192E1865E	417958DC3F	0	0	1	0	1	11	00	01
53	14	0F8436E	6C1112F2	125C30CBD	02F2B1B87F	1	0	0	1	1	01	11	00
54	15	1F086DD	582225E4	04B86197B	05E56370FF	1	0	0	1	1	10	01	11
55	16	1E10DBA	30444BC9	0970C32F7	0BCAC6E1FF	1	0	1	1	1	11	10	01
56	17	1C21B75	60889793	12E1865EE	17958DC3FF	1	1	0	1	0	01	11	10
57	18	18436EA	41112F27	05C30CBDD	2F2B1B87FF	1	0	0	0	0	10	01	11
58	19	1086DD4	02225E4E	0B86197BA	5E56370FFF	0	0	1	0	1	00	10	01
59	20	010DBA8	0444BC9D	170C32F74	3CAC6E1FFF	0	0	0	1	1	01	00	10
60	21	021B750	0889793A	0E1865EE8	7958DC3FFF	0	1	1	0	1	00	01	00
61	22	0436EA0	1112F274	1C30CBDD0	72B1B87FFE	0	0	1	1	0	10	00	01
62	23	086DD40	2225E4E9	186197BA1	656370FFFC	1	0	1	0	0	00	10	00
63	24	10DBA81	444BC9D3	10C32F743	4AC6E1FFF8	0	0	0	1	1	01	00	10
64	25	01B7502	089793A7	01865EE86	158DC3FFF1	0	1	0	1	1	00	01	00
65	26	036EA05	112F274E	030CBDD0D	2B1B87FFE3	0	0	0	0	0	11	00	01
66	27	06DD40B	225E4E9C	06197BA1A	56370FFFC6	0	0	0	0	1	10	11	00
67	28	0DBA817	44BC9D39	0C32F7434	2C6E1FFF8D	1	1	1	0	1	10	10	11
68	29	1B7502E	09793A72	1865EE868	58DC3FFF1B	1	0	1	1	1	01	10	10
69	30	16EA05D	12F274E5	10CBDD0D0	31B87FFE36	0	1	0	1	1	01	01	10
70	31	0DD40BA	25E4E9CB	0197BA1A1	6370FFFC6D	1	1	0	0	1	11	01	01
71	32	1BA8174	4BC9D397	032F74343	46E1FFF8DA	1	1	0	1	0	11	11	01
72	33	17502E8	1793A72F	065EE8687	0DC3FFF1B4	0	1	0	1	1	11	11	11
73	34	0EA05D0	2F274E5E	0CBDD0D0F	1B87FFE369	1	0	1	1	0	10	11	11
74	35	1D40BA0	5E4E9CBD	197BA1A1F	370FFFC6D2	1	0	1	0	0	10	10	11
75	36	1A81741	3C9D397B	12F74343F	6E1FFF8DA5	1	1	0	0	0	01	10	10
76	37	1502E82	793A72F6	05EE8687F	5C3FFF1B4B	0	0	0	0	1	00	01	10
77	38	0A05D05	7274E5ED	0BDD0D0FF	387FFE3696	1	0	1	0	0	10	00	01
78	39	140BA0B	64E9CBDA	17BA1A1FF	70FFFC6D2C	0	1	0	1	0	00	10	00
79	40	0817416	49D397B4	0F74343FE	61FFF8DA59	1	1	1	1	0	11	00	10
80	41	102E82C	13A72F69	1EE8687FD	43FFF1B4B3	0	1	1	1	0	00	11	00
81	42	005D058	274E5ED2	1DD0D0FFA	07FFE36966	0	0	1	1	0	11	00	11
82	43	00BA0B0	4E9CBDA5	1BA1A1FF5	0FFFC6D2CD	0	1	1	1	0	00	11	00
83	44	0174160	1D397B4A	174343FEA	1FFF8DA59B	0	0	0	1	1	10	00	11
84	45	02E82C0	3A72F695	0E8687FD4	3FFF1B4B37	0	0	1	1	0	00	10	00
85	46	05D0580	74E5ED2B	1D0D0FFA9	7FFE36966E	0	1	1	1	1	00	00	10
86	47	0BA0B00	69CBDA56	1A1A1FF53	7FFC6D2CDC	1	1	1	1	0	10	00	00
87	48	1741600	5397B4AC	14343FEA6	7FF8DA59B8	0	1	0	1	0	00	10	00
88	49	0E82C01	272F6959	08687FD4D	7FF1B4B370	1	0	1	1	1	00	00	10

Sample Data

89	50	1D05802	4E5ED2B3	10D0FFA9A	7FE36966E0	1	0	0	1	0	01	00	00
90	51	1A0B004	1CBDA566	01A1FF535	7FC6D2CDC0	1	1	0	1	0	11	01	00
91	52	1416009	397B4ACC	0343FEA6B	7F8DA59B80	0	0	0	1	0	10	11	01
92	53	082C013	72F69599	0687FD4D7	7F1B4B3701	1	1	0	0	0	10	10	11
93	54	1058026	65ED2B33	0D0FFA9AF	7E36966E03	0	1	1	0	0	01	10	10
94	55	00B004D	4BDA5667	1A1FF535E	7C6D2CDC06	0	1	1	0	1	01	01	10
95	56	016009B	17B4ACCE	143FEA6BD	78DA59B80D	0	1	0	1	1	11	01	01
96	57	02C0137	2F69599D	087FD4D7B	71B4B3701A	0	0	1	1	1	10	11	01
97	58	058026F	5ED2B33B	10FFA9AF6	636966E034	0	1	0	0	1	01	10	11
98	59	0B004DF	3DA56677	01FF535ED	46D2CDC068	1	1	0	1	0	10	01	10
99	60	16009BF	7B4ACCEF	03FEA6BDB	0DA59B80D0	0	0	0	1	1	00	10	01
100	61	0C0137F	769599DF	07FD4D7B7	1B4B3701A1	1	1	0	0	0	00	00	10
101	62	18026FE	6D2B33BE	0FFA9AF6E	36966E0342	1	0	1	1	1	01	00	00
102	63	1004DFC	5A56677D	1FF535EDD	6D2CDC0684	0	0	1	0	0	00	01	00
103	64	0009BF9	34ACCEFB	1FEA6BDBB	5A59B80D09	0	1	1	0	0	10	00	01
104	65	00137F2	69599DF7	1FD4D7B76	34B3701A12	0	0	1	1	0	00	10	00
105	66	0026FE5	52B33BEF	1FA9AF6EC	6966E03424	0	1	1	0	0	00	00	10
106	67	004DFCA	256677DF	1F535EDD8	52CDC06848	0	0	1	1	0	01	00	00
107	68	009BF94	4ACCEFBE	1EA6BDBB0	259B80D091	0	1	1	1	0	11	01	00
108	69	0137F29	1599DF7C	1D4D7B760	4B3701A123	0	1	1	0	1	10	11	01
109	70	026FE53	2B33BEF9	1A9AF6EC0	166E034246	0	0	1	0	1	01	10	11
110	71	04DFCA7	56677DF2	1535EDD81	2CDC06848D	0	0	0	1	0	01	01	10
111	72	09BF94F	2CCEFBE4	0A6BDBB03	59B80D091B	1	1	1	1	1	00	01	01
112	73	137F29E	599DF7C9	14D7B7607	33701A1236	0	1	0	0	1	11	00	01
113	74	06FE53C	333BEF93	09AF6EC0E	66E034246C	0	0	1	1	1	01	11	00
114	75	0DFCA79	6677DF26	135EDD81D	4DC06848D8	1	0	0	1	1	10	01	11
115	76	1BF94F2	4CEFBE4D	06BDBB03B	1B80D091B1	1	1	0	1	1	11	10	01
116	77	17F29E5	19DF7C9A	0D7B76077	3701A12363	0	1	1	0	1	00	11	10
117	78	0FE53CA	33BEF934	1AF6EC0EF	6E034246C6	1	1	1	0	1	11	00	11
118	79	1FCA794	677DF269	15EDD81DF	5C06848D8C	1	0	0	0	0	01	11	00
119	80	1F94F29	4EFBE4D2	0BDBB03BE	380D091B19	1	1	1	0	0	01	01	11
120	81	1F29E53	1DF7C9A5	17B76077D	701A123633	1	1	0	0	1	11	01	01
121	82	1E53CA6	3BEF934B	0F6EC0EFB	6034246C66	1	1	1	0	0	11	11	01
122	83	1CA794D	77DF2696	1EDD81DF6	406848D8CD	1	1	1	0	0	10	11	11
123	84	194F29B	6FBE4D2C	1DBB03BED	00D091B19B	1	1	1	1	0	11	10	11
124	85	129E536	5F7C9A59	1B76077DA	01A1236337	0	0	1	1	1	00	11	10
125	86	053CA6C	3EF934B3	16EC0EFB4	034246C66E	0	1	0	0	1	10	00	11
126	87	0A794D9	7DF26967	0DD81DF69	06848D8CDD	1	1	1	1	0	01	10	00
127	88	14F29B3	7BE4D2CF	1BB03BED3	0D091B19BB	0	1	1	0	1	01	01	10
128	89	09E5366	77C9A59F	176077DA6	1A12363377	1	1	0	0	1	11	01	01
129	90	13CA6CD	6F934B3F	0EC0EFB4D	34246C66EF	0	1	1	0	1	10	11	01
130	91	0794D9B	5F26967F	1D81DF69A	6848D8CDDF	0	0	1	0	1	01	10	11
131	92	0F29B37	3E4D2CFE	1B03BED35	5091B19BBE	1	0	1	1	0	10	01	10
132	93	1E5366F	7C9A59FD	16077DA6B	212363377C	1	1	0	0	0	11	10	01
133	94	1CA6CDF	7934B3FB	0C0EFB4D6	4246C66EF9	1	0	1	0	1	00	11	10
134	95	194D9BE	726967F6	181DF69AD	048D8CDDF2	1	0	1	1	1	11	00	11
135	96	129B37D	64D2CFED	103BED35B	091B19BBE5	0	1	0	0	0	01	11	00
136	97	05366FA	49A59FDA	0077DA6B7	12363377CA	0	1	0	0	0	10	01	11
137	98	0A6CDF5	134B3FB4	00EFB4D6E	246C66EF95	1	0	0	0	1	00	10	01
138	99	14D9BEA	26967F69	01DF69ADD	48D8CDDF2B	0	1	0	1	0	00	00	10
139	100	09B37D4	4D2CFED2	03BED35BB	11B19BBE56	1	0	0	1	0	01	00	00
140	101	1366FA8	1A59FDA5	077DA6B77	2363377CAC	0	0	0	0	1	01	01	00
141	102	06CDF51	34B3FB4A	0EFB4D6EF	46C66EF959	0	1	1	1	0	00	01	01
142	103	0D9BEA2	6967F695	1DF69ADDF	0D8CDDF2B2	1	0	1	1	1	10	00	01
143	104	1B37D45	52CFED2A	1BED35BBF	1B19BBE564	1	1	1	0	1	00	10	00
144	105	166FA8A	259FDA54	17DA6B77E	363377CAC8	0	1	0	0	1	01	00	10
145	106	0CDF515	4B3FB4A9	0FB4D6EFC	6C66EF9591	1	0	1	0	1	00	01	00

Sample Data

146	107	19BEA2B	167F6952	1F69ADDF8	58CDDF2B22	1	0	1	1	1	10	00	01
147	108	137D457	2CFED2A5	1ED35BBF1	319BBE5645	0	1	1	1	1	00	10	00
148	109	06FA8AF	59FDA54A	1DA6B77E2	63377CAC8B	0	1	1	0	0	00	00	10
149	110	0DF515F	33FB4A95	1B4D6EFC4	466EF95916	1	1	1	0	1	01	00	00
150	111	1BEA2BF	67F6952A	169ADDF88	0CDDF2B22C	1	1	0	1	0	11	01	00
151	112	17D457F	4FED2A55	0D35BBF10	19BBE56459	0	1	1	1	0	11	11	01
152	113	0FA8AFE	1FDA54AB	1A6B77E20	3377CAC8B3	1	1	1	0	0	10	11	11
153	114	1F515FD	3FB4A957	14D6EFC40	66EF959166	1	1	0	1	1	10	10	11
154	115	1EA2BFA	7F6952AF	09ADDF880	4DDF2B22CC	1	0	1	1	1	01	10	10
155	116	1D457F4	7ED2A55F	135BBF100	1BBE564598	1	1	0	1	0	10	01	10
156	117	1A8AFE8	7DA54ABF	06B77E200	377CAC8B31	1	1	0	0	0	11	10	01
157	118	1515FD0	7B4A957F	0D6EFC401	6EF9591663	0	0	1	1	1	00	11	10
158	119	0A2BFA1	76952AFE	1ADDF8803	5DF2B22CC7	1	1	1	1	0	00	00	11
159	120	1457F42	6D2A55FD	15BBF1007	3BE564598E	0	0	0	1	1	00	00	00
160	121	08AFE84	5A54ABFB	0B77E200F	77CAC8B31C	1	0	1	1	1	01	00	00
161	122	115FD09	34A957F7	16EFC401F	6F95916639	0	1	0	1	1	00	01	00
162	123	02BFA12	6952AFEF	0DDF8803E	5F2B22CC73	0	0	1	0	1	11	00	01
163	124	057F424	52A55FDF	1BBF1007D	3E564598E7	0	1	1	0	1	01	11	00
164	125	0AFE848	254ABFBF	177E200FA	7CAC8B31CF	1	0	0	1	1	10	01	11
165	126	15FD090	4A957F7E	0EFC401F5	795916639E	0	1	1	0	0	11	10	01
166	127	0BFA121	152AFEFD	1DF8803EA	72B22CC73C	1	0	1	1	0	01	11	10
167	128	17F4243	2A55FDFA	1BF1007D4	6564598E78	0	0	1	0	0	10	01	11
168	129	0FE8486	54ABFBF4	17E200FA8	4AC8B31CF0	1	1	0	1	1	11	10	01
169	130	1FD090C	2957F7E8	0FC401F51	15916639E1	1	0	1	1	0	01	11	10
170	131	1FA1219	52AFEFD1	1F8803EA3	2B22CC73C2	1	1	1	0	0	01	01	11
171	132	1F42432	255FDFA2	1F1007D47	564598E785	1	0	1	0	1	11	01	01
172	133	1E84865	4ABFBF44	1E200FA8F	2C8B31CF0B	1	1	1	1	1	11	11	01
173	134	1D090CB	157F7E88	1C401F51E	5916639E17	1	0	1	0	1	11	11	11
174	135	1A12196	2AFEFD11	18803EA3C	322CC73C2E	1	1	1	0	0	10	11	11
175	136	142432C	55FDFA23	11007D479	64598E785C	0	1	0	0	1	01	10	11
176	137	0848659	2BFBF446	0200FA8F2	48B31CF0B9	1	1	0	1	0	10	01	10
177	138	1090CB2	57F7E88C	0401F51E4	116639E173	0	1	0	0	1	00	10	01
178	139	0121964	2FEFD118	0803EA3C8	22CC73C2E6	0	1	1	1	1	00	00	10
179	140	02432C9	5FDFA230	1007D4791	4598E785CD	0	1	0	1	0	01	00	00
180	141	0486593	3FBF4461	000FA8F23	0B31CF0B9B	0	1	0	0	0	00	01	00
181	142	090CB26	7F7E88C3	001F51E47	16639E1736	1	0	0	0	1	11	00	01
182	143	121964D	7EFD1187	003EA3C8F	2CC73C2E6C	0	1	0	1	1	01	11	00
183	144	0432C9B	7DFA230E	007D4791E	598E785CD8	0	1	0	1	1	10	01	11
184	145	0865936	7BF4461C	00FA8F23C	331CF0B9B0	1	1	0	0	0	11	10	01
185	146	10CB26D	77E88C38	01F51E479	6639E17361	0	1	0	0	0	00	11	10
186	147	01964DA	6FD11870	03EA3C8F2	4C73C2E6C2	0	1	0	0	1	10	00	11
187	148	032C9B4	5FA230E1	07D4791E4	18E785CD84	0	1	0	1	0	00	10	00
188	149	0659368	3F4461C2	0FA8F23C9	31CF0B9B09	0	0	1	1	0	00	00	10
189	150	0CB26D0	7E88C384	1F51E4793	639E173612	1	1	1	1	0	10	00	00
190	151	1964DA0	7D118709	1EA3C8F27	473C2E6C24	1	0	1	0	0	00	10	00
191	152	12C9B41	7A230E12	1D4791E4E	0E785CD848	0	0	1	0	1	01	00	10
192	153	0593683	74461C24	1A8F23C9C	1CF0B9B091	0	0	1	1	1	00	01	00
193	154	0B26D06	688C3848	151E47938	39E1736123	1	1	0	1	1	10	00	01
194	155	164DA0D	51187091	0A3C8F271	73C2E6C247	0	0	1	1	0	00	10	00
195	156	0C9B41A	2230E123	14791E4E3	6785CD848F	1	0	0	1	0	00	00	10
196	157	1936835	4461C247	08F23C9C6	4F0B9B091E	1	0	1	0	0	01	00	00
197	158	126D06A	08C3848E	11E47938D	1E1736123C	0	1	0	0	0	00	01	00
198	159	04DA0D5	1187091C	03C8F271B	3C2E6C2478	0	1	0	0	1	11	00	01
199	160	09B41AA	230E1238	0791E4E37	785CD848F1	1	0	0	0	0	01	11	00
200	161	1368354	461C2470	0F23C9C6F	70B9B091E3	0	0	1	1	1	10	01	11
201	162	06D06A9	0C3848E1	1E47938DF	61736123C6	0	0	1	0	1	00	10	01
202	163	0DA0D52	187091C3	1C8F271BE	42E6C2478D	1	0	1	1	1	00	00	10

Sample Data

203	164	1B41AA4	30E12387	191E4E37C	05CD848F1A	1	1	1	1	0	10	00	00
204	165	1683549	61C2470F	123C9C6F9	0B9B091E34	0	1	0	1	0	00	10	00
205	166	0D06A92	43848E1E	047938DF3	1736123C68	1	1	0	0	0	00	00	10
206	167	1A0D524	07091C3C	08F271BE7	2E6C2478D1	1	0	1	0	0	01	00	00
207	168	141AA49	0E123879	11E4E37CF	5CD848F1A2	0	0	0	1	0	00	01	00
208	169	0835492	1C2470F3	03C9C6F9F	39B091E345	1	0	0	1	0	10	00	01
209	170	106A925	3848E1E6	07938DF3F	736123C68B	0	0	0	0	0	11	10	00
210	171	00D524A	7091C3CD	0F271BE7E	66C2478D16	0	1	1	1	0	01	11	10
211	172	01AA495	6123879B	1E4E37CFD	4D848F1A2D	0	0	1	1	1	10	01	11
212	173	035492A	42470F36	1C9C6F9FB	1B091E345B	0	0	1	0	1	00	10	01
213	174	06A9255	048E1E6C	1938DF3F6	36123C68B7	0	1	1	0	0	00	00	10
214	175	0D524AB	091C3CD8	1271BE7EC	6C2478D16E	1	0	0	0	1	00	00	00
215	176	1AA4957	123879B1	04E37CFD8	5848F1A2DD	1	0	0	0	1	00	00	00
216	177	15492AF	2470F363	09C6F9FB0	3091E345BA	0	0	1	1	0	01	00	00
217	178	0A9255E	48E1E6C7	138DF3F61	6123C68B75	1	1	0	0	1	00	01	00
218	179	1524ABD	11C3CD8F	071BE7EC3	42478D16EB	0	1	0	0	1	11	00	01
219	180	0A4957B	23879B1F	0E37CFD87	048F1A2DD6	1	1	1	1	1	00	11	00
220	181	1492AF6	470F363F	1C6F9FB0E	091E345BAD	0	0	1	0	1	10	00	11
221	182	09255EC	0E1E6C7F	18DF3F61D	123C68B75B	1	0	1	0	0	00	10	00
222	183	124ABD9	1C3CD8FF	11BE7EC3A	2478D16EB6	0	0	0	0	0	01	00	10
223	184	04957B3	3879B1FE	037CFD874	48F1A2DD6D	0	0	0	1	0	00	01	00
224	185	092AF66	70F363FD	06F9FB0E9	11E345BADB	1	1	0	1	1	10	00	01
225	186	1255ECD	61E6C7FA	0DF3F61D3	23C68B75B7	0	1	1	1	1	00	10	00
226	187	04ABD9B	43CD8FF5	1BE7EC3A7	478D16EB6E	0	1	1	1	1	00	00	10
227	188	0957B37	079B1FEA	17CFD874E	0F1A2DD6DD	1	1	0	0	0	01	00	00
228	189	12AF66F	0F363FD4	0F9FB0E9C	1E345BADBB	0	0	1	0	0	00	01	00
229	190	055ECDE	1E6C7FA9	1F3F61D39	3C68B75B76	0	0	1	0	1	11	00	01
230	191	0ABD9BC	3CD8FF53	1E7EC3A73	78D16EB6EC	1	1	1	1	1	00	11	00
231	192	157B379	79B1FEA7	1CFD874E6	71A2DD6DD9	0	1	1	1	1	11	00	11
232	193	0AF66F3	7363FD4E	19FB0E9CD	6345BADBB2	1	0	1	0	1	01	11	00
233	194	15ECDE6	66C7FA9D	13F61D39A	468B75B765	0	1	0	1	1	10	01	11
234	195	0BD9BCC	4D8FF53A	07EC3A735	0D16EB6ECA	1	1	0	0	0	11	10	01
235	196	17B3799	1B1FEA75	0FD874E6A	1A2DD6DD94	0	0	1	0	0	00	11	10
236	197	0F66F33	363FD4EA	1FB0E9CD5	345BADBB28	1	0	1	0	0	11	00	11
237	198	1ECDE67	6C7FA9D5	1F61D39AA	68B75B7650	1	0	1	1	0	00	11	00
238	199	1D9BCCF	58FF53AB	1EC3A7354	516EB6ECA0	1	1	1	0	1	11	00	11
239	200	1B3799E	31FEA756	1D874E6A8	22DD6DD940	1	1	1	1	1	00	11	00

```
Z[0]  = 3F
Z[1]  = B1
Z[2]  = 67
Z[3]  = D2
Z[4]  = 2F
Z[5]  = A6
Z[6]  = 1F
Z[7]  = B9
Z[8]  = E6
Z[9]  = 84
Z[10] = 43
Z[11] = 07
Z[12] = D8
Z[13] = 1E
Z[14] = E7
Z[15] = C3
```

Sample Data

```
================================================================================================
         Reload this pattern into the LFSRs
         Hold content of Summation Combiner regs and calculate new C[t+1] and Z values
================================================================================================
LFSR1  <= 0E62F3F
LFSR2  <= 6C84A6B1
LFSR3  <= 11E431F67
LFSR4  <= 61E707B9D2
C[t+1] <= 00

================================================================================================
         Generating 125 key symbols (encryption/decryption sequence)
================================================================================================
240     1    0E62F3F  6C84A6B1  11E431F67  61E707B9D2      1  1  0  1    0  00     11   00
241     2    1CC5E7F  59094D63  03C863ECE  43CE0F73A5      1  0  0  1    0  11     00   11
242     3    198BCFF  32129AC6  0790C7D9D  079C1EE74A      1  0  0  1    1  01     11   00
243     4    13179FE  6425358C  0F218FB3A  0F383DCE94      0  0  1  0    0  10     01   11
244     5    062F3FD  484A6B19  1E431F675  1E707B9D28      0  0  1  0    1  00     10   01
245     6    0C5E7FB  1094D632  1C863ECEB  3CE0F73A50      1  1  1  1    0  11     00   10
246     7    18BCFF7  2129AC64  190C7D9D7  79C1EE74A1      1  0  1  1    0  00     11   00
247     8    1179FEE  425358C8  1218FB3AE  7383DCE942      0  0  0  1    1  10     00   11
248     9    02F3FDD  04A6B190  0431F675D  6707B9D285      0  1  0  0    1  11     10   00
249    10    05E7FBB  094D6320  0863ECEBB  4E0F73A50B      0  0  1  0    0  00     11   10
250    11    0BCFF77  129AC640  10C7D9D77  1C1EE74A16      1  1  0  0    0  11     00   11
251    12    179FEEE  25358C80  018FB3AEE  383DCE942C      0  0  0  0    1  10     11   00
252    13    0F3FDDC  4A6B1900  031F675DD  707B9D2859      1  0  0  0    1  01     10   11
253    14    1E7FBB8  14D63200  063ECEBBA  60F73A50B3      1  1  0  1    0  10     01   10
254    15    1CFF771  29AC6401  0C7D9D774  41EE74A167      1  1  1  1    0  10     10   01
255    16    19FEEE2  5358C803  18FB3AEE9  03DCE942CE      1  0  1  1    1  01     10   10
256    17    13FDDC4  26B19007  11F675DD2  07B9D2859C      0  1  0  1    1  01     01   10
257    18    07FBB88  4D63200E  03ECEBBA4  0F73A50B38      0  0  0  0    1  10     01   01
258    19    0FF7711  1AC6401D  07D9D7748  1EE74A1670      1  1  0  1    1  11     10   01
259    20    1FEEE23  358C803B  0FB3AEE91  3DCE942CE1      1  1  1  1    1  01     11   10
260    21    1FDDC47  6B190076  1F675DD23  7B9D2859C2      1  0  1  1    0  01     01   11
261    22    1FBB88F  563200ED  1ECEBBA47  773A50B385      1  0  1  0    1  11     01   01
262    23    1F7711E  2C6401DB  1D9D7748F  6E74A1670A      1  0  1  0    1  10     11   01
263    24    1EEE23D  58C803B6  1B3AEE91E  5CE942CE15      1  1  1  1    0  11     10   11
264    25    1DDC47A  3190076C  1675DD23D  39D2859C2B      1  1  0  1    0  01     11   10
265    26    1BB88F4  63200ED9  0CEBBA47A  73A50B3856      1  0  1  1    0  01     01   11
266    27    17711E8  46401DB2  19D7748F5  674A1670AD      0  0  1  0    0  11     01   01
267    28    0EE23D0  0C803B64  13AEE91EA  4E942CE15B      1  1  0  1    0  11     11   01
268    29    1DC47A0  190076C8  075DD23D4  1D2859C2B7      1  0  0  0    0  11     11   11
269    30    1B88F41  3200ED90  0EBBA47A9  3A50B3856E      1  0  1  0    1  11     11   11
270    31    1711E83  6401DB20  1D7748F53  74A1670ADC      0  0  1  1    1  11     11   11
271    32    0E23D07  4803B641  1AEE91EA7  6942CE15B8      1  0  1  0    1  11     11   11
272    33    1C47A0F  10076C82  15DD23D4F  52859C2B71      1  0  0  1    1  11     11   11
273    34    188F41E  200ED905  0BBA47A9E  250B3856E3      1  0  1  0    1  11     11   11
274    35    111E83C  401DB20A  17748F53D  4A1670ADC7      0  0  0  0    1  00     11   11
275    36    023D078  003B6414  0EE91EA7A  142CE15B8E      0  0  1  0    1  10     00   11
276    37    047A0F0  0076C828  1DD23D4F5  2859C2B71C      0  0  1  0    1  11     10   00
277    38    08F41E1  00ED9050  1BA47A9EA  50B3856E39      1  1  1  1    1  01     11   10
278    39    11E83C2  01DB20A0  1748F53D5  21670ADC72      0  1  0  0    0  10     01   11
279    40    03D0785  03B64141  0E91EA7AA  42CE15B8E4      0  1  1  1    1  11     10   01
280    41    07A0F0A  076C8283  1D23D4F54  059C2B71C8      0  0  1  1    1  00     11   10
281    42    0F41E14  0ED90507  1A47A9EA9  0B3856E390      1  1  1  0    1  11     00   11
282    43    1E83C29  1DB20A0F  148F53D52  1670ADC720      1  1  0  0    1  01     11   00
283    44    1D07853  3B64141E  091EA7AA5  2CE15B8E40      1  0  1  1    0  01     01   11
```

Sample Data

284	45	1A0F0A6	76C8283C	123D4F54B	59C2B71C81	1	1	0	1	0	00	01	01
285	46	141E14C	6D905079	047A9EA97	33856E3902	0	1	0	1	0	10	00	01
286	47	083C299	5B20A0F2	08F53D52F	670ADC7204	1	0	1	0	0	00	10	00
287	48	1078533	364141E4	11EA7AA5E	4E15B8E408	0	0	0	0	0	01	00	10
288	49	00F0A67	6C8283C8	03D4F54BC	1C2B71C811	0	1	0	0	0	00	01	00
289	50	01E14CE	59050791	07A9EA978	3856E39022	0	0	0	0	0	11	00	01
290	51	03C299C	320A0F23	0F53D52F1	70ADC72045	0	0	1	1	1	01	11	00
291	52	0785339	64141E47	1EA7AA5E2	615B8E408A	0	0	1	0	0	10	01	11
292	53	0F0A673	48283C8E	1D4F54BC4	42B71C8115	1	0	1	1	1	11	10	01
293	54	1E14CE6	1050791C	1A9EA9788	056E39022B	1	0	1	0	1	00	11	10
294	55	1C299CD	20A0F239	153D52F10	0ADC720456	1	1	0	1	1	11	00	11
295	56	185339B	4141E472	0A7AA5E20	15B8E408AC	1	0	1	1	0	00	11	00
296	57	10A6736	0283C8E4	14F54BC41	2B71C81158	0	1	0	0	1	10	00	11
297	58	014CE6C	050791C9	09EA97882	56E39022B0	0	0	1	1	0	00	10	00
298	59	0299CD9	0A0F2393	13D52F104	2DC7204561	0	0	0	1	1	01	00	10
299	60	05339B3	141E4726	07AA5E208	5B8E408AC3	0	0	0	1	0	00	01	00
300	61	0A67366	283C8E4C	0F54BC411	371C811587	1	0	1	0	0	10	00	01
301	62	14CE6CC	50791C98	1EA978822	6E39022B0F	0	0	1	0	1	11	10	00
302	63	099CD99	20F23930	1D52F1045	5C7204561E	1	1	1	0	0	01	11	10
303	64	1339B33	41E47260	1AA5E208B	38E408AC3D	0	1	1	1	0	01	01	11
304	65	0673666	03C8E4C0	154BC4117	71C811587A	0	1	0	1	1	11	01	01
305	66	0CE6CCC	0791C980	0A978822E	639022B0F5	1	1	1	1	1	11	11	01
306	67	19CD999	0F239301	152F1045C	47204561EB	1	0	0	0	0	11	11	11
307	68	139B332	1E472603	0A5E208B9	0E408AC3D6	0	0	1	0	0	11	11	11
308	69	0736664	3C8E4C06	14BC41172	1C811587AD	0	1	0	1	1	11	11	11
309	70	0E6CCC8	791C980C	0978822E5	39022B0F5A	1	0	1	0	1	11	11	11
310	71	1CD9990	72393019	12F1045CB	7204561EB4	1	0	0	0	0	11	11	11
311	72	19B3320	64726033	05E208B97	6408AC3D69	1	0	0	0	0	11	11	11
312	73	1366640	48E4C067	0BC41172F	4811587AD3	0	1	1	0	1	11	11	11
313	74	06CCC81	11C980CF	178822E5E	1022B0F5A6	0	1	0	0	0	11	11	11
314	75	0D99903	2393019E	0F1045CBC	204561EB4C	1	1	1	0	0	10	11	11
315	76	1B33206	4726033D	1E208B979	408AC3D699	1	0	1	1	1	10	10	11
316	77	166640D	0E4C067B	1C41172F2	011587AD33	0	0	1	0	1	10	10	10
317	78	0CCC81B	1C980CF6	18822E5E5	022B0F5A66	1	1	1	0	1	01	10	10
318	79	1999036	393019EC	11045CBCA	04561EB4CD	1	0	0	0	0	01	01	10
319	80	133206C	726033D9	0208B9794	08AC3D699B	0	0	0	1	0	11	01	01
320	81	06640D9	64C067B3	041172F29	11587AD337	0	1	0	0	0	10	11	01
321	82	0CC81B3	4980CF66	0822E5E53	22B0F5A66F	1	1	1	1	0	11	10	11
322	83	1990366	13019ECC	1045CBCA6	4561EB4CDF	1	0	0	0	0	00	11	10
323	84	13206CC	26033D98	008B9794D	0AC3D699BE	0	0	0	1	1	10	00	11
324	85	0640D98	4C067B31	01172F29B	1587AD337C	0	0	0	1	1	11	10	00
325	86	0C81B30	180CF662	022E5E537	2B0F5A66F9	1	0	0	0	0	00	11	10
326	87	1903660	3019ECC5	045CBCA6F	561EB4CDF3	1	0	0	0	1	10	00	11
327	88	1206CC1	6033D98A	08B9794DE	2C3D699BE6	0	0	1	0	1	11	10	00
328	89	040D983	4067B315	1172F29BD	587AD337CC	0	0	0	0	1	11	11	10
329	90	081B306	00CF662A	02E5E537A	30F5A66F98	1	1	0	1	0	10	11	11
330	91	103660C	019ECC55	05CBCA6F4	61EB4CDF31	0	1	0	1	0	10	10	11
331	92	006CC19	033D98AB	0B9794DE8	43D699BE62	0	0	1	1	0	01	10	10
332	93	00D9833	067B3156	172F29BD0	07AD337CC5	0	0	0	1	0	01	01	10
333	94	01B3066	0CF662AC	0E5E537A0	0F5A66F98B	0	1	1	0	1	11	01	01
334	95	03660CD	19ECC559	1CBCA6F41	1EB4CDF317	0	1	1	1	0	11	11	01
335	96	06CC19B	33D98AB2	19794DE83	3D699BE62F	0	1	1	0	1	11	11	11
336	97	0D98336	67B31565	12F29BD06	7AD337CC5F	1	1	0	1	0	10	11	11
337	98	1B3066D	4F662ACA	05E537A0C	75A66F98BF	1	0	0	1	0	10	10	11
338	99	1660CDB	1ECC5594	0BCA6F418	6B4CDF317E	0	1	1	0	0	01	10	10
339	100	0CC19B7	3D98AB29	1794DE831	5699BE62FC	1	1	0	1	0	10	01	10
340	101	198336F	7B315653	0F29BD062	2D337CC5F9	1	0	1	0	0	11	10	01

Sample Data

341	102	13066DE	7662ACA7	1E537A0C5	5A66F98BF2	0	0	1	0	0	00	11	10
342	103	060CDBC	6CC5594F	1CA6F418B	34CDF317E4	0	1	1	1	1	11	00	11
343	104	0C19B78	598AB29F	194DE8317	699BE62FC9	1	1	1	1	1	00	11	00
344	105	18336F1	3315653F	129BD062E	5337CC5F92	1	0	0	0	1	10	00	11
345	106	1066DE2	662ACA7E	0537A0C5C	266F98BF25	0	0	0	0	0	11	10	00
346	107	00CDBC5	4C5594FD	0A6F418B9	4CDF317E4B	0	0	1	1	1	00	11	10
347	108	019B78B	18AB29FA	14DE83172	19BE62FC96	0	1	0	1	0	11	00	11
348	109	0336F16	315653F4	09BD062E5	337CC5F92C	0	0	1	0	0	01	11	00
349	110	066DE2D	62ACA7E8	137A0C5CA	66F98BF258	0	1	0	1	1	10	01	11
350	111	0CDBC5B	45594FD1	06F418B95	4DF317E4B1	1	0	0	1	0	11	10	01
351	112	19B78B6	0AB29FA2	0DE83172B	1BE62FC962	1	1	1	1	1	01	11	10
352	113	136F16C	15653F45	1BD062E57	37CC5F92C5	0	0	1	1	1	10	01	11
353	114	06DE2D9	2ACA7E8B	17A0C5CAE	6F98BF258B	0	1	0	1	0	11	10	01
354	115	0DBC5B2	5594FD16	0F418B95D	5F317E4B16	1	1	1	0	0	01	11	10
355	116	1B78B64	2B29FA2C	1E83172BB	3E62FC962C	1	0	1	0	1	10	01	11
356	117	16F16C8	5653F458	1D062E577	7CC5F92C58	0	0	1	1	0	11	10	01
357	118	0DE2D91	2CA7E8B0	1A0C5CAEF	798BF258B1	1	1	1	1	1	01	11	10
358	119	1BC5B23	594FD161	1418B95DF	7317E4B163	1	0	0	0	0	10	01	11
359	120	178B647	329FA2C2	083172BBF	662FC962C7	0	1	1	0	0	11	10	01
360	121	0F16C8E	653F4584	1062E577F	4C5F92C58E	1	0	0	0	0	00	11	10
361	122	1E2D91C	4A7E8B09	00C5CAEFE	18BF258B1C	1	0	0	1	0	11	00	11
362	123	1C5B238	14FD1613	018B95DFC	317E4B1639	1	1	0	0	1	01	11	00
363	124	18B6471	29FA2C27	03172BBF9	62FC962C72	1	1	0	1	0	01	01	11
364	125	116C8E2	53F4584E	062E577F3	45F92C58E4	0	1	0	1	1	11	01	01

2 FREQUENCY HOPPING SAMPLE DATA

The section contains three sets of sample data showing the basic and adapted hopping schemes for different combinations of addresses and initial clock values.

2.1 FIRST SET

```
Hop sequence {k} for PAGE SCAN/INQUIRY SCAN SUBSTATE:
CLKN start:       0x0000000
UAP / LAP:        0x00000000
#ticks:           0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
                  -------------------------------------------------------
0x0000000:           0 |    2 |    4 |    6 |    8 |   10 |   12 |   14 |
0x0008000:          16 |   18 |   20 |   22 |   24 |   26 |   28 |   30 |
0x0010000:          32 |   34 |   36 |   38 |   40 |   42 |   44 |   46 |
0x0018000:          48 |   50 |   52 |   54 |   56 |   58 |   60 |   62 |
0x0020000:           0 |    2 |    4 |    6 |    8 |   10 |   12 |   14 |
0x0028000:          16 |   18 |   20 |   22 |   24 |   26 |   28 |   30 |
0x0030000:          32 |   34 |   36 |   38 |   40 |   42 |   44 |   46 |
0x0038000:          48 |   50 |   52 |   54 |   56 |   58 |   60 |   62 |

Hop sequence {k} for PAGE STATE/INQUIRY SUBSTATE:
CLKE start:       0x0000000
UAP / LAP:        0x00000000
#ticks:           00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
                  -----------------------------------------------------
0x0000000:        48 50 09 13 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0000010:        00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
0x0000020:        48 50 09 13 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0000030:        00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
...
0x0001000:        48 18 09 05 | 20 22 33 37 | 24 26 03 07 | 28 30 35 39 |
0x0001010:        32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
0x0001020:        48 18 09 05 | 20 22 33 37 | 24 26 03 07 | 28 30 35 39 |
0x0001030:        32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
...
0x0002000:        16 18 01 05 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0002010:        00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
0x0002020:        16 18 01 05 | 52 54 41 45 | 56 58 11 15 | 60 62 43 47 |
0x0002030:        00 02 64 68 | 04 06 17 21 | 08 10 66 70 | 12 14 19 23 |
...
0x0003000:        48 50 09 13 | 52 22 41 37 | 24 26 03 07 | 28 30 35 39 |
0x0003010:        32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |
0x0003020:        48 50 09 13 | 52 22 41 37 | 24 26 03 07 | 28 30 35 39 |
0x0003030:        32 34 72 76 | 36 38 25 29 | 40 42 74 78 | 44 46 27 31 |

Hop sequence {k} for SLAVE PAGE RESPONSE SUBSTATE:
CLKN* =           0x0000010
UAP / LAP:        0x00000000
#ticks:           00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a 1c | 1e
                  ---------------------------------------------------------------
0x0000012:        64 | 02 68 | 04 17 | 06 21 | 08 66 | 10 70 | 12 19 | 14 23 | 16
0x0000032:        01 | 18 05 | 20 33 | 22 37 | 24 03 | 26 07 | 28 35 | 30 39 | 32
0x0000052:        72 | 34 76 | 36 25 | 38 29 | 40 74 | 42 78 | 44 27 | 46 31 | 48
0x0000072:        09 | 50 13 | 52 41 | 54 45 | 56 11 | 58 15 | 60 43 | 62 47 | 00

Hop sequence {k} for MASTER PAGE RESPONSE SUBSTATE:
Offset value:     24
CLKE* =           0x0000012
UAP / LAP:        0x00000000
#ticks:           00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
                  -----------------------------------------------------------------
0x0000014:        02 68 | 04 17 | 06 21 | 08 66 | 10 70 | 12 19 | 14 23 | 16 01 |
0x0000034:        18 05 | 20 33 | 22 37 | 24 03 | 26 07 | 28 35 | 30 39 | 32 72 |
0x0000054:        34 76 | 36 25 | 38 29 | 40 74 | 42 78 | 44 27 | 46 31 | 48 09 |
0x0000074:        50 13 | 52 41 | 54 45 | 56 11 | 58 15 | 60 43 | 62 47 | 00 64 |
```

Sample Data

Hop sequence {k} for CONNECTION STATE (Basic channel hopping sequence; ie, non-AFH):
CLK start: 0x0000010
UAP/LAP: 0x00000000

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010:	08 66	10 70	12 19	14 23	16 01	18 05	20 33	22 37
0x0000030:	24 03	26 07	28 35	30 39	32 72	34 76	36 25	38 29
0x0000050:	40 74	42 78	44 27	46 31	48 09	50 13	52 41	54 45
0x0000070:	56 11	58 15	60 43	62 47	32 17	36 19	34 49	38 51
0x0000090:	40 21	44 23	42 53	46 55	48 33	52 35	50 65	54 67
0x00000b0:	56 37	60 39	58 69	62 71	64 25	68 27	66 57	70 59
0x00000d0:	72 29	76 31	74 61	78 63	01 41	05 43	03 73	07 75
0x00000f0:	09 45	13 47	11 77	15 00	64 49	66 53	68 02	70 06
0x0000110:	01 51	03 55	05 04	07 08	72 57	74 61	76 10	78 14
0x0000130:	09 59	11 63	13 12	15 16	17 65	19 69	21 18	23 22
0x0000150:	33 67	35 71	37 20	39 24	25 73	27 77	29 26	31 30
0x0000170:	41 75	43 00	45 28	47 32	17 02	21 04	19 34	23 36
0x0000190:	33 06	37 08	35 38	39 40	25 10	29 12	27 42	31 44
0x00001b0:	41 14	45 16	43 46	47 48	49 18	53 20	51 50	55 52
0x00001d0:	65 22	69 24	67 54	71 56	57 26	61 28	59 58	63 60
0x00001f0:	73 30	77 32	75 62	00 64	49 34	51 42	57 66	59 74
0x0000210:	53 36	55 44	61 68	63 76	65 50	67 58	73 03	75 11
0x0000230:	69 52	71 60	77 05	00 13	02 38	04 46	10 70	12 78
0x0000250:	06 40	08 48	14 72	16 01	18 54	20 62	26 07	28 15
0x0000270:	22 56	24 64	30 09	32 17	02 66	06 74	10 19	14 27
0x0000290:	04 70	08 78	12 23	16 31	18 03	22 11	26 35	30 43
0x00002b0:	20 07	24 15	28 39	32 47	34 68	38 76	42 21	46 29
0x00002d0:	36 72	40 01	44 25	48 33	50 05	54 13	58 37	62 45
0x00002f0:	52 09	56 17	60 41	64 49	34 19	36 35	50 51	52 67
0x0000310:	38 21	40 37	54 53	56 69	42 27	44 43	58 59	60 75
0x0000330:	46 29	48 45	62 61	64 77	66 23	68 39	03 55	05 71
0x0000350:	70 25	72 41	07 57	09 73	74 31	76 47	11 63	13 00
0x0000370:	78 33	01 49	15 65	17 02	66 51	70 67	03 04	07 20
0x0000390:	68 55	72 71	05 08	09 24	74 59	78 75	11 12	15 28
0x00003b0:	76 63	01 00	13 16	17 32	19 53	23 69	35 06	39 22
0x00003d0:	21 57	25 73	37 10	41 26	27 61	31 77	43 14	47 30
0x00003f0:	29 65	33 02	45 18	49 34	19 04	21 08	23 20	25 24

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with all channel used; ie, AFH(79)):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x7fffffffffffffffffff

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	08 08	10 10	12 12	14 14	16 16	18 18	20 20	22 22
0x0000030	24 24	26 26	28 28	30 30	32 32	34 34	36 36	38 38
0x0000050	40 40	42 42	44 44	46 46	48 48	50 50	52 52	54 54
0x0000070	56 56	58 58	60 60	62 62	32 32	36 36	34 34	38 38
0x0000090	40 40	44 44	42 42	46 46	48 48	52 52	50 50	54 54
0x00000b0	56 56	60 60	58 58	62 62	64 64	68 68	66 66	70 70
0x00000d0	72 72	76 76	74 74	78 78	01 01	05 05	03 03	07 07
0x00000f0	09 09	13 13	11 11	15 15	64 64	66 66	68 68	70 70
0x0000110	01 01	03 03	05 05	07 07	72 72	74 74	76 76	78 78
0x0000130	09 09	11 11	13 13	15 15	17 17	19 19	21 21	23 23
0x0000150	33 33	35 35	37 37	39 39	25 25	27 27	29 29	31 31
0x0000170	41 41	43 43	45 45	47 47	17 17	21 21	19 19	23 23
0x0000190	33 33	37 37	35 35	39 39	25 25	29 29	27 27	31 31
0x00001b0	41 41	45 45	43 43	47 47	49 49	53 53	51 51	55 55
0x00001d0	65 65	69 69	67 67	71 71	57 57	61 61	59 59	63 63
0x00001f0	73 73	77 77	75 75	00 00	49 49	51 51	57 57	59 59
0x0000210	53 53	55 55	61 61	63 63	65 65	67 67	73 73	75 75
0x0000230	69 69	71 71	77 77	00 00	02 02	04 04	10 10	12 12
0x0000250	06 06	08 08	14 14	16 16	18 18	20 20	26 26	28 28
0x0000270	22 22	24 24	30 30	32 32	02 02	06 06	10 10	14 14
0x0000290	04 04	08 08	12 12	16 16	18 18	22 22	26 26	30 30
0x00002b0	20 20	24 24	28 28	32 32	34 34	38 38	42 42	46 46
0x00002d0	36 36	40 40	44 44	48 48	50 50	54 54	58 58	62 62
0x00002f0	52 52	56 56	60 60	64 64	34 34	36 36	50 50	52 52
0x0000310	38 38	40 40	54 54	56 56	42 42	44 44	58 58	60 60
0x0000330	46 46	48 48	62 62	64 64	66 66	68 68	03 03	05 05
0x0000350	70 70	72 72	07 07	09 09	74 74	76 76	11 11	13 13
0x0000370	78 78	01 01	15 15	17 17	66 66	70 70	03 03	07 07
0x0000390	68 68	72 72	05 05	09 09	74 74	78 78	11 11	15 15
0x00003b0	76 76	01 01	13 13	17 17	19 19	23 23	35 35	39 39
0x00003d0	21 21	25 25	37 37	41 41	27 27	31 31	43 43	47 47
0x00003f0	29 29	33 33	45 45	49 49	19 19	21 21	23 23	25 25

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with channels 0 to 21 unused):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x7fffffffffffc00000

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	30 30	32 32	34 34	36 36	38 38	40 40	42 42	22 22
0x0000030	24 24	26 26	28 28	30 30	32 32	34 34	36 36	38 38
0x0000050	40 40	42 42	44 44	46 46	48 48	50 50	52 52	54 54
0x0000070	56 56	58 58	60 60	62 62	32 32	36 36	34 34	38 38
0x0000090	40 40	44 44	42 42	46 46	48 48	52 52	50 50	54 54
0x00000b0	56 56	60 60	58 58	62 62	64 64	68 68	66 66	70 70
0x00000d0	72 72	76 76	74 74	78 78	45 45	49 49	47 47	51 51
0x00000f0	53 53	57 57	55 55	59 59	64 64	66 66	68 68	70 70
0x0000110	45 45	47 47	49 49	51 51	72 72	74 74	76 76	78 78
0x0000130	53 53	55 55	57 57	59 59	61 61	63 63	65 65	23 23
0x0000150	33 33	35 35	37 37	39 39	25 25	27 27	29 29	31 31
0x0000170	41 41	43 43	45 45	47 47	61 61	65 65	63 63	23 23
0x0000190	33 33	37 37	35 35	39 39	25 25	29 29	27 27	31 31
0x00001b0	41 41	45 45	43 43	47 47	49 49	53 53	51 51	55 55
0x00001d0	65 65	69 69	67 67	71 71	57 57	61 61	59 59	63 63
0x00001f0	73 73	77 77	75 75	66 66	49 49	51 51	57 57	59 59
0x0000210	53 53	55 55	61 61	63 63	65 65	67 67	73 73	75 75
0x0000230	69 69	71 71	77 77	66 66	68 68	70 70	76 76	78 78
0x0000250	72 72	74 74	23 23	25 25	27 27	29 29	26 26	28 28
0x0000270	22 22	24 24	30 30	32 32	68 68	72 72	76 76	23 23
0x0000290	70 70	74 74	78 78	25 25	27 27	22 22	26 26	30 30
0x00002b0	29 29	24 24	28 28	32 32	34 34	38 38	42 42	46 46
0x00002d0	36 36	40 40	44 44	48 48	50 50	54 54	58 58	62 62
0x00002f0	52 52	56 56	60 60	64 64	34 34	36 36	50 50	52 52
0x0000310	38 38	40 40	54 54	56 56	42 42	44 44	58 58	60 60
0x0000330	46 46	48 48	62 62	64 64	66 66	68 68	34 34	36 36
0x0000350	70 70	72 72	38 38	40 40	74 74	76 76	42 42	44 44
0x0000370	78 78	32 32	46 46	48 48	66 66	70 70	34 34	38 38
0x0000390	68 68	72 72	36 36	40 40	74 74	78 78	42 42	46 46
0x00003b0	76 76	32 32	44 44	48 48	50 50	23 23	35 35	39 39
0x00003d0	52 52	25 25	37 37	41 41	27 27	31 31	43 43	47 47
0x00003f0	29 29	33 33	45 45	49 49	50 50	52 52	23 23	25 25

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with even channels used):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x55555555555555555

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	08 08	10 10	12 12	14 14	16 16	18 18	20 20	22 22
0x0000030	24 24	26 26	28 28	30 30	32 32	34 34	36 36	38 38
0x0000050	40 40	42 42	44 44	46 46	48 48	50 50	52 52	54 54
0x0000070	56 56	58 58	60 60	62 62	32 32	36 36	34 34	38 38
0x0000090	40 40	44 44	42 42	46 46	48 48	52 52	50 50	54 54
0x00000b0	56 56	60 60	58 58	62 62	64 64	68 68	66 66	70 70
0x00000d0	72 72	76 76	74 74	78 78	00 00	04 04	02 02	06 06
0x00000f0	08 08	12 12	10 10	14 14	64 64	66 66	68 68	70 70
0x0000110	00 00	02 02	04 04	06 06	72 72	74 74	76 76	78 78
0x0000130	08 08	10 10	12 12	14 14	16 16	18 18	20 20	22 22
0x0000150	32 32	34 34	36 36	38 38	24 24	26 26	28 28	30 30
0x0000170	40 40	42 42	44 44	46 46	16 16	20 20	18 18	22 22
0x0000190	32 32	36 36	34 34	38 38	24 24	28 28	26 26	30 30
0x00001b0	40 40	44 44	42 42	46 46	48 48	52 52	50 50	54 54
0x00001d0	64 64	68 68	66 66	70 70	56 56	60 60	58 58	62 62
0x00001f0	72 72	76 76	74 74	00 00	48 48	50 50	56 56	58 58
0x0000210	52 52	54 54	60 60	62 62	64 64	66 66	72 72	74 74
0x0000230	68 68	70 70	76 76	00 00	02 02	04 04	10 10	12 12
0x0000250	06 06	08 08	14 14	16 16	18 18	20 20	26 26	28 28
0x0000270	22 22	24 24	30 30	32 32	02 02	06 06	10 10	14 14
0x0000290	04 04	08 08	12 12	16 16	18 18	22 22	26 26	30 30
0x00002b0	20 20	24 24	28 28	32 32	34 34	38 38	42 42	46 46
0x00002d0	36 36	40 40	44 44	48 48	50 50	54 54	58 58	62 62
0x00002f0	52 52	56 56	60 60	64 64	34 34	36 36	50 50	52 52
0x0000310	38 38	40 40	54 54	56 56	42 42	44 44	58 58	60 60
0x0000330	46 46	48 48	62 62	64 64	66 66	68 68	00 00	02 02
0x0000350	70 70	72 72	04 04	06 06	74 74	76 76	08 08	10 10
0x0000370	78 78	78 78	12 12	14 14	66 66	70 70	00 00	04 04
0x0000390	68 68	72 72	02 02	06 06	74 74	78 78	08 08	12 12
0x00003b0	76 76	78 78	10 10	14 14	16 16	20 20	32 32	36 36
0x00003d0	18 18	22 22	34 34	38 38	24 24	28 28	40 40	44 44
0x00003f0	26 26	30 30	42 42	46 46	16 16	18 18	20 20	22 22

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with odd channels used):

CLK start: 0x0000010

ULAP: 0x00000000

Used Channels:0x2aaaaaaaaaaaaaaaaaaa

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	09 09	11 11	13 13	15 15	17 17	19 19	21 21	23 23
0x0000030	25 25	27 27	29 29	31 31	33 33	35 35	37 37	39 39
0x0000050	41 41	43 43	45 45	47 47	49 49	51 51	53 53	55 55
0x0000070	57 57	59 59	61 61	63 63	33 33	37 37	35 35	39 39
0x0000090	41 41	45 45	43 43	47 47	49 49	53 53	51 51	55 55
0x00000b0	57 57	61 61	59 59	63 63	65 65	69 69	67 67	71 71
0x00000d0	73 73	77 77	75 75	01 01	01 01	05 05	03 03	07 07
0x00000f0	09 09	13 13	11 11	15 15	65 65	67 67	69 69	71 71
0x0000110	01 01	03 03	05 05	07 07	73 73	75 75	77 77	01 01
0x0000130	09 09	11 11	13 13	15 15	17 17	19 19	21 21	23 23
0x0000150	33 33	35 35	37 37	39 39	25 25	27 27	29 29	31 31
0x0000170	41 41	43 43	45 45	47 47	17 17	21 21	19 19	23 23
0x0000190	33 33	37 37	35 35	39 39	25 25	29 29	27 27	31 31
0x00001b0	41 41	45 45	43 43	47 47	49 49	53 53	51 51	55 55
0x00001d0	65 65	69 69	67 67	71 71	57 57	61 61	59 59	63 63
0x00001f0	73 73	77 77	75 75	03 03	49 49	51 51	57 57	59 59
0x0000210	53 53	55 55	61 61	63 63	65 65	67 67	73 73	75 75
0x0000230	69 69	71 71	77 77	03 03	05 05	07 07	13 13	15 15
0x0000250	09 09	11 11	17 17	19 19	21 21	23 23	29 29	31 31
0x0000270	25 25	27 27	33 33	35 35	05 05	09 09	13 13	17 17
0x0000290	07 07	11 11	15 15	19 19	21 21	25 25	29 29	33 33
0x00002b0	23 23	27 27	31 31	35 35	37 37	41 41	45 45	49 49
0x00002d0	39 39	43 43	47 47	51 51	53 53	57 57	61 61	65 65
0x00002f0	55 55	59 59	63 63	67 67	37 37	39 39	53 53	55 55
0x0000310	41 41	43 43	57 57	59 59	45 45	47 47	61 61	63 63
0x0000330	49 49	51 51	65 65	67 67	69 69	71 71	03 03	05 05
0x0000350	73 73	75 75	07 07	09 09	77 77	01 01	11 11	13 13
0x0000370	03 03	01 01	15 15	17 17	69 69	73 73	03 03	07 07
0x0000390	71 71	75 75	05 05	09 09	77 77	03 03	11 11	15 15
0x00003b0	01 01	01 01	13 13	17 17	19 19	23 23	35 35	39 39
0x00003d0	21 21	25 25	37 37	41 41	27 27	31 31	43 43	47 47
0x00003f0	29 29	33 33	45 45	49 49	19 19	21 21	23 23	25 25

2.2 SECOND SET

Hop sequence {k} for PAGE SCAN/INQUIRY SCAN SUBSTATE:
CLKN start: 0x0000000
ULAP: 0x2a96ef25

#ticks:	0000	1000	2000	3000	4000	5000	6000	7000
0x0000000:	49	13	17	51	55	19	23	53
0x0008000:	57	21	25	27	31	74	78	29
0x0010000:	33	76	1	35	39	3	7	37
0x0018000:	41	5	9	43	47	11	15	45
0x0020000:	49	13	17	51	55	19	23	53
0x0028000:	57	21	25	27	31	74	78	29
0x0030000:	33	76	1	35	39	3	7	37
0x0038000:	41	5	9	43	47	11	15	45

Hop sequence {k} for PAGE STATE/INQUIRY SUBSTATE:
CLKE start: 0x0000000
ULAP: 0x2a96ef25

#ticks:	00 01 02 03	04 05 06 07	08 09 0a 0b	0c 0d 0e 0f
0x0000000:	41 05 10 04	09 43 06 16	47 11 18 12	15 45 14 32
0x0000010:	49 13 34 28	17 51 30 24	55 19 26 20	23 53 22 40
0x0000020:	41 05 10 04	09 43 06 16	47 11 18 12	15 45 14 32
0x0000030:	49 13 34 28	17 51 30 24	55 19 26 20	23 53 22 40
...				
0x0001000:	41 21 10 36	25 27 38 63	31 74 65 59	78 29 61 00
0x0001010:	33 76 02 75	01 35 77 71	39 03 73 67	07 37 69 08
0x0001020:	41 21 10 36	25 27 38 63	31 74 65 59	78 29 61 00
0x0001030:	33 76 02 75	01 35 77 71	39 03 73 67	07 37 69 08
...				
0x0002000:	57 21 42 36	09 43 06 16	47 11 18 12	15 45 14 32
0x0002010:	49 13 34 28	17 51 30 24	55 19 26 20	23 53 22 40
0x0002020:	57 21 42 36	09 43 06 16	47 11 18 12	15 45 14 32
0x0002030:	49 13 34 28	17 51 30 24	55 19 26 20	23 53 22 40
...				
0x0003000:	41 05 10 04	09 27 06 63	31 74 65 59	78 29 61 00
0x0003010:	33 76 02 75	01 35 77 71	39 03 73 67	07 37 69 08
0x0003020:	41 05 10 04	09 27 06 63	31 74 65 59	78 29 61 00
0x0003030:	33 76 02 75	01 35 77 71	39 03 73 67	07 37 69 08

Hop sequence {k} for SLAVE PAGE RESPONSE SUBSTATE:
CLKN* = 0x0000010
ULAP: 0x2a96ef25

#ticks:	00	02 04	06 08	0a 0c	0e 10	12 14	16 18	1a 1c	1e
0x0000012:	34	13 28	17 30	51 24	55 26	19 20	23 22	53 40	57
0x0000032:	42	21 36	25 38	27 63	31 65	74 59	78 61	29 00	33
0x0000052:	02	76 75	01 77	35 71	39 73	03 67	07 69	37 08	41
0x0000072:	10	05 04	09 06	43 16	47 18	11 12	15 14	45 32	49

Hop sequence {k} for MASTER PAGE RESPONSE SUBSTATE:
Offset value: 24
CLKE* = 0x0000012
ULAP: 0x2a96ef25

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000014:	13 28	17 30	51 24	55 26	19 20	23 22	53 40	57 42
0x0000034:	21 36	25 38	27 63	31 65	74 59	78 61	29 00	33 02
0x0000054:	76 75	01 77	35 71	39 73	03 67	07 69	37 08	41 10
0x0000074:	05 04	09 06	43 16	47 18	11 12	15 14	45 32	49 34

Sample Data

Hop sequence {k} for CONNECTION STATE (Basic channel hopping sequence; ie, non-AFH):
CLK start: 0x0000010
ULAP: 0x2a96ef25

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010:	55 26	19 20	23 22	53 40	57 42	21 36	25 38	27 63
0x0000030:	31 65	74 59	78 61	29 00	33 02	76 75	01 77	35 71
0x0000050:	39 73	03 67	07 69	37 08	41 10	05 04	09 06	43 16
0x0000070:	47 18	11 12	15 14	45 32	02 66	47 60	49 64	04 54
0x0000090:	06 58	51 52	53 56	08 70	10 74	55 68	57 72	59 14
0x00000b0:	61 18	27 12	29 16	63 30	65 34	31 28	33 32	67 22
0x00000d0:	69 26	35 20	37 24	71 38	73 42	39 36	41 40	75 46
0x00000f0:	77 50	43 44	45 48	00 62	26 11	69 05	73 07	36 17
0x0000110:	40 19	04 13	08 15	38 25	42 27	06 21	10 23	12 48
0x0000130:	16 50	59 44	63 46	14 56	18 58	61 52	65 54	28 64
0x0000150:	32 66	75 60	00 62	30 72	34 74	77 68	02 70	20 01
0x0000170:	24 03	67 76	71 78	22 09	58 43	24 37	26 41	68 47
0x0000190:	70 51	36 45	38 49	72 55	74 59	40 53	42 57	44 78
0x00001b0:	46 03	12 76	14 01	48 07	50 11	16 05	18 09	60 15
0x00001d0:	62 19	28 13	30 17	64 23	66 27	32 21	34 25	52 31
0x00001f0:	54 35	20 29	22 33	56 39	19 04	62 63	66 00	07 73
0x0000210:	11 10	54 69	58 06	23 75	27 12	70 71	74 08	76 33
0x0000230:	01 49	44 29	48 45	13 35	17 51	60 31	64 47	05 41
0x0000250:	09 57	52 37	56 53	21 43	25 59	68 39	72 55	78 65
0x0000270:	03 02	46 61	50 77	15 67	51 36	17 18	19 34	41 24
0x0000290:	43 40	09 22	11 38	57 28	59 44	25 26	27 42	29 63
0x00002b0:	31 00	76 61	78 77	45 67	47 04	13 65	15 02	37 71
0x00002d0:	39 08	05 69	07 06	53 75	55 12	21 73	23 10	33 16
0x00002f0:	35 32	01 14	03 30	49 20	75 60	39 48	43 56	00 66
0x0000310:	04 74	47 62	51 70	08 68	12 76	55 64	59 72	61 18
0x0000330:	65 26	29 14	33 22	69 20	73 28	37 16	41 24	77 34
0x0000350:	02 42	45 30	49 38	06 36	10 44	53 32	57 40	63 50
0x0000370:	67 58	31 46	35 54	71 52	28 13	73 03	75 11	34 17
0x0000390:	36 25	02 15	04 23	42 21	44 29	10 19	12 27	14 48
0x00003b0:	16 56	61 46	63 54	22 52	24 60	69 50	71 58	30 64
0x00003d0:	32 72	77 62	00 70	38 68	40 76	06 66	08 74	18 01
0x00003f0:	20 09	65 78	67 07	26 05	44 29	32 23	36 25	70 43

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with all channel used; ie, AFH(79)):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x7fffffffffffffffff

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	55 55	19 19	23 23	53 53	57 57	21 21	25 25	27 27
0x0000030	31 31	74 74	78 78	29 29	33 33	76 76	01 01	35 35
0x0000050	39 39	03 03	07 07	37 37	41 41	05 05	09 09	43 43
0x0000070	47 47	11 11	15 15	45 45	02 02	47 47	49 49	04 04
0x0000090	06 06	51 51	53 53	08 08	10 10	55 55	57 57	59 59
0x00000b0	61 61	27 27	29 29	63 63	65 65	31 31	33 33	67 67
0x00000d0	69 69	35 35	37 37	71 71	73 73	39 39	41 41	75 75
0x00000f0	77 77	43 43	45 45	00 00	26 26	69 69	73 73	36 36
0x0000110	40 40	04 04	08 08	38 38	42 42	06 06	10 10	12 12
0x0000130	16 16	59 59	63 63	14 14	18 18	61 61	65 65	28 28
0x0000150	32 32	75 75	00 00	30 30	34 34	77 77	02 02	20 20
0x0000170	24 24	67 67	71 71	22 22	58 58	24 24	26 26	68 68
0x0000190	70 70	36 36	38 38	72 72	74 74	40 40	42 42	44 44
0x00001b0	46 46	12 12	14 14	48 48	50 50	16 16	18 18	60 60
0x00001d0	62 62	28 28	30 30	64 64	66 66	32 32	34 34	52 52
0x00001f0	54 54	20 20	22 22	56 56	19 19	62 62	66 66	07 07
0x0000210	11 11	54 54	58 58	23 23	27 27	70 70	74 74	76 76
0x0000230	01 01	44 44	48 48	13 13	17 17	60 60	64 64	05 05
0x0000250	09 09	52 52	56 56	21 21	25 25	68 68	72 72	78 78
0x0000270	03 03	46 46	50 50	15 15	51 51	17 17	19 19	41 41
0x0000290	43 43	09 09	11 11	57 57	59 59	25 25	27 27	29 29
0x00002b0	31 31	76 76	78 78	45 45	47 47	13 13	15 15	37 37
0x00002d0	39 39	05 05	07 07	53 53	55 55	21 21	23 23	33 33
0x00002f0	35 35	01 01	03 03	49 49	75 75	39 39	43 43	00 00
0x0000310	04 04	47 47	51 51	08 08	12 12	55 55	59 59	61 61
0x0000330	65 65	29 29	33 33	69 69	73 73	37 37	41 41	77 77
0x0000350	02 02	45 45	49 49	06 06	10 10	53 53	57 57	63 63
0x0000370	67 67	31 31	35 35	71 71	28 28	73 73	75 75	34 34
0x0000390	36 36	02 02	04 04	42 42	44 44	10 10	12 12	14 14
0x00003b0	16 16	61 61	63 63	22 22	24 24	69 69	71 71	30 30
0x00003d0	32 32	77 77	00 00	38 38	40 40	06 06	08 08	18 18
0x00003f0	20 20	65 65	67 67	26 26	44 44	32 32	36 36	70 70

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with channels 0 to 21 unused):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x7fffffffffffc00000

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	55 55	50 50	23 23	53 53	57 57	52 52	25 25	27 27
0x0000030	31 31	74 74	78 78	29 29	33 33	76 76	32 32	35 35
0x0000050	39 39	34 34	38 38	37 37	41 41	36 36	40 40	43 43
0x0000070	47 47	42 42	46 46	45 45	55 55	47 47	49 49	57 57
0x0000090	59 59	51 51	53 53	61 61	63 63	55 55	57 57	59 59
0x00000b0	61 61	27 27	29 29	63 63	65 65	31 31	33 33	67 67
0x00000d0	69 69	35 35	37 37	71 71	73 73	39 39	41 41	75 75
0x00000f0	77 77	43 43	45 45	53 53	26 26	69 69	73 73	36 36
0x0000110	40 40	57 57	61 61	38 38	42 42	59 59	63 63	65 65
0x0000130	69 69	59 59	63 63	67 67	71 71	61 61	65 65	28 28
0x0000150	32 32	75 75	53 53	30 30	34 34	77 77	55 55	73 73
0x0000170	24 24	67 67	71 71	22 22	58 58	24 24	26 26	68 68
0x0000190	70 70	36 36	38 38	72 72	74 74	40 40	42 42	44 44
0x00001b0	46 46	65 65	67 67	48 48	50 50	69 69	71 71	60 60
0x00001d0	62 62	28 28	30 30	64 64	66 66	32 32	34 34	52 52
0x00001f0	54 54	73 73	22 22	56 56	37 37	62 62	66 66	25 25
0x0000210	29 29	54 54	58 58	23 23	27 27	70 70	74 74	76 76
0x0000230	76 76	44 44	48 48	31 31	35 35	60 60	64 64	23 23
0x0000250	27 27	52 52	56 56	39 39	25 25	68 68	72 72	78 78
0x0000270	78 78	46 46	50 50	33 33	51 51	35 35	37 37	41 41
0x0000290	43 43	27 27	29 29	57 57	59 59	25 25	27 27	29 29
0x00002b0	31 31	76 76	78 78	45 45	47 47	31 31	33 33	37 37
0x00002d0	39 39	23 23	25 25	53 53	55 55	39 39	23 23	33 33
0x00002f0	35 35	76 76	78 78	49 49	75 75	39 39	43 43	40 40
0x0000310	44 44	47 47	51 51	48 48	52 52	55 55	59 59	61 61
0x0000330	65 65	29 29	33 33	69 69	73 73	37 37	41 41	77 77
0x0000350	42 42	45 45	49 49	46 46	50 50	53 53	57 57	63 63
0x0000370	67 67	31 31	35 35	71 71	28 28	73 73	75 75	34 34
0x0000390	36 36	42 42	44 44	42 42	44 44	50 50	52 52	54 54
0x00003b0	56 56	61 61	63 63	22 22	24 24	69 69	71 71	30 30
0x00003d0	32 32	77 77	40 40	38 38	40 40	46 46	48 48	58 58
0x00003f0	60 60	65 65	67 67	26 26	44 44	32 32	36 36	70 70

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with even channels used):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x55555555555555555555

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	52 52	16 16	20 20	50 50	54 54	18 18	22 22	24 24
0x0000030	28 28	74 74	78 78	26 26	30 30	76 76	78 78	32 32
0x0000050	36 36	00 00	04 04	34 34	38 38	02 02	06 06	40 40
0x0000070	44 44	08 08	12 12	42 42	02 02	44 44	46 46	04 04
0x0000090	06 06	48 48	50 50	08 08	10 10	52 52	54 54	56 56
0x00000b0	58 58	24 24	26 26	60 60	62 62	28 28	30 30	64 64
0x00000d0	66 66	32 32	34 34	68 68	70 70	36 36	38 38	72 72
0x00000f0	74 74	40 40	42 42	00 00	26 26	66 66	70 70	36 36
0x0000110	40 40	04 04	08 08	38 38	42 42	06 06	10 10	12 12
0x0000130	16 16	56 56	60 60	14 14	18 18	58 58	62 62	28 28
0x0000150	32 32	72 72	00 00	30 30	34 34	74 74	02 02	20 20
0x0000170	24 24	64 64	68 68	22 22	58 58	24 24	26 26	68 68
0x0000190	70 70	36 36	38 38	72 72	74 74	40 40	42 42	44 44
0x00001b0	46 46	12 12	14 14	48 48	50 50	16 16	18 18	60 60
0x00001d0	62 62	28 28	30 30	64 64	66 66	32 32	34 34	52 52
0x00001f0	54 54	20 20	22 22	56 56	14 14	62 62	66 66	02 02
0x0000210	06 06	54 54	58 58	18 18	22 22	70 70	74 74	76 76
0x0000230	76 76	44 44	48 48	08 08	12 12	60 60	64 64	00 00
0x0000250	04 04	52 52	56 56	16 16	20 20	68 68	72 72	78 78
0x0000270	78 78	46 46	50 50	10 10	46 46	12 12	14 14	36 36
0x0000290	38 38	04 04	06 06	52 52	54 54	20 20	22 22	24 24
0x00002b0	26 26	76 76	78 78	40 40	42 42	08 08	10 10	32 32
0x00002d0	34 34	00 00	02 02	48 48	50 50	16 16	18 18	28 28
0x00002f0	30 30	76 76	78 78	44 44	70 70	34 34	38 38	00 00
0x0000310	04 04	42 42	46 46	08 08	12 12	50 50	54 54	56 56
0x0000330	60 60	24 24	28 28	64 64	68 68	32 32	36 36	72 72
0x0000350	02 02	40 40	44 44	06 06	10 10	48 48	52 52	58 58
0x0000370	62 62	26 26	30 30	66 66	28 28	68 68	70 70	34 34
0x0000390	36 36	02 02	04 04	42 42	44 44	10 10	12 12	14 14
0x00003b0	16 16	56 56	58 58	22 22	24 24	64 64	66 66	30 30
0x00003d0	32 32	72 72	00 00	38 38	40 40	06 06	08 08	18 18
0x00003f0	20 20	60 60	62 62	26 26	44 44	32 32	36 36	70 70

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with odd channels used):

CLK start: 0x0000010

ULAP: 0x2a96ef25

Used Channels:0x2aaaaaaaaaaaaaaaaaaa

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	55 55	19 19	23 23	53 53	57 57	21 21	25 25	27 27
0x0000030	31 31	77 77	03 03	29 29	33 33	01 01	01 01	35 35
0x0000050	39 39	03 03	07 07	37 37	41 41	05 05	09 09	43 43
0x0000070	47 47	11 11	15 15	45 45	07 07	47 47	49 49	09 09
0x0000090	11 11	51 51	53 53	13 13	15 15	55 55	57 57	59 59
0x00000b0	61 61	27 27	29 29	63 63	65 65	31 31	33 33	67 67
0x00000d0	69 69	35 35	37 37	71 71	73 73	39 39	41 41	75 75
0x00000f0	77 77	43 43	45 45	05 05	31 31	69 69	73 73	41 41
0x0000110	45 45	09 09	13 13	43 43	47 47	11 11	15 15	17 17
0x0000130	21 21	59 59	63 63	19 19	23 23	61 61	65 65	33 33
0x0000150	37 37	75 75	05 05	35 35	39 39	77 77	07 07	25 25
0x0000170	29 29	67 67	71 71	27 27	63 63	29 29	31 31	73 73
0x0000190	75 75	41 41	43 43	77 77	01 01	45 45	47 47	49 49
0x00001b0	51 51	17 17	19 19	53 53	55 55	21 21	23 23	65 65
0x00001d0	67 67	33 33	35 35	69 69	71 71	37 37	39 39	57 57
0x00001f0	59 59	25 25	27 27	61 61	19 19	67 67	71 71	07 07
0x0000210	11 11	59 59	63 63	23 23	27 27	75 75	01 01	03 03
0x0000230	01 01	49 49	53 53	13 13	17 17	65 65	69 69	05 05
0x0000250	09 09	57 57	61 61	21 21	25 25	73 73	77 77	05 05
0x0000270	03 03	51 51	55 55	15 15	51 51	17 17	19 19	41 41
0x0000290	43 43	09 09	11 11	57 57	59 59	25 25	27 27	29 29
0x00002b0	31 31	03 03	05 05	45 45	47 47	13 13	15 15	37 37
0x00002d0	39 39	05 05	07 07	53 53	55 55	21 21	23 23	33 33
0x00002f0	35 35	01 01	03 03	49 49	75 75	39 39	43 43	07 07
0x0000310	11 11	47 47	51 51	15 15	19 19	55 55	59 59	61 61
0x0000330	65 65	29 29	33 33	69 69	73 73	37 37	41 41	77 77
0x0000350	09 09	45 45	49 49	13 13	17 17	53 53	57 57	63 63
0x0000370	67 67	31 31	35 35	71 71	35 35	73 73	75 75	41 41
0x0000390	43 43	09 09	11 11	49 49	51 51	17 17	19 19	21 21
0x00003b0	23 23	61 61	63 63	29 29	31 31	69 69	71 71	37 37
0x00003d0	39 39	77 77	07 07	45 45	47 47	13 13	15 15	25 25
0x00003f0	27 27	65 65	67 67	33 33	51 51	39 39	43 43	77 77

2.3 THIRD SET

```
Hop sequence {k} for PAGE SCAN/INQUIRY SCAN SUBSTATE:
CLKN start:      0x0000000
ULAP:            0x6587cba9
#ticks:          0000 | 1000 | 2000 | 3000 | 4000 | 5000 | 6000 | 7000 |
                 -----------------------------------------------------
0x0000000:         16 |   65 |   67 |   18 |   20 |   53 |   55 |    6 |
0x0008000:          8 |   57 |   59 |   10 |   12 |   69 |   71 |   22 |
0x0010000:         24 |   73 |   75 |   26 |   28 |   45 |   47 |   77 |
0x0018000:          0 |   49 |   51 |    2 |    4 |   61 |   63 |   14 |
0x0020000:         16 |   65 |   67 |   18 |   20 |   53 |   55 |    6 |
0x0028000:          8 |   57 |   59 |   10 |   12 |   69 |   71 |   22 |
0x0030000:         24 |   73 |   75 |   26 |   28 |   45 |   47 |   77 |
0x0038000:          0 |   49 |   51 |    2 |    4 |   61 |   63 |   14 |

Hop sequence {k} for PAGE STATE/INQUIRY SUBSTATE:
CLKE start:      0x0000000
ULAP:            0x6587cba9
#ticks:          00 01 02 03 | 04 05 06 07 | 08 09 0a 0b | 0c 0d 0e 0f |
                 -----------------------------------------------------
0x0000000:       00 49 36 38 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0000010:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
0x0000020:       00 49 36 38 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0000030:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
...
0x0001000:       00 57 36 70 | 59 10 74 72 | 12 69 76 78 | 71 22 03 01 |
0x0001010:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
0x0001020:       00 57 36 70 | 59 10 74 72 | 12 69 76 78 | 71 22 03 01 |
0x0001030:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
...
0x0002000:       08 57 68 70 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0002010:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
0x0002020:       08 57 68 70 | 51 02 42 40 | 04 61 44 46 | 63 14 50 48 |
0x0002030:       16 65 52 54 | 67 18 58 56 | 20 53 60 62 | 55 06 66 64 |
...
0x0003000:       00 49 36 38 | 51 10 42 72 | 12 69 76 78 | 71 22 03 01 |
0x0003010:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |
0x0003020:       00 49 36 38 | 51 10 42 72 | 12 69 76 78 | 71 22 03 01 |
0x0003030:       24 73 05 07 | 75 26 11 09 | 28 45 13 30 | 47 77 34 32 |

Hop sequence {k} for SLAVE PAGE RESPONSE SUBSTATE:
CLKN* =          0x0000010
ULAP:            0x6587cba9
#ticks:          00 | 02 04 | 06 08 | 0a 0c | 0e 10 | 12 14 | 16 18 | 1a 1c | 1e
                 ---------------------------------------------------------------
0x0000012:       52 | 65 54 | 67 58 | 18 56 | 20 60 | 53 62 | 55 66 | 06 64 | 08
0x0000032:       68 | 57 70 | 59 74 | 10 72 | 12 76 | 69 78 | 71 03 | 22 01 | 24
0x0000052:       05 | 73 07 | 75 11 | 26 09 | 28 13 | 45 30 | 47 34 | 77 32 | 00
0x0000072:       36 | 49 38 | 51 42 | 02 40 | 04 44 | 61 46 | 63 50 | 14 48 | 16

Hop sequence {k} for MASTER PAGE RESPONSE SUBSTATE:
Offset value:    24
CLKE* =          0x0000012
ULAP:            0x6587cba9
#ticks:          00 02 | 04 06 | 08 0a | 0c 0e | 10 12 | 14 16 | 18 1a | 1c 1e |
                 -----------------------------------------------------------------
0x0000014:       65 54 | 67 58 | 18 56 | 20 60 | 53 62 | 55 66 | 06 64 | 08 68 |
0x0000034:       57 70 | 59 74 | 10 72 | 12 76 | 69 78 | 71 03 | 22 01 | 24 05 |
0x0000054:       73 07 | 75 11 | 26 09 | 28 13 | 45 30 | 47 34 | 77 32 | 00 36 |
0x0000074:       49 38 | 51 42 | 02 40 | 04 44 | 61 46 | 63 50 | 14 48 | 16 52 |
```

Hop sequence {k} for CONNECTION STATE (Basic channel hopping sequence; ie, non-AFH):
CLK start: 0x0000010
ULAP: 0x6587cba9

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010:	20 60	53 62	55 66	06 64	08 68	57 70	59 74	10 72
0x0000030:	12 76	69 78	71 03	22 01	24 05	73 07	75 11	26 09
0x0000050:	28 13	45 30	47 34	77 32	00 36	49 38	51 42	02 40
0x0000070:	04 44	61 46	63 50	14 48	50 05	16 07	20 09	48 11
0x0000090:	52 13	06 15	10 17	38 19	42 21	08 23	12 25	40 27
0x00000b0:	44 29	22 31	26 33	54 35	58 37	24 39	28 41	56 43
0x00000d0:	60 45	77 62	02 64	30 66	34 68	00 70	04 72	32 74
0x00000f0:	36 76	14 78	18 01	46 03	72 29	42 39	44 43	74 41
0x0000110:	76 45	46 47	48 51	78 49	01 53	50 63	52 67	03 65
0x0000130:	05 69	54 55	56 59	07 57	09 61	58 71	60 75	11 73
0x0000150:	13 77	30 15	32 19	62 17	64 21	34 31	36 35	66 33
0x0000170:	68 37	38 23	40 27	70 25	27 61	72 71	76 73	25 75
0x0000190:	29 77	78 00	03 02	31 04	35 06	01 16	05 18	33 20
0x00001b0:	37 22	07 08	11 10	39 12	43 14	09 24	13 26	41 28
0x00001d0:	45 30	62 47	66 49	15 51	19 53	64 63	68 65	17 67
0x00001f0:	21 69	70 55	74 57	23 59	53 22	35 12	37 28	67 14
0x0000210:	69 30	23 32	25 48	55 34	57 50	39 40	41 56	71 42
0x0000230:	73 58	27 36	29 52	59 38	61 54	43 44	45 60	75 46
0x0000250:	77 62	15 00	17 16	47 02	49 18	31 08	33 24	63 10
0x0000270:	65 26	19 04	21 20	51 06	06 54	65 42	69 58	18 46
0x0000290:	22 62	55 64	59 01	08 68	12 05	71 72	75 09	24 76
0x00002b0:	28 13	57 66	61 03	10 70	14 07	73 74	77 11	26 78
0x00002d0:	30 15	47 32	51 48	00 36	04 52	63 40	67 56	16 44
0x00002f0:	20 60	49 34	53 50	02 38	38 78	12 05	14 13	44 07
0x0000310:	46 15	16 17	18 25	48 19	50 27	24 33	26 41	56 35
0x0000330:	58 43	20 21	22 29	52 23	54 31	28 37	30 45	60 39
0x0000350:	62 47	00 64	02 72	32 66	34 74	08 01	10 09	40 03
0x0000370:	42 11	04 68	06 76	36 70	70 31	42 35	46 43	74 39
0x0000390:	78 47	48 49	52 57	01 53	05 61	56 65	60 73	09 69
0x00003b0:	13 77	50 51	54 59	03 55	07 63	58 67	62 75	11 71
0x00003d0:	15 00	32 17	36 25	64 21	68 29	40 33	44 41	72 37
0x00003f0:	76 45	34 19	38 27	66 23	11 71	05 18	07 22	13 20

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with all channel used; ie, AFH(79)):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x7fffffffffffffffff

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	20 20	53 53	55 55	06 06	08 08	57 57	59 59	10 10
0x0000030	12 12	69 69	71 71	22 22	24 24	73 73	75 75	26 26
0x0000050	28 28	45 45	47 47	77 77	00 00	49 49	51 51	02 02
0x0000070	04 04	61 61	63 63	14 14	50 50	16 16	20 20	48 48
0x0000090	52 52	06 06	10 10	38 38	42 42	08 08	12 12	40 40
0x00000b0	44 44	22 22	26 26	54 54	58 58	24 24	28 28	56 56
0x00000d0	60 60	77 77	02 02	30 30	34 34	00 00	04 04	32 32
0x00000f0	36 36	14 14	18 18	46 46	72 72	42 42	44 44	74 74
0x0000110	76 76	46 46	48 48	78 78	01 01	50 50	52 52	03 03
0x0000130	05 05	54 54	56 56	07 07	09 09	58 58	60 60	11 11
0x0000150	13 13	30 30	32 32	62 62	64 64	34 34	36 36	66 66
0x0000170	68 68	38 38	40 40	70 70	27 27	72 72	76 76	25 25
0x0000190	29 29	78 78	03 03	31 31	35 35	01 01	05 05	33 33
0x00001b0	37 37	07 07	11 11	39 39	43 43	09 09	13 13	41 41
0x00001d0	45 45	62 62	66 66	15 15	19 19	64 64	68 68	17 17
0x00001f0	21 21	70 70	74 74	23 23	53 53	35 35	37 37	67 67
0x0000210	69 69	23 23	25 25	55 55	57 57	39 39	41 41	71 71
0x0000230	73 73	27 27	29 29	59 59	61 61	43 43	45 45	75 75
0x0000250	77 77	15 15	17 17	47 47	49 49	31 31	33 33	63 63
0x0000270	65 65	19 19	21 21	51 51	06 06	65 65	69 69	18 18
0x0000290	22 22	55 55	59 59	08 08	12 12	71 71	75 75	24 24
0x00002b0	28 28	57 57	61 61	10 10	14 14	73 73	77 77	26 26
0x00002d0	30 30	47 47	51 51	00 00	04 04	63 63	67 67	16 16
0x00002f0	20 20	49 49	53 53	02 02	38 38	12 12	14 14	44 44
0x0000310	46 46	16 16	18 18	48 48	50 50	24 24	26 26	56 56
0x0000330	58 58	20 20	22 22	52 52	54 54	28 28	30 30	60 60
0x0000350	62 62	00 00	02 02	32 32	34 34	08 08	10 10	40 40
0x0000370	42 42	04 04	06 06	36 36	70 70	42 42	46 46	74 74
0x0000390	78 78	48 48	52 52	01 01	05 05	56 56	60 60	09 09
0x00003b0	13 13	50 50	54 54	03 03	07 07	58 58	62 62	11 11
0x00003d0	15 15	32 32	36 36	64 64	68 68	40 40	44 44	72 72
0x00003f0	76 76	34 34	38 38	66 66	11 11	05 05	07 07	13 13

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with channels 0 to 21 unused):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x7fffffffffffc00000

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	29 29	53 53	55 55	72 72	74 74	57 57	59 59	76 76
0x0000030	78 78	69 69	71 71	22 22	24 24	73 73	75 75	26 26
0x0000050	28 28	45 45	47 47	77 77	66 66	49 49	51 51	68 68
0x0000070	70 70	61 61	63 63	23 23	50 50	25 25	29 29	48 48
0x0000090	52 52	72 72	76 76	38 38	42 42	74 74	78 78	40 40
0x00000b0	44 44	22 22	26 26	54 54	58 58	24 24	28 28	56 56
0x00000d0	60 60	77 77	68 68	30 30	34 34	66 66	70 70	32 32
0x00000f0	36 36	23 23	27 27	46 46	72 72	42 42	44 44	74 74
0x0000110	76 76	46 46	48 48	78 78	32 32	50 50	52 52	34 34
0x0000130	36 36	54 54	56 56	38 38	40 40	58 58	60 60	42 42
0x0000150	44 44	30 30	32 32	62 62	64 64	34 34	36 36	66 66
0x0000170	68 68	38 38	40 40	70 70	27 27	72 72	76 76	25 25
0x0000190	29 29	78 78	34 34	31 31	35 35	32 32	36 36	33 33
0x00001b0	37 37	38 38	42 42	39 39	43 43	40 40	44 44	41 41
0x00001d0	45 45	62 62	66 66	46 46	50 50	64 64	68 68	48 48
0x00001f0	52 52	70 70	74 74	23 23	53 53	35 35	37 37	67 67
0x0000210	69 69	23 23	25 25	55 55	57 57	39 39	41 41	71 71
0x0000230	73 73	27 27	29 29	59 59	61 61	43 43	45 45	75 75
0x0000250	77 77	46 46	48 48	47 47	49 49	31 31	33 33	63 63
0x0000270	65 65	50 50	52 52	51 51	59 59	65 65	69 69	71 71
0x0000290	22 22	55 55	59 59	61 61	65 65	71 71	75 75	24 24
0x00002b0	28 28	57 57	61 61	63 63	67 67	73 73	77 77	26 26
0x00002d0	30 30	47 47	51 51	53 53	57 57	63 63	67 67	69 69
0x00002f0	73 73	49 49	53 53	55 55	38 38	65 65	67 67	44 44
0x0000310	46 46	69 69	71 71	48 48	50 50	24 24	26 26	56 56
0x0000330	58 58	73 73	22 22	52 52	54 54	28 28	30 30	60 60
0x0000350	62 62	53 53	55 55	32 32	34 34	61 61	63 63	40 40
0x0000370	42 42	57 57	59 59	36 36	70 70	42 42	46 46	74 74
0x0000390	78 78	48 48	52 52	76 76	23 23	56 56	60 60	27 27
0x00003b0	31 31	50 50	54 54	78 78	25 25	58 58	62 62	29 29
0x00003d0	33 33	32 32	36 36	64 64	68 68	40 40	44 44	72 72
0x00003f0	76 76	34 34	38 38	66 66	29 29	23 23	25 25	31 31

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with even channels used):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x55555555555555555

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	20 20	52 52	54 54	06 06	08 08	56 56	58 58	10 10
0x0000030	12 12	68 68	70 70	22 22	24 24	72 72	74 74	26 26
0x0000050	28 28	44 44	46 46	76 76	00 00	48 48	50 50	02 02
0x0000070	04 04	60 60	62 62	14 14	50 50	16 16	20 20	48 48
0x0000090	52 52	06 06	10 10	38 38	42 42	08 08	12 12	40 40
0x00000b0	44 44	22 22	26 26	54 54	58 58	24 24	28 28	56 56
0x00000d0	60 60	76 76	02 02	30 30	34 34	00 00	04 04	32 32
0x00000f0	36 36	14 14	18 18	46 46	72 72	42 42	44 44	74 74
0x0000110	76 76	46 46	48 48	78 78	78 78	50 50	52 52	00 00
0x0000130	02 02	54 54	56 56	04 04	06 06	58 58	60 60	08 08
0x0000150	10 10	30 30	32 32	62 62	64 64	34 34	36 36	66 66
0x0000170	68 68	38 38	40 40	70 70	24 24	72 72	76 76	22 22
0x0000190	26 26	78 78	00 00	28 28	32 32	78 78	02 02	30 30
0x00001b0	34 34	04 04	08 08	36 36	40 40	06 06	10 10	38 38
0x00001d0	42 42	62 62	66 66	12 12	16 16	64 64	68 68	14 14
0x00001f0	18 18	70 70	74 74	20 20	50 50	32 32	34 34	64 64
0x0000210	66 66	20 20	22 22	52 52	54 54	36 36	38 38	68 68
0x0000230	70 70	24 24	26 26	56 56	58 58	40 40	42 42	72 72
0x0000250	74 74	12 12	14 14	44 44	46 46	28 28	30 30	60 60
0x0000270	62 62	16 16	18 18	48 48	06 06	62 62	66 66	18 18
0x0000290	22 22	52 52	56 56	08 08	12 12	68 68	72 72	24 24
0x00002b0	28 28	54 54	58 58	10 10	14 14	70 70	74 74	26 26
0x00002d0	30 30	44 44	48 48	00 00	04 04	60 60	64 64	16 16
0x00002f0	20 20	46 46	50 50	02 02	38 38	12 12	14 14	44 44
0x0000310	46 46	16 16	18 18	48 48	50 50	24 24	26 26	56 56
0x0000330	58 58	20 20	22 22	52 52	54 54	28 28	30 30	60 60
0x0000350	62 62	00 00	02 02	32 32	34 34	08 08	10 10	40 40
0x0000370	42 42	04 04	06 06	36 36	70 70	42 42	46 46	74 74
0x0000390	78 78	48 48	52 52	76 76	00 00	56 56	60 60	04 04
0x00003b0	08 08	50 50	54 54	78 78	02 02	58 58	62 62	06 06
0x00003d0	10 10	32 32	36 36	64 64	68 68	40 40	44 44	72 72
0x00003f0	76 76	34 34	38 38	66 66	06 06	00 00	02 02	08 08

Sample Data

Hop Sequence {k} for CONNECTION STATE (Adapted channel hopping sequence with odd channels used):

CLK start: 0x0000010

ULAP: 0x6587cba9

Used Channels:0x2aaaaaaaaaaaaaaaaaa

#ticks:	00 02	04 06	08 0a	0c 0e	10 12	14 16	18 1a	1c 1e
0x0000010	23 23	53 53	55 55	09 09	11 11	57 57	59 59	13 13
0x0000030	15 15	69 69	71 71	25 25	27 27	73 73	75 75	29 29
0x0000050	31 31	45 45	47 47	77 77	03 03	49 49	51 51	05 05
0x0000070	07 07	61 61	63 63	17 17	53 53	19 19	23 23	51 51
0x0000090	55 55	09 09	13 13	41 41	45 45	11 11	15 15	43 43
0x00000b0	47 47	25 25	29 29	57 57	61 61	27 27	31 31	59 59
0x00000d0	63 63	77 77	05 05	33 33	37 37	03 03	07 07	35 35
0x00000f0	39 39	17 17	21 21	49 49	75 75	45 45	47 47	77 77
0x0000110	01 01	49 49	51 51	03 03	01 01	53 53	55 55	03 03
0x0000130	05 05	57 57	59 59	07 07	09 09	61 61	63 63	11 11
0x0000150	13 13	33 33	35 35	65 65	67 67	37 37	39 39	69 69
0x0000170	71 71	41 41	43 43	73 73	27 27	75 75	01 01	25 25
0x0000190	29 29	03 03	03 03	31 31	35 35	01 01	05 05	33 33
0x00001b0	37 37	07 07	11 11	39 39	43 43	09 09	13 13	41 41
0x00001d0	45 45	65 65	69 69	15 15	19 19	67 67	71 71	17 17
0x00001f0	21 21	73 73	77 77	23 23	53 53	35 35	37 37	67 67
0x0000210	69 69	23 23	25 25	55 55	57 57	39 39	41 41	71 71
0x0000230	73 73	27 27	29 29	59 59	61 61	43 43	45 45	75 75
0x0000250	77 77	15 15	17 17	47 47	49 49	31 31	33 33	63 63
0x0000270	65 65	19 19	21 21	51 51	11 11	65 65	69 69	23 23
0x0000290	27 27	55 55	59 59	13 13	17 17	71 71	75 75	29 29
0x00002b0	33 33	57 57	61 61	15 15	19 19	73 73	77 77	31 31
0x00002d0	35 35	47 47	51 51	05 05	09 09	63 63	67 67	21 21
0x00002f0	25 25	49 49	53 53	07 07	43 43	17 17	19 19	49 49
0x0000310	51 51	21 21	23 23	53 53	55 55	29 29	31 31	61 61
0x0000330	63 63	25 25	27 27	57 57	59 59	33 33	35 35	65 65
0x0000350	67 67	05 05	07 07	37 37	39 39	13 13	15 15	45 45
0x0000370	47 47	09 09	11 11	41 41	75 75	47 47	51 51	01 01
0x0000390	05 05	53 53	57 57	01 01	05 05	61 61	65 65	09 09
0x00003b0	13 13	55 55	59 59	03 03	07 07	63 63	67 67	11 11
0x00003d0	15 15	37 37	41 41	69 69	73 73	45 45	49 49	77 77
0x00003f0	03 03	39 39	43 43	71 71	11 11	05 05	07 07	13 13

3 ACCESS CODE SAMPLE DATA

Different access codes (GIAC, DIACs, others...)

LAP with LSB as rightmost bit.

	Bit transmit order on air		
	---------->		
LAP:	Preamble:	Sync word:	Trailer:
000000	5	7e7041e3 4000000d	5
ffffff	a	e758b522 7ffffff2	a
9e8b33	5	475c58cc 73345e72	a
9e8b34	5	28ed3c34 cb345e72	a
9e8b36	5	62337b64 1b345e72	a
9e8b39	a	c05747b9 e7345e72	a
9e8b3d	5	7084eab0 2f345e72	a
9e8b42	5	64c86d2b 90b45e72	a
9e8b48	a	e3c3725e 04b45e72	a
9e8b4f	a	8c7216a6 bcb45e72	a
9e8b57	a	b2f16c30 fab45e72	a
9e8b60	5	57bd3b22 c1b45e72	a
9e8b6a	a	d0b62457 55b45e72	a
9e8b75	a	81843a39 abb45e72	a
9e8b81	5	0ca96681 e0745e72	a
9e8b8e	a	aecd5a5c 1c745e72	a
9e8b9c	5	17453fbf ce745e72	a
9e8bab	a	f20968ad f5745e72	a
9e8bbb	5	015f4a1e f7745e72	a
9e8bcc	a	d8c695a0 0cf45e72	a
9e8bde	5	614ef043 def45e72	a
9e8bf1	a	ba81ddc7 a3f45e72	a
9e8c05	5	64a7dc4f 680c5e72	a
9e8c1a	5	3595c221 960c5e72	a
9e8c30	a	cb35cc0d 830c5e72	a
9e8c47	5	12ac13b3 788c5e72	a
9e8c5f	5	2c2f6925 3e8c5e72	a
9e8c78	5	3a351c84 078c5e72	a
9e8c92	5	7396d0f3 124c5e72	a
9e8cad	5	5b0fdfc4 6d4c5e72	a
9e8cc9	a	aea2eb38 e4cc5e72	a
9e8ce6	5	756dc6bc 99cc5e72	a
9e8d04	5	214cf934 882c5e72	a
9e8d23	5	37568c95 b12c5e72	a
9e8d43	5	72281560 f0ac5e72	a
9e8d64	5	643260c1 c9ac5e72	a
9e8d86	a	e044f493 986c5e72	a
9e8da9	5	3b8bd917 e56c5e72	a
9e8dcd	a	ce26edeb 6cec5e72	a
9e8df2	a	e6bfe2dc 13ec5e72	a
9e8e18	a	82dcde3d c61c5e72	a
9e8e3f	a	94c6ab9c ff1c5e72	a

Sample Data

9e8e67	a	969059a6 799c5e72	a
9e8e90	a	c4dfccef 425c5e72	a
9e8eba	5	3a7fc2c3 575c5e72	a
9e8ee5	5	57985401 69dc5e72	a
9e8f11	5	0ae2a363 623c5e72	a
9e8f3e	a	d12d8ee7 1f3c5e72	a
9e8f6c	5	547063a8 0dbc5e72	a
9e8f9b	5	063ff6e1 367c5e72	a
9e8fcb	a	c9bc5cfe f4fc5e72	a
9e8ffc	5	2cf00bec cffc5e72	a
9e902e	a	8ec5052f 5d025e72	a
9e9061	5	1074b15e 61825e72	a
9e9095	a	9d59ede6 2a425e72	a
9e90ca	a	f0be7b24 14c25e72	a
9e9100	5	10e10dd0 c0225e72	a
9e9137	a	f5ad5ac2 fb225e72	a
9e916f	a	f7fba8f8 7da25e72	a
9e91a8	5	2f490e5b c5625e72	a
9e91e2	a	94979982 91e25e72	a
9e921d	5	26cda478 2e125e72	a
9e9259	a	aacb81dd 26925e72	a
9e9296	a	bfac7f5b da525e72	a
9e92d4	a	c9a7b0a7 cad25e72	a
9e9313	a	c142bdde 32325e72	a
616cec	5	586a491f 0dcda18d	5
616ceb	5	37db2de7 b5cda18d	5
616ce9	5	7d056ab7 65cda18d	5
616ce6	a	df61566a 99cda18d	5
616ce2	5	6fb2fb63 51cda18d	5
616cdd	5	472bf454 2ecda18d	5
616cd7	a	c020eb21 bacda18d	5
616cd0	a	af918fd9 02cda18d	5
616cc8	a	9112f54f 44cda18d	5
616cbf	5	488b2af1 bf4da18d	5
616cb5	a	cf803584 2b4da18d	5
616caa	a	9eb22bea d54da18d	5
616c9e	a	a49cb509 9e4da18d	5
616c91	5	06f889d4 624da18d	5
616c83	a	bf70ec37 b04da18d	5
616c74	a	ed3f797e 8b8da18d	5
616c64	5	1e695bcd 898da18d	5
616c53	a	fb250cdf b28da18d	5
616c41	5	42ad693c 608da18d	5
616c2e	a	a5b7cc14 dd0da18d	5
616c1a	a	9f9952f7 960da18d	5
616c05	a	ceab4c99 680da18d	5
616bef	a	d403ddde fdf5a18d	5
616bd8	5	314f8acc c6f5a18d	5
616bc0	5	0fccf05a 80f5a18d	5
616ba7	5	25030d57 7975a18d	5
616b8d	a	dba3037b 6c75a18d	5
616b72	5	4439ce17 13b5a18d	5

Sample Data

616b56	a	8d417247 5ab5a18d	5
616b39	5	6a5bd76f e735a18d	5
616b1b	5	592e8166 b635a18d	5
616afc	5	28609d46 cfd5a18d	5
616adc	5	51cb8c1f 4ed5a18d	5
616abb	5	7b047112 b755a18d	5
616a99	5	4871271b e655a18d	5
616a76	5	24bdc8c4 9b95a18d	5
616a52	a	edc57494 d295a18d	5
616a2d	a	f989f30f 6d15a18d	5
616a07	5	0729fd23 7815a18d	5
6169e0	a	8bf0ba4f 81e5a18d	5
6169b8	a	89a64875 0765a18d	5
61698f	5	6cea1f67 3c65a18d	5
616965	5	2549d310 29a5a18d	5
61693a	5	48ae45d2 1725a18d	5
61690e	5	7280db31 5c25a18d	5
6168e1	a	ce1b9f34 61c5a18d	5
6168b3	5	4b46727b 7345a18d	5
616884	a	ae0a2569 4845a18d	5
616854	a	ea5fc581 4a85a18d	5
616823	5	33c61a3f b105a18d	5
6167f1	a	c49fb8c5 63f9a18d	5
6167be	5	5a2e0cb4 5f79a18d	5
61678a	5	60009257 1479a18d	5
616755	a	86314e62 eab9a18d	5
61671f	5	3defd9bb be39a18d	5
6166e8	a	bff7e728 c5d9a18d	5
6166b0	a	bda11512 4359a18d	5
616677	5	6513b3b1 fb99a18d	5
61663d	a	decd2468 af19a18d	5
616602	a	f6542b5f d019a18d	5
6165c6	a	dc44b49b d8e9a18d	5
616589	5	42f500ea e469a18d	5
61654b	a	bf2885e1 34a9a18d	5
61650c	a	ec4c69b5 4c29a18d	5

4 HEC AND PACKET HEADER SAMPLE DATA

This section contains examples of HECs computed for sample UAP and packet header contents (Data). The resulting 54 bit packet headers are shown in the rightmost column. Note that the UAP, Data and HEC values are in hexadecimal notation, while the header is in octal notation. The header is transmitted from left to right over the air.

UAP	Data	HEC	Header (octal)
00	123	e1	770007 007070 000777
47	123	06	770007 007007 700000
00	124	32	007007 007007 007700
47	124	d5	007007 007070 707077
00	125	5a	707007 007007 077070
47	125	bd	707007 007070 777707
00	126	e2	077007 007007 000777
47	126	05	077007 007070 700000
00	127	8a	777007 007007 070007
47	127	6d	777007 007070 770770
00	11b	9e	770770 007007 777007
47	11b	79	770770 007070 077770
00	11c	4d	007770 007070 770070
47	11c	aa	007770 007007 070707
00	11d	25	707770 007070 700700
47	11d	c2	707770 007007 000077
00	11e	9d	077770 007070 777007
47	11e	7a	077770 007007 077770
00	11f	f5	777770 007070 707777
47	11f	12	777770 007007 007000

5 CRC SAMPLE DATA

This section shows the CRC computed for a sample 10 byte payload and a UAP of 0x47.

Data:
-----

data[0] = 0x4e
data[1] = 0x01
data[2] = 0x02
data[3] = 0x03
data[4] = 0x04
data[5] = 0x05
data[6] = 0x06
data[7] = 0x07
data[8] = 0x08
data[9] = 0x09

UAP = 0x47

==> CRC = 6d d2

Codeword (hexadecimal notation):

4e 01 02 03 04 05 06 07 08 09 6d d2

NB: Over the air each byte in the codeword
    is sent with the LSB first.

6 COMPLETE SAMPLE PACKETS

6.1 EXAMPLE OF DH1 PACKET

```
Packet header: (MSB...LSB)
--------------------------
LT_ADDR = 011
TYPE = 0100 (DH1)
FLOW = 0
ARQN = 1
SEQN = 0

Payload: (MSB...LSB)
--------------------
payload length: 5 bytes
logical channel = 10 (UA/I, Start L2CAP message)
flow = 1
data byte 1 = 00000001
data byte 2 = 00000010
data byte 3 = 00000011
data byte 4 = 00000100
data byte 5 = 00000101

HEC and CRC initialization: (MSB...LSB)
---------------------------------------
uap = 01000111

NO WHITENING USED




AIR DATA (LSB...MSB)

Packet header (incl HEC):
-------------------------
111111000
000000111000
000111000
000111111000000000000000

Payload (incl payload header and CRC):
--------------------------------------------------------------
01110100
10000000
01000000
11000000
00100000
10100000
1110110000110110
```

6.2 EXAMPLE OF DM1 PACKET

```
Packet header: (MSB...LSB)
--------------------------
LT_ADDR = 011
TYPE = 0011 (DM1)
FLOW = 0
ARQN = 1
SEQN = 0

Payload: (MSB...LSB)
--------------------
payload length: 5 bytes
logical channel = 10 (UA/I, Start L2CAP message)
flow = 1
data byte 1 = 00000001
data byte 2 = 00000010
data byte 3 = 00000011
data byte 4 = 00000100
data byte 5 = 00000101

HEC and CRC initialization: (MSB...LSB)
---------------------------------------
uap = 01000111

NO WHITENING USED




AIR DATA (LSB...MSB)

Packet header (incl HEC):
-------------------------
111111000
111111000000
000111000
111000000111111111111000

Payload (incl payload header, FEC23, CRC and 6 padded zeros):
-------------------------------------------------------------
0111010010 11001
0000000100 01011
0000110000 11110
0000100000 00111
1010000011 01100
1011000011 00010
0110000000 10001
```

7 WHITENING SEQUENCE SAMPLE DATA

This section shows the output of the whitening sequence generator.

Whitening Sequence (=D7)	Whitening LFSR D7.....D0
1	1111111
1	1101111
1	1001111
0	0001111
0	0011110
0	0111100
1	1111000
1	1100001
1	1010011
0	0110111
1	1101110
1	1001101
0	0001011
0	0010110
0	0101100
1	1011000
0	0100001
1	1000010
0	0010101
0	0101010
1	1010100
0	0111001
1	1110010
1	1110101
1	1111011
1	1100111
1	1011111
0	0101111
1	1011110
0	0101101
1	1011010
0	0100101
1	1001010
0	0000101
0	0001010
0	0010100
0	0101000
1	1010000
0	0110001
1	1100010
1	1010101
0	0111011
1	1110110

Sample Data

1	1111101
1	1101011
1	1000111
0	0011111
0	0111110
1	1111100
1	1101001
1	1000011
0	0010111
0	0101110
1	1011100
0	0101001
1	1010010
0	0110101
1	1101010
1	1000101
0	0011011
0	0110110
1	1101100
1	1001001
0	0000011
0	0000110
0	0001100
0	0011000
0	0110000
1	1100000
1	1010001
0	0110011
1	1100110
1	1011101
0	0101011
1	1010110
0	0111101
1	1111010
1	1100101
1	1011011
0	0100111
1	1001110
0	0001101
0	0011010
0	0110100
1	1101000
1	1000001
0	0010011
0	0100110
1	1001100
0	0001001
0	0010010
0	0100100
1	1001000
0	0000001
0	0000010

Sample Data

0	0000100
0	0001000
0	0010000
0	0100000
1	1000000
0	0010001
0	0100010
1	1000100
0	0011001
0	0110010
1	1100100
1	1011001
0	0100011
1	1000110
0	0011101
0	0111010
1	1110100
1	1111001
1	1100011
1	1010111
0	0111111
1	1111110
1	1101101
1	1001011
0	0000111
0	0001110
0	0011100
0	0111000
1	1110000
1	1110001
1	1110011
1	1110111
1	1111111

8 FEC SAMPLE DATA

```
================================================
 Rate 2/3 FEC -- (15,10) Shortened Hamming Code
================================================

Data is in hexadecimal notation, the codewords are in binary notation. The codeword bits
are sent from left to right over the air interface. The space in the codeword indicates
the start of parity bits.

Data:           Codeword:

0x001       1000000000 11010
0x002       0100000000 01101
0x004       0010000000 11100
0x008       0001000000 01110
0x010       0000100000 00111
0x020       0000010000 11001
0x040       0000001000 10110
0x080       0000000100 01011
0x100       0000000010 11111
0x200       0000000001 10101
```

9 ENCRYPTION KEY SAMPLE DATA

Explanation:

For the sections 9.1 - 9.5, the hexadecimal sample data is written with the least significant byte at the leftmost position and the most significant byte at the rightmost position. Within each byte, the *least significant bit* (LSB) is at the rightmost position and the *most significant bit* (MSB) is at the leftmost position Thus, a line reading:

aco: 48afcdd4bd40fef76693b113

means aco[0]=0x48, ac[1]=0xaf, ..., aco[11]=0x13. The LSB of aco[11] is '1' and the MSB of aco[11] is '0'.

Key [i]: denotes the ith sub-key in Ar or A'r;
round r: denotes the input to the rth round;
added ->: denotes the input to round 3 in A'r after adding original input (of round 1).

9.1 FOUR TESTS OF E1

```
rand    :00000000000000000000000000000000
address :000000000000
key     :00000000000000000000000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000000
Key [ 2]:4697b1baa3b7100ac537b3c95a28ac64
round  2:78d19f9307d2476a523ec7a8a026042a
Key [ 3]:ecabaac66795580df89af66e66dc053d
Key [ 4]:8ac3d8896ae9364943bfebd4969b68a0
round  3:600265247668dda0e81c07bbb30ed503
Key [ 5]:5d57921fd5715cbb22c1be7bbc996394
Key [ 6]:2a61b8343219fdfb1740e6511d41448f
round  4:d7552ef7cc9dbde568d80c2215bc4277
Key [ 7]:dd0480dee731d67f01a2f739da6f23ca
Key [ 8]:3ad01cd1303e12a1cd0fe0a8af82592c
round  5:fb06bef32b52ab8f2a4f2b6ef7f6d0cd
Key [ 9]:7dadb2efc287ce75061302904f2e7233
Key [10]:c08dcfa981e2c4272f6c7a9f52e11538
round  6:b46b711ebb3cf69e847a75f0ab884bdd
Key [11]:fc2042c708e409555e8c147660ffdfd7
Key [12]:fa0b21001af9a6b9e89e624cd99150d2
round  7:c585f308ff19404294f06b292e978994
Key [13]:18b40784ea5ba4c80ecb48694b4e9c35
Key [14]:454d54e5253c0c4a8b3fcca7db6baef4
round  8:2665fadb13acf952bf74b4ab12264b9f
Key [15]:2df37c6d9db52674f29353b0f011ed83
Key [16]:b60316733b1e8e70bd861b477e2456f1
Key [17]:884697b1baa3b7100ac537b3c95a28ac
```

Sample Data

```
round  1:158ffe43352085e8a5ec7a88e1ff2ba8
Key [ 1]:e9e5dfc1b3a79583e9e5dfc1b3a79583
Key [ 2]:7595bf57e0632c59f435c16697d4c864
round  2:0b5cc75febcdf7827ca29ec0901b6b5b
Key [ 3]:e31b96afcc75d286ef0ae257cbbc05b7
Key [ 4]:0d2a27b471bc0108c6263aff9d9b3b6b
round  3:e4278526c8429211f7f2f0016220aef4
added ->:f1b68365fd6217f952de6a89831fd95c
Key [ 5]:98d1eb5773cf59d75d3b17b3bc37c191
Key [ 6]:fd2b79282408ddd4ea0aa7511133336f
round  4:d0304ad18337f86040145d27aa5c8153
Key [ 7]:331227756638a41d57b0f7e071ee2a98
Key [ 8]:aa0dd8cc68b406533d0f1d64aabacf20
round  5:84db909d213bb0172b8b6aaf71bf1472
Key [ 9]:669291b0752e63f806fce76f10e119c8
Key [10]:ef8bdd46be8ee0277e9b78adef1ec154
round  6:f835f52921e903dfa762f1df5abd7f95
Key [11]:f3902eb06dc409cfd78384624964bf51
Key [12]:7d72702b21f97984a721c99b0498239d
round  7:ae6c0b4bb09f25c6a5d9788a31b605d1
Key [13]:532e60bceaf902c52a06c2c283ecfa32
Key [14]:181715e5192efb2a64129668cf5d9dd4
round  8:744a6235b86cc0b853cc9f74f6b65311
Key [15]:83017c1434342d4290e961578790f451
Key [16]:2603532f365604646ff65803795ccce5
Key [17]:882f7c907b565ea58dae1c928a0dcf41
sres    :056c0fe6
aco     :48afcdd4bd40fef76693b113
----------------------------------------
rand    :bc3f30689647c8d7c5a03ca80a91eceb
address :7ca89b233c2d
key     :159dd9f43fc3d328efba0cd8a861fa57
round  1:bc3f30689647c8d7c5a03ca80a91eceb
Key [ 1]:159dd9f43fc3d328efba0cd8a861fa57
Key [ 2]:326558b3c15551899a97790e65ff669e
round  2:3e950edf197615638cc19c09f8fedc9b
Key [ 3]:62e879b65b9f53bbfbd020c624b1d682
Key [ 4]:73415f30bac8ab61f410adc9442992db
round  3:6a7640791cb536678936c5ecd4ae5a73
Key [ 5]:5093cfa1d31c1c48acd76df030ea3c31
Key [ 6]:0b4acc2b8f1f694fc7bd91f4a70f3009
round  4:fca2c022a577e2ffb2aa007589693ec7
Key [ 7]:2ca43fc817947804ecff148d50d6f6c6
Key [ 8]:3fcd73524b533e00b7f7825bea2040a4
round  5:e97f8ea4ed1a6f4a36ffc179dc6bb563
Key [ 9]:6c67bec76ae8c8cc4d289f69436d3506
Key [10]:95ed95ee8cb97e61d75848464bffb379
round  6:38b07261d7340d028749de1773a415c7
Key [11]:ff566c1fc6b9da9ac502514550f3e9d2
Key [12]:ab5ce3f5c887d0f49b87e0d380e12f47
round  7:58241f1aed7c1c3e047d724331a0b774
Key [13]:a2cab6f95eac7d655dbe84a6cd4c47f5
```

Sample Data

```
Key [14]:f5caff88af0af8c42a20b5bbd2c8b460
round  8:3d1aaeff53c0910de63b9788b13c490f
Key [15]:185099c1131cf97001e2f36fda415025
Key [16]:a0ebb82676bc75e8378b189eff3f6b1d
Key [17]:cf5b348aaee27ae332b4f1bfa10289a6
round  1:2e4b417b9a2a9cfd7d8417d9a6a556eb
Key [ 1]:fe78b835f26468ab069fd3991b086fda
Key [ 2]:095c5a51c6fa6d3ac1d57fa19aa382bd
round  2:b8bca81d6bb45af9d92beadd9300f5ed
Key [ 3]:1af866df817fd9f4ec00bc704192cffc
Key [ 4]:f4a8a059c1f575f076f5fbb24bf16590
round  3:351aa16dec2c3a4787080249ed323eae
added ->:1b65e2167656d6bafa8c19904bd79445
Key [ 5]:8c9d18d9356a9954d341b4286e88ea1f
Key [ 6]:5c958d370102c9881bf753e69c7da029
round  4:2ce8fef47dda6a5bee74372e33e478a2
Key [ 7]:7eb2985c3697429fbe0da334bb51f795
Key [ 8]:af900f4b63a1138e2874bfb7c628b7b8
round  5:572787f563e1643c1c862b7555637fb4
Key [ 9]:834c8588dd8f3d4f31117a488420d69b
Key [10]:bc2b9b81c15d9a80262f3f48e9045895
round  6:16b4968c5d02853c3a43aa4cdb5f26ac
Key [11]:f08608c9e39ad3147cba61327919c958
Key [12]:2d4131decf4fa3a959084714a9e85c11
round  7:10e4120c7cccef9dd4ba4e6da8571b01
Key [13]:c934fd319c4a2b5361fa8eef05ae9572
Key [14]:4904c17aa47868e40471007cde3a97c0
round  8:f9081772498fed41b6ffd72b71fcf6c6
Key [15]:ea5e28687e97fa3f833401c86e6053ef
Key [16]:1168f58252c4ecfccafbdb3af857b9f2
Key [17]:b3440f69ef951b78b5cbd6866275301b
sres    :8d5205c5
aco     :3ed75df4abd9af638d144e94
----------------------------------------
rand    :0891caee063f5da1809577ff94ccdcfb
address :c62f19f6ce98
key     :45298d06e46bac21421ddfbed94c032b
round  1:0891caee063f5da1809577ff94ccdcfb
Key [ 1]:45298d06e46bac21421ddfbed94c032b
Key [ 2]:8f03e1e1fe1c191cad35a897bc400597
round  2:1c6ca013480a685c1b28e0317f7167e1
Key [ 3]:4f2ce3a092dde854ef496c8126a69e8e
Key [ 4]:968caee2ac6d7008c07283daec67f2f2
round  3:06b4915f5fcc1fc551a52048f0af8a26
Key [ 5]:ab0d5c31f94259a6bf85ee2d22edf56c
Key [ 6]:dfb74855c0085ce73dc17b84bfd50a92
round  4:077a92b040acc86e6e0a877db197a167
Key [ 7]:8f888952662b3db00d4e904e7ea53b5d
Key [ 8]:5e18bfcc07799b0132db88cd6042f599
round  5:7204881fb300914825fdc863e8ceadf3
Key [ 9]:bfca91ad9bd3d1a06c582b1d5512dddf
Key [10]:a88bc477e3fa1d5a59b5e6cf793c7a41
```

Sample Data

```
round  6:27031131d86cea2d747deb4f756143aa
Key [11]:f3cfb8dac8aea2a6a8ef95af3a2a2767
Key [12]:77beb90670c5300b03aa2b2232d3d40c
round  7:fc8c13d49149b1ce8d86f96e44a00065
Key [13]:b578373650af36a06e19fe335d726d32
Key [14]:6bcee918c7d0d24dfdf42237fcf99d53
round  8:04ef5f5a7ddf846cda0a07782fc23866
Key [15]:399f158241eb3e079f45d7b96490e7ea
Key [16]:1bcfbe98ecde2add52aa63ea79fb917a
Key [17]:ee8bc03ec08722bc2b075492873374af
round  1:d989d7a40cde7032d17b52f8117b69d5
Key [ 1]:2ecc6cc797cc41a2ab02007f6af396ae
Key [ 2]:acfaef7609c12567d537ae1cf9dc2198
round  2:8e76eb9a29b2ad5eea790db97aee37c1
Key [ 3]:079c8ff9b73d428df879906a0b87a6c8
Key [ 4]:19f2710baf403a494193d201f3a8c439
round  3:346bb7c35b2539676375aafe3af69089
added ->:edf48e675703a955b2f0fc062b71f95c
Key [ 5]:d623a6498f915cb2c8002765247b2f5a
Key [ 6]:900109093319bc30108b3d9434a77a72
round  4:fafb6c1f3ebbd2477be2da49dd923f69
Key [ 7]:e28e2ee6e72e7f4e5b5c11f10d204228
Key [ 8]:8e455cd11f8b9073a2dfa5413c7a4bc5
round  5:7c72230df588060a3cf920f9b0a08f06
Key [ 9]:28afb26e2c7a64238c41cefc16c53e74
Key [10]:d08dcafc2096395ba0d2dddd0e471f4d
round  6:55991df991db26ff00073a12baa3031d
Key [11]:fcffdcc3ad8faae091a7055b934f87c1
Key [12]:f8df082d77060252c02d91e55bd6a7d6
round  7:70ec682ff864375f63701fa4f6be5377
Key [13]:bef3706e523d708e8a44147d7508bc35
Key [14]:3e98ab283ca2422d56a56cf8b06caeb3
round  8:172f12ec933da85504b4ea5c90f8f0ea
Key [15]:87ad9625d06645d22598dd5ef811ea2c
Key [16]:8bd3db0cc8168009e5da90877e13a36f
Key [17]:0e74631d813a8351ac7039b348c41b42
sres    :00507e5f
aco     :2a5f19fbf60907e69f39ca9f
----------------------------------------
rand    :0ecd61782b4128480c05dc45542b1b8c
address :f428f0e624b3
key     :35949a914225fabad91995d226de1d92
round  1:0ecd61782b4128480c05dc45542b1b8c
Key [ 1]:35949a914225fabad91995d226de1d92
Key [ 2]:ea6b3dcccc8ee5d88de349fa5010404f
round  2:8935e2e263fbc4b9302cabdfc06bce3e
Key [ 3]:920f3a0f2543ce535d4e7f25ad80648a
Key [ 4]:ad47227edf9c6874e80ba80ebb95d2c9
round  3:b4c8b878675f184a0c72f3dab51f8f05
Key [ 5]:81a941ca7202b5e884ae8fa493ecac3d
Key [ 6]:bcde1520bee3660e86ce2f0fb78b9157
round  4:77ced9f2fc42bdd5c6312b87fc2377c5
```

Sample Data

```
Key [ 7]:c8eee7423d7c6efa75ecec0d2cd969d3
Key [ 8]:910b3f838a02ed441fbe863a02b4a1d0
round  5:fe28e8056f3004d60bb207e628b39cf2
Key [ 9]:56c647c1e865eb078348962ae070972d
Key [10]:883965da77ca5812d8104e2b640aec0d
round  6:1f2ba92259d9e88101518f145a33840f
Key [11]:61d4cb7e4f8868a283327806a9bd8d4d
Key [12]:9f57de3a3ff310e21dc1e696ce060304
round  7:cc9b5d0218d29037e88475152ebebb2f
Key [13]:7aa1d8adc1aeed7127ef9a18f6eb2d8e
Key [14]:b4db9da3bf865912acd14904c7f7785d
round  8:b04d352bedc02682e4a7f59d7cda1dba
Key [15]:a13d7141ef1f6c7d867e3d175467381b
Key [16]:08b2bc058e50d6141cdd566a307e1acc
Key [17]:057b2b4b4be5dc0ac49e50489b8006c9
round  1:5cfacc773bae995cd7f1b81e7c9ec7df
Key [ 1]:1e717950f5828f3930fe4a9395858815
Key [ 2]:d1623369b733d98bbc894f75866c544c
round  2:d571ffa21d9daa797b1a0a3c962fc64c
Key [ 3]:4abf27664ae364cc8a7e5bcf88214cc4
Key [ 4]:2aaedda8dc4933dd6aeaf6e5c0d5a482
round  3:e17c8e498a00f125bf654c938c23f36d
added ->:bd765a3eb1ae8a796856048df0c1bab2
Key [ 5]:bc7f8ab2d86000f47b1946cc8d7a7a2b
Key [ 6]:6b28544cb13ec6c5d98470df2cf900b7
round  4:a9727c26f2f06bd9920e83c8605dcd76
Key [ 7]:1be840d9107f2c9523f66bb19f5464a1
Key [ 8]:61d6fb1aa2f0c2b26fb2a3d6de8c177c
round  5:aeff751f146eab7e4626b2e2c9e2fb39
Key [ 9]:adabfc82570c568a233173099f23f4c2
Key [10]:b7df6b55ad266c0f1ff7452101f59101
round  6:cf412b95f454d5185e67ca671892e5bd
Key [11]:8e04a7282a2950dcbaea28f300e22de3
Key [12]:21362c114433e29bda3e4d51f803b0cf
round  7:16165722fe4e07ef88f8056b17d89567
Key [13]:710c8fd5bb3cbb5f132a7061de518bd9
Key [14]:0791de7334f4c87285809343f3ead3bd
round  8:28854cd6ad4a3c572b15490d4b81bc3f
Key [15]:4f47f0e5629a674bfcd13770eb3a3bd9
Key [16]:58a6d9a16a284cc0aead2126c79608a1
Key [17]:a564082a0a98399f43f535fd5cefad34
sres    :80e5629c
aco     :a6fe4dcde3924611d3cc6ba1


========================================
```

9.2 FOUR TESTS OF E21

```
rand    :00000000000000000000000000000000
address :000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000006
Key [ 2]:4697b1baa3b7100ac537b3c95a28dc94
round  2:98611307ab76bbde9a86af1ce8cad412
Key [ 3]:ecabaac66795580df89af66e665d863d
Key [ 4]:8ac3d8896ae9364943bfebd4a2a768a0
round  3:820999ad2e6618f4b578974beeedf9e7
added ->:820999ad2e6618f4b578974beeedf9e7
Key [ 5]:5d57921fd5715cbb22c1bedb1c996394
Key [ 6]:2a61b8343219fdfb1740e9541d41448f
round  4:acd6edec87581ac22dbdc64ea4ced3a2
Key [ 7]:dd0480dee731d67f01ba0f39da6f23ca
Key [ 8]:3ad01cd1303e12a18dcfe0a8af82592c
round  5:1c7798732f09fbfe25795a4a2fbc93c2
Key [ 9]:7dadb2efc287ce7b0c1302904f2e7233
Key [10]:c08dcfa981e2f4572f6c7a9f52e11538
round  6:c05b88b56aa70e9c40c79bb81cd911bd
Key [11]:fc2042c708658a555e8c147660ffdfd7
Key [12]:fa0b21002605a6b9e89e624cd99150d2
round  7:abacc71b481c84c798d1bdf3d62f7e20
Key [13]:18b407e44a5ba4c80ecb48694b4e9c35
Key [14]:454d57e8253c0c4a8b3fcca7db6baef4
round  8:e8204e1183ae85cf19edb2c86215b700
Key [15]:2d0b946d9db52674f29353b0f011ed83
Key [16]:76c316733b1e8e70bd861b477e2456f1
Key [17]:8e4697b1baa3b7100ac537b3c95a28ac
Ka      :d14ca028545ec262cee700e39b5c39ee
----------------------------------------
rand    :2dd9a550343191304013b2d7e1189d09
address :cac4364303b6
round  1:cac4364303b6cac4364303b6cac43643
Key [ 1]:2dd9a550343191304013b2d7e1189d0f
Key [ 2]:14c4335b2c43910c5dcc71d81a14242b
round  2:e169f788aad45a9011f11db5270b1277
Key [ 3]:55bfb712cba168d1a48f6e74cd9f4388
Key [ 4]:2a2b3aacca695caef2821b0fb48cc253
round  3:540f9c76652e92c44987c617035037bf
added ->:9ed3d23566e45c007fcac9a1c9146dfc
Key [ 5]:a06aab22d9a287384042976b4b6b00ee
Key [ 6]:c229d054bb72e8eb230e6dcdb32d16b7
round  4:83659a41675f7171ea57909dc5a79ab4
Key [ 7]:23c4812ab1905ddf77dedaed4105649a
Key [ 8]:40d87e272a7a1554ae2e85e3638cdf52
round  5:0b9382d0ed4f2fccdbb69d0db7b130a4
Key [ 9]:bdc064c6a39f6b84fe40db359f62a3c4
Key [10]:58228db841ce3cee983aa721f36aa1b9
round  6:c6ebda0f8f489792f09c189568226c1f
Key [11]:a815bacd6fa747a0d4f52883ac63ebe7
```

Sample Data

```
Key [12]:a9ce513b38ea006c333ecaaefcf1d0f8
round  7:75a8aba07e69c9065bcd831c40115116
Key [13]:3635e074792d4122130e5b824e52cd60
Key [14]:511bdb61bb28de72a5d794bffbf407df
round  8:57a6e279dcb764cf7dd6a749dd60c735
Key [15]:a32f5f21044b6744b6d913b13cdb4c0a
Key [16]:9722bbaeef281496ef8c23a9d41e92f4
Key [17]:807370560ad7e8a13a054a65a03b4049
Ka      :e62f8bac609139b3999aedbc9d228042
----------------------------------------
rand    :dab3cffe9d5739d1b7bf4a667ae5ee24
address :02f8fd4cd661
round  1:02f8fd4cd66102f8fd4cd66102f8fd4c
Key [ 1]:dab3cffe9d5739d1b7bf4a667ae5ee22
Key [ 2]:e315a8a65d809ec7c289e69c899fbdcc
round  2:ef85ff081b8709405e19f3e275cec7dc
Key [ 3]:df6a119bb50945fc8a3394e7216448f3
Key [ 4]:87fe86fb0d58b5dd0fb3b6b1dab51d07
round  3:aa25c21bf577d92dd97381e3e9edcc54
added ->:a81dbf5723d8dbd524bf5782ebe5c918
Key [ 5]:36cc253c506c0021c91fac9d8c469e90
Key [ 6]:d5fda00f113e303809b7f7d78a1a2b0e
round  4:9e69ce9b53caec3990894d2baed41e0d
Key [ 7]:c14b5edc10cabf16bc2a2ba4a8ae1e40
Key [ 8]:74c6131afc8dce7e11b03b1ea8610c16
round  5:a5460fa8cedca48a14fd02209e01f02e
Key [ 9]:346cfc553c6cbc9713edb55f4dcbc96c
Key [10]:bddf027cb059d58f0509f8963e9bdec6
round  6:92b33f11eadcacc5a43dd05f13d334dd
Key [11]:8eb9e040c36c4c0b4a7fd3dd354d53c4
Key [12]:c6ffecdd5e135b20879b9dfa4b34bf51
round  7:fb0541aa5e5df1a61c51aef606eb5a41
Key [13]:bf12f5a6ba08dfc4fda4bdfc68c997d9
Key [14]:37c4656b9215f3c959ea688fb64ad327
round  8:f0bbd2b94ae374346730581fc77a9c98
Key [15]:e87bb0d86bf421ea4f779a8eee3a866c
Key [16]:faa471e934fd415ae4c0113ec7f0a5ad
Key [17]:95204a80b8400e49db7cf6fd2fd40d9a
Ka      :b0376d0a9b338c2e133c32b69cb816b3
----------------------------------------
rand    :13ecad08ad63c37f8a54dc56e82f4dc1
address :9846c5ead4d9
round  1:9846c5ead4d99846c5ead4d99846c5ea
Key [ 1]:13ecad08ad63c37f8a54dc56e82f4dc7
Key [ 2]:ad04f127bed50b5e671d6510d392eaed
round  2:97374e18cdd0a6f7a5aa49d1ac875c84
Key [ 3]:57ad159e5774fa222f2f3039b9cd5101
Key [ 4]:9a1e9e1068fede02ef90496e25fd8e79
round  3:9dd3260373edd9d5f4e774826b88fd2d
added ->:0519ebe9a7c6719331d1485bf3cec2c7
Key [ 5]:378dce167db62920b0b392f7cfca316e
Key [ 6]:db4277795c87286faee6c9e9a6b71a93
```

```
round  4:40ec6563450299ac4e120d88672504d6
Key [ 7]:ec01aa2f5a8a793b36c1bb858d254380
Key [ 8]:2921a66cfa5bf74ac535424564830e98
round  5:57287bbb041bd6a56c2bd931ed410cd4
Key [ 9]:07018e45aab61b3c3726ee3d57dbd5f6
Key [10]:627381f0fa4c02b0c7d3e7dfbffc3333
round  6:66affa66a8dcd36e36bf6c3f1c6a276e
Key [11]:33b57c925bd5551999f716e138efbe79
Key [12]:a6dc7f9aa95bcc9243aebd12608f657a
round  7:450e65184fd8c72c578d5cdecb286743
Key [13]:a6a6db00fd8c72a28ea57ea542f6e102
Key [14]:dcf3377daeb2e24e61f0ad6620951c1f
round  8:e5eb180b519a4e673f21b7c4f4573f3d
Key [15]:621240b9506b462a7fa250da41844626
Key [16]:ae297810f01f43dc35756cd119ee73d6
Key [17]:b959835ec2501ad3894f8b8f1f4257f9
Ka      :5b61e83ad04d23e9d1c698851fa30447
========================================
```

9.3 THREE TESTS OF E22

(for K_master and overlay generation)

```
rand    :001de169248850245a5f7cc7f0d6d633
PIN     :d5a51083a04a1971f18649ea8b79311a
round  1:001de169248850245a5f7cc7f0d6d623
Key [ 1]:d5a51083a04a1971f18649ea8b79311a
Key [ 2]:7317cdbff57f9b99f9810a2525b17cc7
round  2:5f05c143347b59acae3cb002db23830f
Key [ 3]:f08bd258adf1d4ae4a54d8ccb26220b2
Key [ 4]:91046cbb4ccc43db18d6dd36ca7313eb
round  3:c8f3e3300541a25b6ac5a80c3105f3c4
added ->:c810c45921c9f27f302424cbc1dbc9e7
Key [ 5]:67fb2336f4d9f069da58d11c82f6bd95
Key [ 6]:4fed702c75bd72c0d3d8f38707134c50
round  4:bd5e0c3a97fa55b91a3bbbf306ebb978
Key [ 7]:41c947f80cdc0464c50aa89070af314c
Key [ 8]:680eecfa8daf41c7109c9a5cb1f26d75
round  5:21c1a762c3cc33e75ce8976a73983087
Key [ 9]:6e33fbd94d00ff8f72e8a7a0d2cebc4c
Key [10]:f4d726054c6b948add99fabb5733ddc3
round  6:56d0df484345582f6b574a449ba155eb
Key [11]:4eda2425546a24cac790f49ef2453b53
Key [12]:cf2213624ed1510408a5a3e00b7333df
round  7:120cf9963fe9ff22993f7fdf9600d9b8
Key [13]:d04b1a25b0b8fec946d5ecfa626d04c9
Key [14]:01e5611b0f0e140bdb64585fd3ae5269
round  8:a6337400ad8cb47fefb91332f5cb2713
Key [15]:f15b2dc433f534f61bf718770a3698b1
Key [16]:f990d0273d8ea2b9e0b45917a781c720
Key [17]:f41b3cc13d4301297bb6bdfcb3e5a1dd
Ka      :539e4f2732e5ae2de1e0401f0813bd0d
```

Sample Data

```
----------------------------------------
rand    :67ed56bfcf99825f0c6b349369da30ab
PIN     :7885b515e84b1f082cc499976f1725ce
round  1:67ed56bfcf99825f0c6b349369da30bb
Key [ 1]:7885b515e84b1f082cc499976f1725ce
Key [ 2]:72445901fdaf506beb036f4412512248
round  2:6b160b66a1f6c26c1f3432f463ef5aa1
Key [ 3]:59f0e4982e97633e5e7fd133af8f2c5b
Key [ 4]:b4946ec77a41bf7c729d191e33d458ab
round  3:3f22046c964c3e5ca2a26ec9a76a9f67
added ->:580f5ad359e5c003ae0da25ace44cfdc
Key [ 5]:eb0b839f97bdf534183210678520bbef
Key [ 6]:cff0bc4a94e5c8b2a2d24d9f59031e19
round  4:87aa61fc0ff88e744c195249b9a33632
Key [ 7]:592430f14d8f93db95dd691af045776d
Key [ 8]:3b55b404222bf445a6a2ef5865247695
round  5:83dcf592a854226c4dcd94e1ecf1bc75
Key [ 9]:a9714b86319ef343a28b87456416bd52
Key [10]:e6598b24390b3a0bf2982747993b0d78
round  6:dee0d13a52e96bcf7c72045a21609fc6
Key [11]:62051d8c51973073bff959b032c6e1e2
Key [12]:29e94f4ab73296c453c833e217a1a85b
round  7:08488005761e6c7c4dbb203ae453fe3a
Key [13]:0e255970b3e2fc235f59fc5acb10e8ce
Key [14]:d0dfbb3361fee6d4ffe45babf1cd7abf
round  8:0d81e89bddde7a7065316c47574feb8f
Key [15]:c12eee4eb38b7a171f0f736003774b40
Key [16]:8f962523f1c0abd9a087a0dfb11643d3
Key [17]:24be1c66cf8b022f12f1fb4c60c93fd1
Ka      :04435771e03a9daceb8bb1a493ee9bd8
----------------------------------------
rand    :40a94509238664f244ff8e3d13b119d3
PIN     :1ce44839badde30396d03c4c36f23006
round  1:40a94509238664f244ff8e3d13b119c3
Key [ 1]:1ce44839badde30396d03c4c36f23006
Key [ 2]:6dd97a8f91d628be4b18157af1a9dcba
round  2:0eac5288057d9947a24eabc1744c4582
Key [ 3]:fef9583d5f55fd4107ad832a725db744
Key [ 4]:fc3893507016d7c1db2bd034a230a069
round  3:60b424f1082b0cc3bd61be7b4c0155f0
added ->:205d69f82bb17031f9604c465fb26e33
Key [ 5]:0834d04f3e7e1f7f85f0c1db685ab118
Key [ 6]:1852397f9a3723169058e9b62bb3682b
round  4:2c6b65a49d66af6566675afdd6fa7d7d
Key [ 7]:6c10da21d762ae4ac1ba22a96d9007b4
Key [ 8]:9aa23658b90470a78d686344b8a9b0e7
round  5:a2c537899665113a42f1ac24773bdc31
Key [ 9]:137dee3bf879fe7bd02fe6d888e84f16
Key [10]:466e315a1863f47d0f93bc6827cf3450
round  6:e26982980d79b21ed3e20f8c3e71ba96
Key [11]:0b33cf831465bb5c979e6224d7f79f7c
Key [12]:92770660268ede827810d707a0977d73
```

Sample Data

```
round  7:e7b063c4e2e3110b89b7e1631c762dd5
Key [13]:7be30ae4961cf24ca17625a77bb7a9f8
Key [14]:be65574a33ae30e6e82dbd2826d3cc1a
round  8:7a963e37b2c2e76b489cfe40a2cf00e5
Key [15]:ed0ba7dd30d60a5e69225f0a33011e5b
Key [16]:765c990f4445e52b39e6ed6105ad1c4f
Key [17]:52627bf9f35d94f30d5b07ef15901adc
Ka      :9cde4b60f9b5861ed9df80858bac6f7f


========================================
```

9.4 TESTS OF E22 WITH PIN AUGMENTING

for PIN lengths 1,...,16 bytes

```
rand    :24b101fd56117d42c0545a4247357048
PIN length =16 octets
PIN     :fd397c7f5c1f937cdf82d8816cc377e2
round  1:24b101fd56117d42c0545a4247357058
Key [ 1]:fd397c7f5c1f937cdf82d8816cc377e2
Key [ 2]:0f7aac9c9b53f308d9fdbf2c78e3c30e
round  2:838edfe1226266953ccba8379d873107
Key [ 3]:0b8ac18d4bb44fad2efa115e43945abc
Key [ 4]:887b16b062a83bfa469772c25b456312
round  3:8cd0c9283120aba89a7f9d635dd4fe3f
added ->:a881cad5673128ea5ad3f7211a096e67
Key [ 5]:2248cbe6d299e9d3e8fd35a91178f65b
Key [ 6]:b92af6237385bd31f8fb57fb1bdd824e
round  4:2648d9c618a622b10ef80c4dbaf68b99
Key [ 7]:2bf5ffe84a37878ede2d4c30be60203b
Key [ 8]:c9cb6cec60cb8a8f29b99fcf3e71e40f
round  5:b5a7d9e96f68b14ccebf361de3914d0f
Key [ 9]:5c2f8a702e4a45575b103b0cce8a91c6
Key [10]:d453db0c9f9ddbd11e355d9a34d9b11b
round  6:632a091e7eefe1336857ddafd1ff3265
Key [11]:32805db7e59c5ed4acabf38d27e3fece
Key [12]:fde3a8eedfa3a12be09c1a8a00890fd7
round  7:048531e9fd3efa95910540150f8b137b
Key [13]:def07eb23f3a378f059039a2124bc4c2
Key [14]:2608c58f23d84a09b9ce95e5caac1ab4
round  8:461814ec7439d412d0732f0a6f799a6a
Key [15]:0a7ed16481a623e56ee1442ffa74f334
Key [16]:12add59aca0d19532f1516979954e369
Key [17]:dd43d02d39ffd6a386a4b98b4ac6eb23
Ka      :a5f2adf328e4e6a2b42f19c8b74ba884
----------------------------------------
rand    :321964061ac49a436f9fb9824ac63f8b
PIN length =15 octets
PIN     :ad955d58b6b8857820ac1262d617a6
address :0314c0642543
round  1:321964061ac49a436f9fb9824ac63f9b
```

Sample Data

```
Key [ 1]:ad955d58b6b8857820ac1262d617a603
Key [ 2]:f281736f68e3d30b2ac7c67f125dc416
round  2:7c4a4ece1398681f4bafd309328b7770
Key [ 3]:43c157f4c8b360387c32ab330f9c9aa8
Key [ 4]:3a3049945a298f6d076c19219c47c3cb
round  3:9672b00738bdfaf9bd92a855bc6f3afb
added ->:a48b1401228194bad23161d7f6357960
Key [ 5]:c8e2eaa6d73b7de18f3228ab2173bc69
Key [ 6]:8623f44488222e66a293677cf30bf2bb
round  4:9b30247aad3bf133712d034b46d21c68
Key [ 7]:f3e500902fba31db9bae50ef30e484a4
Key [ 8]:49d4b1137c18f4752dd9955a5a8d2f43
round  5:4492c25fda08083a768b4b5588966b23
Key [ 9]:9d59c451989e74785cc097eda7e42ab8
Key [10]:251de25f3917dcd99c18646107a641fb
round  6:21ae346635714d2623041f269978c0ee
Key [11]:80b8f78cb1a49ec0c3e32a238e60fddf
Key [12]:beb84f4d20a501e4a24ecfbde481902b
round  7:9b56a3d0f8932f20c6a77a229514fb00
Key [13]:852571b44f35fd9d9336d3c1d2506656
Key [14]:d0a0d510fb06ba76e69b8ee3ebc1b725
round  8:6cd8492b2fd31a86978bcdf644eb08a8
Key [15]:c7ffd523f32a874ed4a93430a25976de
Key [16]:16cdcb25e62964876d951fdcc07030d3
Key [17]:def32c0e12596f9582e5e3c52b303f52
Ka      :c0ec1a5694e2b48d54297911e6c98b8f
----------------------------------------
rand    :d4ae20c80094547d7051931b5cc2a8d6
PIN length =14 octets
PIN     :e1232e2c5f3b833b3309088a87b6
address :fabecc58e609
round  1:d4ae20c80094547d7051931b5cc2a8c6
Key [ 1]:e1232e2c5f3b833b3309088a87b6fabe
Key [ 2]:5f0812b47cd3e9a30d7707050fffa1f2
round  2:1f45f16be89794bef33e4547c9c0916a
Key [ 3]:77b6819447632 44ffa3cd71b248b79b5
Key [ 4]:e2814e90e04f485958ce58c9133e2be6
round  3:b10d2f4ac941035263cee3552d774d2f
added ->:65bb4f82c9d5572f131f764e7139f5e9
Key [ 5]:520acad20801dc639a2c6d66d9b79576
Key [ 6]:c72255cdb61d42be72bd45390dd25ba5
round  4:ead4dc34207b6ea721c62166e155aaad
Key [ 7]:ebf04c02075bf459ec9c3ec06627d347
Key [ 8]:a1363dd2812ee800a4491c0c74074493
round  5:f507944f3018e20586d81d7f326aae9d
Key [ 9]:b0b6ba79493dc833d7f425be7b8dadb6
Key [10]:08cd23e536b9b9b53e85eb004cba3111
round  6:fff450f4302a2b3571e8405e148346da
Key [11]:fec22374c6937dcd26171f4d2edfada3
Key [12]:0f1a8ef5979c69ff44f620c2e007b6e4
round  7:de558779589897f3402a90ee78c3f921
Key [13]:901fb66f0779d6aad0c0fba1fe812cb5
```

Sample Data

```
Key [14]:a0cab3cd15cd23603adc8d4474efb239
round  8:b2df0aa0c9f07fbbaa02f510a29cf540
Key [15]:18edc3f4296dd6f1dea13f7c143117a1
Key [16]:8d3d52d700a379d72ded81687f7546c7
Key [17]:5927badfe602f29345f840bb53e1dea6
Ka      :d7b39be13e3692c65b4a9e17a9c55e17
----------------------------------------
rand    :272b73a2e40db52a6a61c6520549794a
PIN length =13 octets
PIN     :549f2694f353f5145772d8ae1e
address :20487681eb9f
round  1:272b73a2e40db52a6a61c6520549795a
Key [ 1]:549f2694f353f5145772d8ae1e204876
Key [ 2]:42c855593d66b0c458fd28b95b6a5fbf
round  2:d7276dc8073f7677c31f855bde9501e2
Key [ 3]:75d0a69ae49a2da92e457d767879df52
Key [ 4]:b3aa7e7492971afaa0fb2b64827110df
round  3:71aae503831133d19bc452da4d0e409b
added ->:56d558a1671ee8fbf12518884857b9c1
Key [ 5]:9c8cf1604a98e9a503c342e272de5cf6
Key [ 6]:d35bc2df6b85540a27642106471057d9
round  4:f41a709c89ea80481aa3d2b9b2a9f8ca
Key [ 7]:b454dda74aeb4eff227ba48a58077599
Key [ 8]:bcba6aec050116aa9b7c6a9b7314d796
round  5:20fdda20f4a26b1bd38eb7f355a7be87
Key [ 9]:d41f8a9de0a716eb7167a1b6e321c528
Key [10]:5353449982247782d168ab43f17bc4d8
round  6:a70e316997eeed49a5a9ef9ba5e913b5
Key [11]:32cbc9cf1a81e36a45153972347ce4ac
Key [12]:5747619006cf4ef834c749f2c4b9feb6
round  7:e66f2317a825f589f76b47b6aa6e73fb
Key [13]:f9b68beba0a09d2a570a7dc88cc3c3c2
Key [14]:55718f9a4f0b1f9484e8c6b186a41a4b
round  8:5f68f940440a9798e074776019804ada
Key [15]:4ecc29be1b4d78433f6aa30db974a7fb
Key [16]:8470a066ffb00cda7b08059599f919f5
Key [17]:f39a36d74e960a051e1ca98b777848f4
Ka      :9ac64309a37c25c3b4a584fc002a1618
----------------------------------------
rand    :7edb65f01a2f45a2bc9b24fb3390667e
PIN length =12 octets
PIN     :2e5a42797958557b23447ca8
address :04f0d2737f02
round  1:7edb65f01a2f45a2bc9b24fb3390666e
Key [ 1]:2e5a42797958557b23447ca804f0d273
Key [ 2]:18a97c856561eb23e71af8e9e1be4799
round  2:3436e12db8ffdc1265cb5a86da2fed0b
Key [ 3]:7c0908dcbc73201e17c4f7aa1ab8aec8
Key [ 4]:7cb58833602fbe4194c7cc797ce8c454
round  3:caed6af4226f67e4ad1914620803ef2a
added ->:b4c8cf04389eac4611b438993b935544
Key [ 5]:f4dce7d607b5234562d0ebb2267b08b8
```

Sample Data

```
Key [ 6]:560b75c5545751fd8fa99fa4346e654b
round  4:ee67c87d6f74bb75db98f68bff0192c1
Key [ 7]:32f10cefd8d3e6424c6f91f1437808af
Key [ 8]:a934a46045be30fb3be3a5f3f7b18837
round  5:792398dcbeb8d10bdb07ae3c819e943c
Key [ 9]:a0f12e97c677a0e8ac415cd2c8a7ca88
Key [10]:e27014c908785f5ca03e8c6a1da3bf13
round  6:e778b6e0c3e8e7edf90861c7916d97a8
Key [11]:1b4a4303bcc0b2e0f41c72d47654bd9f
Key [12]:4b1302a50046026d6c9054fc8387965a
round  7:1fafddc7efa5f04c1dec1869d3f2d9bb
Key [13]:58c334bb543d49eca562cdbe0280e0fc
Key [14]:bdb60d383c692d06476b76646c8dec48
round  8:3d7c326d074bd6aa222ea050f04a3c7f
Key [15]:78c0162506be0b5953e8403c01028f93
Key [16]:24d7dbbe834dbd7b67f57fcf0d39d60f
Key [17]:2e74f1f3331c0f6585e87b2f715e187e
Ka      :d3af4c81e3f482f062999dee7882a73b
----------------------------------------
rand    :26a92358294dce97b1d79ec32a67e81a
PIN length =11 octets
PIN     :05fbad03f52fa9324f7732
address :b9ac071f9d70
round  1:26a92358294dce97b1d79ec32a67e80a
Key [ 1]:05fbad03f52fa9324f7732b9ac071f9d
Key [ 2]:2504c9691c04a18480c8802e922098c0
round  2:0be20e3d76888e57b6bf77f97a8714fb
Key [ 3]:576b2791d1212bea8408212f2d43e77e
Key [ 4]:90ae36dcce8724adb618f912d1b27297
round  3:1969667060764453257d906b7e58bd5b
added ->:3f12892849c312c494542ea854bfa551
Key [ 5]:bc492c42c9e87f56ec31af5474e9226e
Key [ 6]:c135d1dbed32d9519acfb4169f3e1a10
round  4:ac404205118fe771e54aa6f392da1153
Key [ 7]:83ccbdbbaf17889b7d18254dc9252fa1
Key [ 8]:80b90a1767d3f2848080802764e21711
round  5:41795e89ae9a0cf776ffece76f47fd7a
Key [ 9]:cc24e4a86e8eed129118fd3d5223a1dc
Key [10]:7b1e9c0eb9dab083574be7b7015a62c9
round  6:29ca9e2f87ca00370ef1633505bfba4b
Key [11]:888e6d88cf4beb965cf7d4f32b696baa
Key [12]:6d642f3e5510b0b043a44daa2cf5eec0
round  7:81fc891c3c6fd99acc00028a387e2366
Key [13]:e224f85da2ab63a23e2a3a036e421358
Key [14]:c8dc22aaa739e2cb85d6a0c08226c7d0
round  8:e30b537e7a000e3d2424a9c0f04c4042
Key [15]:a969aa818c6b324bae391bedcdd9d335
Key [16]:6974b6f2f07e4c55f2cc0435c45bebd1
Key [17]:134b925ebd98e6b93c14aee582062fcb
Ka      :be87b44d079d45a08a71d15208c5cb50
----------------------------------------
rand    :0edef05327eab5262430f21fc91ce682
```

Sample Data

```
PIN length =10 octets
PIN     :8210e47390f3f48c32b3
address :7a3cdfe377d1
round  1:0edef05327eab5262430f21fc91ce692
Key [ 1]:8210e47390f3f48c32b37a3cdfe377d1
Key [ 2]:c6be4c3e425e749b620a94c779e33a7e
round  2:07ca3c7a7a6bcbc31d79a856d9cffc0e
Key [ 3]:2587cec2a4b8e4f996a9ed664350d5dd
Key [ 4]:70e4bf72834d9d3dbb7eb2c239216dc0
round  3:792ad2ac4e4559d1463714d2f161b6f4
added ->:7708c2ff692f0ef7626706cd387d9c66
Key [ 5]:6696e1e7f8ac037e1fff3598f0c164e2
Key [ 6]:23dbfe4d0b561bea08fbcef25e49b648
round  4:7d8c71a9d7fbdcbd851bdf074550b100
Key [ 7]:b03648acd021550edee904431a02f00c
Key [ 8]:cb169220b7398e8f077730aa4bf06baa
round  5:b6fcaa45064ffd557e4b7b30cfbb83e0
Key [ 9]:af602c2ba16a454649951274c2be6527
Key [10]:5d60b0a7a09d524143eca13ad680bc9c
round  6:b3416d391a0c26c558843debd0601e9e
Key [11]:9a2f39bfe558d9f562c5f09a5c3c0263
Key [12]:72cae8eebd7fabd9b1848333c2aab439
round  7:abe4b498d9c36ea97b8fd27d7f813913
Key [13]:15f27ea11e83a51645d487b81371d7dc
Key [14]:36083c8666447e03d33846edf444eb12
round  8:8032104338a945ba044d102eabda3b22
Key [15]:0a3a8977dd48f3b6c1668578befadd02
Key [16]:f06b6675d78ca0ee5b1761bdcdab516d
Key [17]:cbc8a7952d33aa0496f7ea2d05390b23
Ka      :bf0706d76ec3b11cce724b311bf71ff5
----------------------------------------
rand    :86290e2892f278ff6c3fb917b020576a
PIN length = 9 octets
PIN     :3dcdffcfd086802107
address :791a6a2c5cc3
round  1:86290e2892f278ff6c3fb917b0205765
Key [ 1]:3dcdffcfd086802107791a6a2c5cc33d
Key [ 2]:b4962f40d7bb19429007062a3c469521
round  2:1ec59ffd3065f19991872a7863b0ef02
Key [ 3]:eb9ede6787dd196b7e340185562bf28c
Key [ 4]:2964e58aacf7287d1717a35b100ae23b
round  3:f817406f1423fc2fe33e25152679eaaf
added ->:7e404e47861574d08f7dde02969941ca
Key [ 5]:6abf9a314508fd61e486fa4e376c3f93
Key [ 6]:6da148b7ee2632114521842cbb274376
round  4:e9c2a8fac22b8c7cf0c619e2b3f890ed
Key [ 7]:df889cc34fda86f01096d52d116e620d
Key [ 8]:5eb04b147dc39d1974058761ae7b73fc
round  5:444a8aac0efee1c02f8d38f8274b7b28
Key [ 9]:8426cc59eee391b2bd50cf8f1efef8b3
Key [10]:8b5d220a6300ade418da791dd8151941
round  6:9185f983db150b1bccab1e5c12eb63a1
```

Sample Data

```
Key [11]:82ba4ddef833f6a4d18b07aa011f2798
Key [12]:ce63d98794682054e73d0359dad35ec4
round  7:5eded2668f5916dfd036c09e87902886
Key [13]:da794357652e80c70ad8b0715dbe33d6
Key [14]:732ef2c0c3220b31f3820c375e27bb29
round  8:88a5291b4acbba009a85b7dd6a834b3b
Key [15]:3ce75a61d4b465b70c95d7ccd5799633
Key [16]:5df9bd2c3a17a840cdaafb76c171db7c
Key [17]:3f8364b089733d902bccb0cd3386846f
Ka      :cdb0cc68f6f6fbd70b46652de3ef3ffb
----------------------------------------
rand    :3ab52a65bb3b24a08eb6cd284b4b9d4b
PIN length = 8 octets
PIN     :d0fb9b6838d464d8
address :25a868db91ab
round  1:3ab52a65bb3b24a08eb6cd284b4b9d45
Key [ 1]:d0fb9b6838d464d825a868db91abd0fb
Key [ 2]:2573f47b49dad6330a7a9155b7ae8ba1
round  2:ad2ffdff408fcfab44941016a9199251
Key [ 3]:d2c5b8fb80cba13712905a589adaee71
Key [ 4]:5a3381511b338719fae242758dea0997
round  3:2ddc17e570d7931a2b1d13f6ace928a5
added ->:17914180cb12b7baa5d3e0dee734c5e0
Key [ 5]:e0a4d8ac27fbe2783b7bcb3a36a6224d
Key [ 6]:949324c6864deac3eca8e324853e11c3
round  4:62c1db5cf31590d331ec40ad692e8df5
Key [ 7]:6e67148088a01c2d4491957cc9ddc4aa
Key [ 8]:557431deab7087bb4c03fa27228f60c6
round  5:9c8933bc361f4bde4d1bda2b5f8bb235
Key [ 9]:a2551aca53329e70ade3fd2bb7664697
Key [10]:05d0ad35de68a364b54b56e2138738fe
round  6:9156db34136aa06655bf28a05be0596a
Key [11]:1616a6b13ce2f2895c722e8495181520
Key [12]:b12e78a1114847b01f6ed2f5a1429a23
round  7:84dcc292ed836c1c2d523f2a899a2ad5
Key [13]:316e144364686381944e95afd8a026bb
Key [14]:1ab551b88d39d97ea7a9fe136dbfe2e1
round  8:87bdcac878d777877f4eccf042cfee5e
Key [15]:70e21ab08c23c7544524b64492b25cc9
Key [16]:35f730f2ae2b950a49a1bf5c8b9f8866
Key [17]:2f16924c22db8b74e2eadf1ba4ebd37c
Ka      :983218718ca9aa97892e312d86dd9516
----------------------------------------
rand    :a6dc447ff08d4b366ff96e6cf207e179
PIN length = 7 octets
PIN     :9c57e10b4766cc
address :54ebd9328cb6
round  1:a6dc447ff08d4b366ff96e6cf207e174
Key [ 1]:9c57e10b4766cc54ebd9328cb69c57e1
Key [ 2]:00a609f4d61db26993c8177e3ee2bba8
round  2:1ed26b96a306d7014f4e5c9ee523b73d
Key [ 3]:646d7b5f9aaa528384bda3953b542764
```

Sample Data

```
Key [ 4]:a051a42212c0e9ad5c2c248259aca14e
round  3:a53f526db18e3d7d53edbfc9711041ed
added ->:031b9612411b884b3ce62da583172299
Key [ 5]:d1bd5e64930e7f838d8a33994462d8b2
Key [ 6]:5dc7e2291e32435665ebd6956bec3414
round  4:9438be308ec83f35c560e2796f4e0559
Key [ 7]:10552f45af63b0f15e2919ab37f64fe7
Key [ 8]:c44d5717c114a58b09207392ebe341f8
round  5:b79a7b14386066d339f799c40479cb3d
Key [ 9]:6886e47b782325568eaf59715a75d8ff
Key [10]:8e1e335e659cd36b132689f78c147bda
round  6:ef232462228aa166438d10c34e17424b
Key [11]:8843efeedd5c2b7c3304d647f932f4d1
Key [12]:13785aaedd0adf67abb4f01872392785
round  7:02d133fe40d15f1073673b36bba35abd
Key [13]:837d7ca2722419e6be3fae35900c3958
Key [14]:93f8442973e7fccf2e7232d1d057c73a
round  8:275506a3d08c84e94cc58ed60054505e
Key [15]:8a7a9edffa3c52918bc6a45f57d91f5d
Key [16]:f214a95d777f763c56109882c4b52c84
Key [17]:10e2ee92c5ea1ddc5eb010e55510c403
Ka      :9cd6650ead86323e87cafb1ff516d1e0
----------------------------------------
rand    :3348470a7ea6cc6eb81b40472133262c
PIN length = 6 octets
PIN     :fcad169d7295
address :430d572f8842
round  1:3348470a7ea6cc6eb81b404721332620
Key [ 1]:fcad169d7295430d572f8842fcad169d
Key [ 2]:b3479d4d4fd178c43e7bc5b0c7d8983c
round  2:af976da9225066d563e10ab955e6fc32
Key [ 3]:7112462b37d82dd81a2a35d9eb43cb7c
Key [ 4]:c5a7030f8497945ac7b84600d1d161fb
round  3:d08f826ebd55a0bd7591c19a89ed9bde
added ->:e3d7c964c3fb6cd3cdac01dda820c1fe
Key [ 5]:84b0c6ef4a63e4dff19b1f546d683df5
Key [ 6]:f4023edfc95d1e79ed4bb4de9b174f5d
round  4:6cd952785630dfc7cf81eea625e42c5c
Key [ 7]:ea38dd9a093ac9355918632c90c79993
Key [ 8]:dbba01e278ddc76380727f5d7135a7de
round  5:93573b2971515495978264b88f330f7f
Key [ 9]:d4dc3a31be34e412210fafa6eca00776
Key [10]:39d1e190ee92b0ff16d92a8be58d2fa0
round  6:b3f01d5e7fe1ce6da7b46d8c389baf47
Key [11]:1eb081328d4bcf94c9117b12c5cf22ac
Key [12]:7e047c2c552f9f1414d946775fabfe30
round  7:0b833bff6106d5bae033b4ce5af5a924
Key [13]:e78e685d9b2a7e29e7f2a19d1bc38ebd
Key [14]:1b582272a3121718c4096d2d8602f215
round  8:23de0bbdc70850a7803f4f10c63b2c0f
Key [15]:8569e860530d9c3d48a0870dac33f676
Key [16]:6966b528fdd1dc222527052c8f6cf5a6
```

Sample Data

```
Key [17]:a34244c757154c53171c663b0b56d5c2
Ka      :98f1543ab4d87bd5ef5296fb5e3d3a21
----------------------------------------
rand    :0f5bb150b4371ae4e5785293d22b7b0c
PIN length = 5 octets
PIN     :b10d068bca
address :b44775199f29
round  1:0f5bb150b4371ae4e5785293d22b7b07
Key [ 1]:b10d068bcab44775199f29b10d068bca
Key [ 2]:aec70d1048f1bbd2c18040318a8402ad
round  2:342d2b79d7fb7cd110379742b9842c79
Key [ 3]:6d8d5cf338f29ef4420639ef488e4fa9
Key [ 4]:a1584117541b759ba6d9f7eb2bedcbba
round  3:9407e8e3e810603921bf81cfda62770a
added ->:9b6299b35c477addc437d35c088df20d
Key [ 5]:09a20676666aeed6f22176274eb433f4
Key [ 6]:840472c001add5811a054be5f5c74754
round  4:9a3ba953225a7862c0a842ed3d0b2679
Key [ 7]:fad9e45c8bf70a972fcd9bff0e8751f5
Key [ 8]:e8f30ff666dfd212263416496ff3b2c2
round  5:2c573b6480852e875df34b28a5c44509
Key [ 9]:964cdba0cf8d593f2fc40f96daf8267a
Key [10]:bcd65c11b13e1a70bcd4aafba8864fe3
round  6:21b0cc49e880c5811d24dee0194e6e9e
Key [11]:468c8548ea9653c1a10df6288dd03c1d
Key [12]:5d252d17af4b09d3f4b5f7b5677b8211
round  7:e6d6bdcd63e1d37d9883543ba86392fd
Key [13]:e814bf307c767428c67793dda2df95c7
Key [14]:4812b979fdc20f0ff0996f61673a42cc
round  8:e3dde7ce6bd7d8a34599aa04d6a760ab
Key [15]:5b1e2033d1cd549fc4b028146eb5b3b7
Key [16]:0f284c14fb8fe706a5343e3aa35af7b1
Key [17]:b1f7a4b7456d6b577fded6dc7a672e37
Ka      :c55070b72bc982adb972ed05d1a74ddb
----------------------------------------
rand    :148662a4baa73cfadb55489159e476e1
PIN length = 4 octets
PIN     :fb20f177
address :a683bd0b1896
round  1:148662a4baa73cfadb55489159e476eb
Key [ 1]:fb20f177a683bd0b1896fb20f177a683
Key [ 2]:47266cefbfa468ca7916b458155dc825
round  2:3a942eb6271c3f4e433838a5d3fcbd27
Key [ 3]:688853a6d6575eb2f6a2724b0fbc133b
Key [ 4]:7810df048019634083a2d9219d0b5fe0
round  3:9c835b98a063701c0887943596780769
added ->:8809bd3c1a0aace6d3dcdca4cf5c7d82
Key [ 5]:c78f6dcf56da1bbd413828b33f5865b3
Key [ 6]:eb3f3d407d160df3d293a76d1a513c4a
round  4:7e68c4bafa020a4a59b5a1968105bab5
Key [ 7]:d330e038d6b19d5c9bb0d7285a360064
Key [ 8]:9bd3ee50347c00753d165faced702d9c
```

Sample Data

```
round  5:227bad0cf0838bdb15b3b3872c24f592
Key [ 9]:9543ad0fb3fe74f83e0e2281c6d4f5f0
Key [10]:746cd0383c17e0e80e6d095a87fd0290
round  6:e026e98c71121a0cb739ef6f59e14d26
Key [11]:fa28bea4b1c417536608f11f406ea1dd
Key [12]:3aee0f4d21699df9cb8caf5354a780ff
round  7:cd6a6d8137d55140046f8991da1fa40a
Key [13]:372b71bc6d1aa6e785358044fbcf05f4
Key [14]:00a01501224c0405de00aa2ce7b6ab04
round  8:52cd7257fe8d0c782c259bcb6c9f5942
Key [15]:c7015c5c1d7c030e00897f104a006d4a
Key [16]:260a9577790c62e074e71e19fd2894df
Key [17]:c041b7a231493acd15ddcdaee94b9f52
Ka      :7ec864df2f1637c7e81f2319ae8f4671
----------------------------------------
rand    :193a1b84376c88882c8d3b4ee93ba8d5
PIN length = 3 octets
PIN     :a123b9
address :4459a44610f6
round  1:193a1b84376c88882c8d3b4ee93ba8dc
Key [ 1]:a123b94459a44610f6a123b94459a446
Key [ 2]:5f64d384c8e990c1d25080eb244dde9b
round  2:3badbd58f100831d781ddd3ccedefd3f
Key [ 3]:5abc00eff8991575c00807c48f6dbea5
Key [ 4]:127521158ad6798fb6479d1d2268abe6
round  3:0b53075a49c6bf2df2421c655fdedf68
added ->:128d22de7e3247a5decf572bb61987b4
Key [ 5]:f2a1f620448b8e56665608df2ab3952f
Key [ 6]:7c84c0af02aad91dc39209c4edd220b1
round  4:793f4484fb592e7a78756fd4662f990d
Key [ 7]:f6445b647317e7e493bb92bf6655342f
Key [ 8]:3cae503567c63d3595eb140ce60a84c0
round  5:9e46a8df925916a342f299a8306220a0
Key [ 9]:734ed5a806e072bbecb4254993871679
Key [10]:cda69ccb4b07f65e3c8547c11c0647b8
round  6:6bf9cd82c9e1be13fc58eae0b936c75a
Key [11]:c48e531d3175c2bd26fa25cc8990e394
Key [12]:6d93d349a6c6e9ff5b26149565b13d15
round  7:e96a9871471240f198811d4b8311e9a6
Key [13]:5c4951e85875d663526092cd4cbdb667
Key [14]:f19f7758f5cde86c3791efaf563b3fd0
round  8:e94ca67d3721d5fb08ec069191801a46
Key [15]:bf0c17f3299b37d984ac938b769dd394
Key [16]:7edf4ad772a6b9048588f97be25bde1c
Key [17]:6ee7ba6afefc5b561abbd8d6829e8150
Ka      :ac0daabf17732f632e34ef193658bf5d
----------------------------------------
rand    :1453db4d057654e8eb62d7d62ec3608c
PIN length = 2 octets
PIN     :3eaf
address :411fbbb51d1e
round  1:1453db4d057654e8eb62d7d62ec36084
```

Sample Data

```
Key [ 1]:3eaf411fbbb51d1e3eaf411fbbb51d1e
Key [ 2]:c3a1a997509f00fb4241aba607109c64
round  2:0b78276c1ebc65707d38c9c5fa1372bd
Key [ 3]:3c729833ae1ce7f84861e4dbad6305cc
Key [ 4]:c83a43c3a66595cb8136560ed29be4ff
round  3:23f3f0f6441563d4c202cee0e5cb2335
added ->:3746cbbb418bb73c2964a536cb8e83b1
Key [ 5]:18b26300b86b70acdd1c8f5cbc7c5da8
Key [ 6]:04efc75309b98cd8f1cef5513c18e41e
round  4:c61afa90d3c14bdf588320e857afdc00
Key [ 7]:517c789cecadc455751af73198749fb8
Key [ 8]:fd9711f913b5c844900fa79dd765d0e2
round  5:a8a0e02ceb556af8bfa321789801183a
Key [ 9]:bb5cf30e7d3ceb930651b1d16ee92750
Key [10]:3d97c7862ecab42720e984972f8efd28
round  6:0b58e922438d224db34b68fca9a5ea12
Key [11]:4ce730344f6b09e449dcdb64cd466666
Key [12]:38828c3a56f922186adcd9b713cdcc31
round  7:b90664c4ac29a8b4bb26debec9ffc5f2
Key [13]:d30fd865ea3e9edcff86a33a2c319649
Key [14]:1fdb63e54413acd968195ab6fa424e83
round  8:6934de3067817cefd811abc5736c163b
Key [15]:a16b7c655bbaa262c807cba8ae166971
Key [16]:7903dd68630105266049e23ca607cda7
Key [17]:888446f2d95e6c2d2803e6f4e815ddc9
Ka      :1674f9dc2063cc2b83d3ef8ba692ebef
----------------------------------------
rand    :1313f7115a9db842fcedc4b10088b48d
PIN length = 1 octets
PIN     :6d
address :008aa9be62d5
round  1:1313f7115a9db842fcedc4b10088b48a
Key [ 1]:6d008aa9be62d56d008aa9be62d56d00
Key [ 2]:46ebfeafb6657b0a1984a8dc0893accf
round  2:839b23b83b5701ab095bafd162ec0ac7
Key [ 3]:8e15595edcf058af62498ee3c1dc6098
Key [ 4]:dd409c3444e94b9cc08396ae967542a0
round  3:c0a2010cc44f2139427f093f4f97ae68
added ->:d3b5f81d9eecd97bbe6ccd8e4f1f62e2
Key [ 5]:487deff5d519f6a6481e947b926f633c
Key [ 6]:5b4b6e3477ed5c2c01f6e607d3418963
round  4:1a5517a0efad3575931d8ea3bee8bd07
Key [ 7]:34b980088d2b5fd6b6a2aceeda99c9c4
Key [ 8]:e7d06d06078acc4ecdbc8da800b73078
round  5:d3ce1fdfe716d72c1075ff37a8a2093f
Key [ 9]:7d375bad245c3b757380021af8ecd408
Key [10]:14dac4bc2f4dc4929a6cceec47f4c3a3
round  6:47e90cb55be6e8dd0f583623c2f2257b
Key [11]:66cfda3c63e464b05e2e7e25f8743ad7
Key [12]:77cfccda1ad380b9fdf1df10846b50e7
round  7:f866ae6624f7abd4a4f5bd24b04b6d43
Key [13]:3e11dd84c031a470a8b66ec6214e44cf
```

```
Key [14]:2f03549bdb3c511eea70b65ddbb08253
round  8:02e8e17cf8be4837c9c40706b613dfa8
Key [15]:e2f331229ddfcc6e7bea08b01ab7e70c
Key [16]:b6b0c3738c5365bc77331b98b3fba2ab
Key [17]:f5b3973b636119e577c5c15c87bcfd19
Ka      :38ec0258134ec3f08461ae5c328968a1

========================================
```

9.5 FOUR TESTS OF E3

```
rand    :00000000000000000000000000000000
aco     :48afcdd4bd40fef76693b113
key     :00000000000000000000000000000000
round  1:00000000000000000000000000000000
Key [ 1]:00000000000000000000000000000000
Key [ 2]:4697b1baa3b7100ac537b3c95a28ac64
round  2:78d19f9307d2476a523ec7a8a026042a
Key [ 3]:ecabaac66795580df89af66e66dc053d
Key [ 4]:8ac3d8896ae9364943bfebd4969b68a0
round  3:600265247668dda0e81c07bbb30ed503
Key [ 5]:5d57921fd5715cbb22c1be7bbc996394
Key [ 6]:2a61b8343219fdfb1740e6511d41448f
round  4:d7552ef7cc9dbde568d80c2215bc4277
Key [ 7]:dd0480dee731d67f01a2f739da6f23ca
Key [ 8]:3ad01cd1303e12a1cd0fe0a8af82592c
round  5:fb06bef32b52ab8f2a4f2b6ef7f6d0cd
Key [ 9]:7dadb2efc287ce75061302904f2e7233
Key [10]:c08dcfa981e2c4272f6c7a9f52e11538
round  6:b46b711ebb3cf69e847a75f0ab884bdd
Key [11]:fc2042c708e409555e8c147660ffdfd7
Key [12]:fa0b21001af9a6b9e89e624cd99150d2
round  7:c585f308ff19404294f06b292e978994
Key [13]:18b40784ea5ba4c80ecb48694b4e9c35
Key [14]:454d54e5253c0c4a8b3fcca7db6baef4
round  8:2665fadb13acf952bf74b4ab12264b9f
Key [15]:2df37c6d9db52674f29353b0f011ed83
Key [16]:b60316733b1e8e70bd861b477e2456f1
Key [17]:884697b1baa3b7100ac537b3c95a28ac
round  1:5d3ecb17f26083df0b7f2b9b29aef87c
Key [ 1]:e9e5dfc1b3a79583e9e5dfc1b3a79583
Key [ 2]:7595bf57e0632c59f435c16697d4c864
round  2:de6fe85c5827233fe22514a16f321bd8
Key [ 3]:e31b96afcc75d286ef0ae257cbbc05b7
Key [ 4]:0d2a27b471bc0108c6263aff9d9b3b6b
round  3:7cd335b50d09d139ea6702623af85edb
added ->:211100a2ff6954e6e1e62df913a656a7
Key [ 5]:98d1eb5773cf59d75d3b17b3bc37c191
Key [ 6]:fd2b79282408ddd4ea0aa7511133336f
round  4:991dccb3201b5b1c4ceff65a3711e1e9
Key [ 7]:331227756638a41d57b0f7e071ee2a98
```

Sample Data

```
Key [ 8]:aa0dd8cc68b406533d0f1d64aabacf20
round  5:18768c7964818805fe4c6ecae8a38599
Key [ 9]:669291b0752e63f806fce76f10e119c8
Key [10]:ef8bdd46be8ee0277e9b78adef1ec154
round  6:82f9aa127a72632af43d1a17e7bd3a09
Key [11]:f3902eb06dc409cfd78384624964bf51
Key [12]:7d72702b21f97984a721c99b0498239d
round  7:1543d7870bf2d6d6efab3cbf62dca97d
Key [13]:532e60bceaf902c52a06c2c283ecfa32
Key [14]:181715e5192efb2a64129668cf5d9dd4
round  8:eee3e8744a5f8896de95831ed837ffd5
Key [15]:83017c1434342d4290e961578790f451
Key [16]:2603532f365604646ff65803795ccce5
Key [17]:882f7c907b565ea58dae1c928a0dcf41
kc      :cc802aecc7312285912e90af6a1e1154
----------------------------------------
rand    :950e604e655ea3800fe3eb4a28918087
aco     :68f4f472b5586ac5850f5f74
key     :34e86915d20c485090a6977931f96df5
round  1:950e604e655ea3800fe3eb4a28918087
Key [ 1]:34e86915d20c485090a6977931f96df5
Key [ 2]:8de2595003f9928efaf37e5229935bdb
round  2:d46f5a04c967f55840f83d1cdb5f9afc
Key [ 3]:46f05ec979a97cb6ddf842ecc159c04a
Key [ 4]:b468f0190a0a83783521deae8178d071
round  3:e16edede9cb6297f32e1203e442ac73a
Key [ 5]:8a171624dedbd552356094daaadcf12a
Key [ 6]:3085e07c85e4b99313f6e0c837b5f819
round  4:805144e55e1ece96683d23366fc7d24b
Key [ 7]:fe45c27845169a66b679b2097d147715
Key [ 8]:44e2f0c35f64514e8bec66c5dc24b3ad
round  5:edbaf77af070bd22e9304398471042f1
Key [ 9]:0d534968f3803b6af447eaf964007e7b
Key [10]:f5499a32504d739ed0b3c547e84157ba
round  6:0dab1a4c846aef0b65b1498812a73b50
Key [11]:e17e8e456361c46298e6592a6311f3fb
Key [12]:ec6d14da05d60e8abac807646931711f
round  7:1e7793cac7f55a8ab48bd33bc9c649e0
Key [13]:2b53dde3d89e325e5ff808ed505706ae
Key [14]:41034e5c3fb0c0d4f445f0cf23be79b0
round  8:3723768baa78b6a23ade095d995404da
Key [15]:e2ca373d405a7abf22b494f28a6fd247
Key [16]:74e09c9068c0e8f1c6902d1b70537c30
Key [17]:767a7f1acf75c3585a55dd4a428b2119
round  1:39809afb773efd1b7510cd4cb7c49f34
Key [ 1]:1d0d48d485abddd3798b483a82a0f878
Key [ 2]:aed957e600a5aed5217984dd5fef6fd8
round  2:6436ddbabe92655c87a7d0c12ae5e5f6
Key [ 3]:fee00bb0de89b6ef0a289696a4faa884
Key [ 4]:33ce2f4411db4dd9b7c42cc586b8a2ba
round  3:cec690f7e0aa5f063062301e049a5cc5
added ->:f7462a0c97e85c1d4572fd52b35efbf1
```

Sample Data

```
Key [ 5]:b5116f5c6c29e05e4acb4d02a46a3318
Key [ 6]:ff4fa1f0f73d1a3c67bc2298abc768f9
round  4:dcdfe942e9f0163fc24a4718844b417d
Key [ 7]:5453650c0819e001e48331ad0e9076e0
Key [ 8]:b4ff8dda778e26c0dce08349b81c09a1
round  5:265a16b2f766afae396e7a98c189fda9
Key [ 9]:f638fa294427c6ed94300fd823b31d10
Key [10]:1ccfa0bd86a9879b17d4bc457e3e03d6
round  6:628576b5291d53d1eb8611c8624e863e
Key [11]:0eaee2ef4602ac9ca19e49d74a76d335
Key [12]:6e1062f10a16e0d378476da3943842e9
round  7:d7b9c2e9b2d5ea5c27019324cae882b3
Key [13]:40be960bd22c744c5b23024688e554b9
Key [14]:95c9902cb3c230b44d14ba909730d211
round  8:97fb6065498385e47eb3df6e2ca439dd
Key [15]:10d4b6e1d1d6798aa00aa2951e32d58d
Key [16]:c5d4b91444b83ee578004ab8876ba605
Key [17]:1663a4f98e2862eddd3ec2fb03dcc8a4
kc      :c1beafea6e747e304cf0bd7734b0a9e2
----------------------------------------
rand    :6a8ebcf5e6e471505be68d5eb8a3200c
aco     :658d791a9554b77c0b2f7b9f
key     :35cf77b333c294671d426fa79993a133
round  1:6a8ebcf5e6e471505be68d5eb8a3200c
Key [ 1]:35cf77b333c294671d426fa79993a133
Key [ 2]:c4524e53b95b4bf2d7b2f095f63545fd
round  2:ade94ec585db0d27e17474b58192c87a
Key [ 3]:c99776768c6e9f9dd3835c52cea8d18a
Key [ 4]:f1295db23823ba2792f21217fc01d23f
round  3:da8dc1a10241ef9e6e069267cd2c6825
Key [ 5]:9083db95a6955235bbfad8aeefec5f0b
Key [ 6]:8bab6bc253d0d0c7e0107feab728ff68
round  4:e6665ca0772ceecbc21222ff7be074f8
Key [ 7]:2fa1f4e7a4cf3ccd876ec30d194cf196
Key [ 8]:267364be247184d5337586a19df8bf84
round  5:a857a9326c9ae908f53fee511c5f4242
Key [ 9]:9aef21965b1a6fa83948d107026134c7
Key [10]:d2080c751def5dc0d8ea353cebf7b973
round  6:6678748a1b5f21ac05cf1b117a7c342f
Key [11]:d709a8ab70b0d5a2516900421024b81e
Key [12]:493e4843805f1058d605c8d1025f8a56
round  7:766c66fe9c460bb2ae39ec01e435f725
Key [13]:b1ed21b71daea03f49fe74b2c11fc02b
Key [14]:0e1ded7ebf23c72324a0165a698c65c7
round  8:396e0ff7b2b9b7a3b35c9810882c7596
Key [15]:b3bf4841dc92f440fde5f024f9ce8be9
Key [16]:1c69bc6c2994f4c84f72be8f6b188963
Key [17]:bb7b66286dd679a471e2792270f3bb4d
round  1:45654f2f26549675287200f07cb10ec9
Key [ 1]:1e2a5672e66529e4f427b0682a3a34b6
Key [ 2]:974944f1ce0037b1febcf61a2bc961a2
round  2:990cd869c534e76ed4f4af7b3bfbc6c8
```

Sample Data

```
Key [ 3]:8147631fb1ce95d624b480fc7389f6c4
Key [ 4]:6e90a2db33d284aa13135f3c032aa4f4
round  3:ceb662f875aa6b94e8192b5989abf975
added ->:8b1bb1d753fe01e1c08b2ba9f55c07bc
Key [ 5]:cbad246d24e36741c46401e6387a05f9
Key [ 6]:dcf52aaec5713110345a41342c566fc8
round  4:d4e000be5de78c0f56ff218f3c1df61b
Key [ 7]:8197537aa9d27e67d17c16b182c8ec65
Key [ 8]:d66e00e73d835927a307a3ed79d035d8
round  5:9a4603bdef954cfaade2052604bed4e4
Key [ 9]:71d46257ecc1022bcd312ce6c114d75c
Key [10]:f91212fa528379651fbd2c32890c5e5f
round  6:09a0fd197ab81eb933eece2fe0132dbb
Key [11]:283acc551591fadce821b02fb9491814
Key [12]:ca5f95688788e20d94822f162b5a3920
round  7:494f455a2e7a5db861ece816d4e363e4
Key [13]:ba574aef663c462d35399efb999d0e40
Key [14]:6267afc834513783fef1601955fe0628
round  8:37a819f91c8380fb7880e640e99ca947
Key [15]:fdcd9be5450eef0f8737e6838cd38e2b
Key [16]:8cfbd9b8056c6a1ce222b92b94319b38
Key [17]:4f64c1072c891c39eeb95e63318462e0
kc      :a3032b4df1cceba8adc1a04427224299
----------------------------------------
rand    :5ecd6d75db322c75b6afbd799cb18668
aco     :63f701c7013238bbf88714ee
key     :b9f90c53206792b1826838b435b87d4d
round  1:5ecd6d75db322c75b6afbd799cb18668
Key [ 1]:b9f90c53206792b1826838b435b87d4d
Key [ 2]:15f74bbbde4b9d1e08f858721f131669
round  2:72abb85fc80c15ec2b00d72873ef9ad4
Key [ 3]:ef7fb29f0b01f82706c7439cc52f2dab
Key [ 4]:3003a6aecdee06b9ac295cce30dcdb93
round  3:2f10bab93a0f73742183c68f712dfa24
Key [ 5]:5fcdbb3afdf7df06754c954fc6340254
Key [ 6]:ddaa90756635579573fe8ca1f93d4a38
round  4:183b145312fd99d5ad08e7ca4a52f04e
Key [ 7]:27ca8a7fc703aa61f6d7791fc19f704a
Key [ 8]:702029d8c6e42950762317e730ec5d18
round  5:cbad52d3a026b2e38b9ae6fefffecc32
Key [ 9]:ff15eaa3f73f4bc2a6ccfb9ca24ed9c5
Key [10]:034e745246cd2e2cfc3bda39531ca9c5
round  6:ce5f159d0a1acaacd9fb4643272033a7
Key [11]:0a4d8ff5673731c3dc8fe87e39a34b77
Key [12]:637592fab43a19ac0044a21afef455a2
round  7:8a49424a10c0bea5aba52dbbffcbcce8
Key [13]:6b3fde58f4f6438843cdbe92667622b8
Key [14]:a10bfa35013812f39bf2157f1c9fca4e
round  8:f5e12da0e93e26a5850251697ec0b917
Key [15]:2228fe5384e573f48fdd19ba91f1bf57
Key [16]:5f174db2bc88925c0fbc6b5485bafc08
Key [17]:28ff90bd0dc31ea2bb479feb7d8fe029
```

Sample Data

```
round  1:0c75eed2b54c1cfb9ff522daef94ed4d
Key [ 1]:a21ceb92d3c027326b4de775865fe8d0
Key [ 2]:26f64558a9f0a1652f765efd546f3208
round  2:48d537ac209a6aa07b70000016c602e8
Key [ 3]:e64f9ef630213260f1f79745a0102ae5
Key [ 4]:af6a59d7cebfd0182dcca9a537c4add8
round  3:8b6d517ac893743a401b3fb7911b64e1
added ->:87e23fa87ddf90c1df10616d7eaf51ac
Key [ 5]:9a6304428b45da128ab64c8805c32452
Key [ 6]:8af4d1e9d80cb73ec6b44e9b6e4f39d8
round  4:9f0512260a2f7a5067efc35bf1706831
Key [ 7]:79cc2d138606f0fca4e549c34a1e6d19
Key [ 8]:803dc5cdde0efdbee7a1342b2cd4d344
round  5:0cfd7856edfafac51f29e86365de6f57
Key [ 9]:e8fa996448e6b6459ab51e7be101325a
Key [10]:2acc7add7b294acb444cd933f0e74ec9
round  6:2f1fa34bf352dc77c0983a01e8b7d622
Key [11]:f57de39e42182efd6586b86a90c86bb1
Key [12]:e418dfd1bb22ebf1bfc309cd27f5266c
round  7:ee4f7a53849bf73a747065d35f3752b1
Key [13]:80a9959133856586370854db6e0470b3
Key [14]:f4c1bc2f764a0193749f5fc09011a1ae
round  8:8fec6f7249760ebf69e370e9a4b80a92
Key [15]:d036cef70d6470c3f52f1b5d25b0c29d
Key [16]:d0956af6b8700888a1cc88f07ad226dc
Key [17]:1ce8b39c4c7677373c30849a3ee08794
kc      :ea520cfc546b00eb7c3a6cea3ecb39ed

========================================
```

Core System Package [Controller volume]

Part H

SECURITY SPECIFICATION

This document describes the specification of the security system which may be used at the link layer. The Encryption, Authentication and Key Generation schemes are specified. The requirements for the supporting process of random number generation are also specified.

CONTENTS

1 SECURITY OVERVIEW

Bluetooth wireless technology provides peer-to-peer communications over short distances. In order to provide usage protection and information confidentiality, the system provides security measures both at the application layer and the link layer. These measures are designed to be appropriate for a peer environment. This means that in each device, the authentication and encryption routines are implemented in the same way. Four different entities are used for maintaining security at the link layer: a Bluetooth device address, two secret keys, and a pseudo-random number that shall be regenerated for each new transaction. The four entities and their sizes are summarized in Table 1.1.

Entity	Size
BD_ADDR	48 bits
Private user key, authentication	128 bits
Private user key, encryption configurable length (byte-wise)	8-128 bits
RAND	128 bits

Table 1.1: Entities used in authentication and encryption procedures.

The Bluetooth device address (BD_ADDR) is the 48-bit address. The BD_ADDR can be obtained via user interactions, or, automatically, via an inquiry routine by a device.

The secret keys are derived during initialization and are never disclosed. The encryption key is derived from the authentication key during the authentication process. For the authentication algorithm, the size of the key used is always 128 bits. For the encryption algorithm, the key size may vary between 1 and 16 octets (8 - 128 bits). The size of the encryption key is configurable for two reasons. The first has to do with the many different requirements imposed on cryptographic algorithms in different countries – both with respect to export regulations and official attitudes towards privacy in general. The second reason is to facilitate a future upgrade path for the security without the need of a costly redesign of the algorithms and encryption hardware; increasing the effective key size is the simplest way to combat increased computing power at the opponent side.

The encryption key is entirely different from the authentication key (even though the latter is used when creating the former, as is described in Section 6.4 on page 807). Each time encryption is activated, a new encryption key shall be generated. Thus, the lifetime of the encryption key does not necessarily correspond to the lifetime of the authentication key.

It is anticipated that the authentication key will be more static in its nature than the encryption key – once established, the particular application running on the device decides when, or if, to change it. To underline the fundamental impor-

tance of the authentication key to a specific link, it is often be referred to as the link key.

The RAND is a pseudo-random number which can be derived from a random or pseudo-random process in the device. This is not a static parameter, it will change frequently.

In the remainder of this chapter, the terms user and application will be used interchangeably to designate the entity that is at either side.

2 RANDOM NUMBER GENERATION

Each device has a pseudo-random number generator. Pseudo-random numbers are used for many purposes within the security functions – for instance, for the challenge-response scheme, for generating authentication and encryption keys, etc. Ideally, a true random generator based on some physical process with inherent randomness should be used as a seed. Examples of such processes are thermal noise from a semiconductor or resistor and the frequency instability of a free running oscillator. For practical reasons, a software based solution with a pseudo-random generator is probably preferable. In general, it is quite difficult to classify the randomness of a pseudo-random sequence. Within this specification, the requirements placed on the random numbers used are non-repeating and randomly generated.

The expression ‘non-repeating’ means that it shall be highly unlikely that the value will repeat itself within the lifetime of the authentication key. For example, a non-repeating value could be the output of a counter that is unlikely to repeat during the lifetime of the authentication key, or a date/time stamp.

The expression ‘randomly generated’ means that it shall not be possible to predict its value with a chance that is significantly larger than 0 (e.g., greater than $1/2^L$ for a key length of L bits).

The LM may use such a generator for various purposes; i.e. whenever a random number is needed (such as the RANDs, the unit keys, K_{init}, K_{master}, and random back-off or waiting intervals).

3 KEY MANAGEMENT

It is important that the encryption key size within a specific device cannot be set by the user – this should be a factory preset entity. In order to prevent the user from over-riding the permitted key size, the Bluetooth baseband processing shall not accept an encryption key given from higher software layers. Whenever a new encryption key is required, it shall be created as defined in Section 6.4 on page 807.

Changing a link key shall also be done through the defined baseband procedures. Depending on what kind of link key it is, different approaches are required. The details are found in Section 3.2.7 on page 784.

3.1 KEY TYPES

The link key is a 128-bit random number which is shared between two or more parties and is the base for all security transactions between these parties. The link key itself is used in the authentication routine. Moreover, the link key is used as one of the parameters when the encryption key is derived.

In the following, a session is defined as the time interval for which the device is a member of a particular piconet. Thus, the session terminates when the device disconnects from the piconet.

The link keys are either semi-permanent or temporary. A semi-permanent link key may be stored in non-volatile memory and may be used after the current session is terminated. Consequently, once a semi-permanent link key is defined, it may be used in the authentication of several subsequent connections between the devices sharing it. The designation semi-permanent is justified by the possibility of changing it. How to do this is described in Section 3.2.7 on page 784.

The lifetime of a temporary link key is limited by the lifetime of the current session – it shall not be reused in a later session. Typically, in a point-to-multipoint configuration where the same information is to be distributed securely to several recipients, a common encryption key is useful. To achieve this, a special link key (denoted master key) may temporarily replace the current link keys. The details of this procedure are found in Section 3.2.6 on page 783.

In the following, the current link key is the link key in use at the current moment. It can be semi-permanent or temporary. Thus, the current link key is used for all authentications and all generation of encryption keys in the on-going connection (session).

In order to accommodate different types of applications, four types of link keys have been defined:

- the combination key K_{AB}
- the unit key K_A
- the temporary key K_{master}
- the initialization key K_{init}

Note: the use of unit keys is deprecated since it is implicitly insecure.

In addition to these keys there is an encryption key, denoted K_c. This key is derived from the current link key. Whenever encryption is activated by an LM command, the encryption key shall be changed automatically. The purpose of separating the authentication key and encryption key is to facilitate the use of a shorter encryption key without weakening the strength of the authentication procedure. There are no governmental restrictions on the strength of authentication algorithms. However, in some countries, such restrictions exist on the strength of encryption algorithms.

The combination key K_{AB} and the unit key K_A are functionally indistinguishable; the difference is in the way they are generated. The unit key K_A is generated in, and therefore dependent on, a single device A. The unit key shall be generated once at installation of the device; thereafter, it is very rarely changed. The combination key is derived from information in both devices A and B, and is therefore always dependent on two devices. The combination key is derived for each new combination of two devices.

It depends on the application or the device whether a unit key or a combination key is used. Devices which have little memory to store keys, or are installed in equipment that will be accessible to a large group of users, should use their own unit key. In that case, they only have to store a single key. Applications that require a higher security level should use the combination keys. These applications will require more memory since a combination key for each link to a different device has to be stored.

The master key, K_{master}, shall only be used during the current session. It shall only replace the original link key temporarily. For example, this may be utilized when a master wants to reach more than two devices simultaneously using the same encryption key, see Section 3.2.6 on page 783.

The initialization key, K_{init}, shall be used as the link key during the initialization process when no combination or unit keys have been defined and exchanged yet or when a link key has been lost. The initialization key protects the transfer of initialization parameters. The key is derived from a random number, an L-octet PIN code, and a BD_ADDR. This key shall only be used during initialization.

The PIN may be a fixed number provided with the device (for example when there is no user interface as in a PSTN plug). Alternatively, the PIN can be selected by the user, and then entered in both devices that are to be matched. The latter procedure should be used when both devices have a user interface, for example a phone and a laptop. Entering a PIN in both devices is more secure than using a fixed PIN in one of the devices, and should be used whenever possible. Even if a fixed PIN is used, it shall be possible to change the PIN; this is in order to prevent re-initialization by users who once got hold of the PIN. If no PIN is available, a default value of zero may be used. The length of this default PIN is one byte, PIN(default) = 0x00. This default PIN may be provided by the host.

For many applications the PIN code will be a relatively short string of numbers. Typically, it may consist of only four decimal digits. Even though this gives sufficient security in many cases, there exist countless other, more sensitive, situations where this is not reliable enough. Therefore, the PIN code may be chosen to be any length from 1 to 16 octets. For the longer lengths, the devices exchanging PIN codes may not use mechanical (i.e. human) interaction, but rather may use software at the application layer. For example, this can be a Diffie-Hellman key agreement, where the exchanged key is passed on to the K_{init} generation process in both devices, just as in the case of a shorter PIN code.

3.2 KEY GENERATION AND INITIALIZATION

The link keys must be generated and distributed among the devices in order to be used in the authentication procedure. Since the link keys shall be secret, they shall not be obtainable through an inquiry routine in the same way as the Bluetooth device addresses. The exchange of the keys takes place during an initialization phase which shall be carried out separately for each two devices that are using authentication and encryption. The initialization procedures consist of the following five parts:

- generation of an initialization key
- generation of link key
- link key exchange
- authentication
- generation of encryption key in each device (optional)

After the initialization procedure, the devices can proceed to communicate, or the link can be disconnected. If encryption is implemented, the E_0 algorithm shall be used with the proper encryption key derived from the current link key. For any new connection established between devices A and B, they should use the common link key for authentication, instead of once more deriving K_{init} from the PIN. A new encryption key derived from that particular link key shall be created next time encryption is activated.

If no link key is available, the LM shall automatically start an initialization procedure.

3.2.1 Generation of initialization key, K_{init}

A link key is used temporarily during initialization, the initialization key K_{init}. This key shall be derived by the E_{22} algorithm from a BD_ADDR, a PIN code, the length of the PIN (in octets), and a random number IN_RAND. The principle is depicted in Figure 6.4 on page 807. The 128-bit output from E_{22} shall be used for key exchange during the generation of a link key. When the devices have performed the link key exchange, the initialization key shall be discarded.

When the initialization key is generated, the PIN is augmented with the BD_ADDR. If one device has a fixed PIN the BD_ADDR of the other device shall be used. If both devices have a variable PIN the BD_ADDR of the device that received IN_RAND shall be used. If both devices have a fixed PIN they cannot be paired. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD_ADDR will be used. This procedure ensures that K_{init} depends on the identity of the device with a variable PIN. A fraudulent device may try to test a large number of PINs by claiming another BD_ADDR each time. It is the application's responsibility to take countermeasures against this threat. If the device address is kept fixed, the waiting interval before the next try may be increased exponentially (see Section 5.1 on page 799).

The details of the E_{22} algorithm can be found in Section 6.3 on page 805.

3.2.2 Authentication

The authentication procedure shall be carried out as described in Section 5 on page 797. During each authentication, a new AU_RAND_A shall be issued.

Mutual authentication is achieved by first performing the authentication procedure in one direction and then immediately performing the authentication procedure in the opposite direction.

As a side effect of a successful authentication procedure an auxiliary parameter, the Authenticated Ciphering Offset (ACO), will be computed. The ACO shall be used for ciphering key generation as described in Section 3.2.5 on page 782.

The claimant/verifier status is determined by the LM.

3.2.3 Generation of a unit key

A unit key K_A shall be generated when the device is in operation for the first time; i.e. not during each initialization. The unit key shall be generated by the E_{21} algorithm as described in Section 6.3 on page 805. Once created, the unit key should be stored in non-volatile memory and very rarely changed. If after initialization

the unit key is changed, any previously initialized devices will possess a wrong link key. At initialization, the application must determine which of the two parties will provide the unit key as the link key. Typically, this will be the device with restricted memory capabilities, since this device only has to remember its own unit key. The unit key shall be transferred to the other party and then stored as the link key for that particular party. So, for example in Figure 3.1 on page 781, the unit key of device A, K_A, is being used as the link key for the connection A-B; device A sends the unit key K_A to device B; device B will store K_A as the link key K_{BA}. For another initialization, for example with device C, device A will reuse its unit key K_A, whereas device C stores it as K_{CA}.

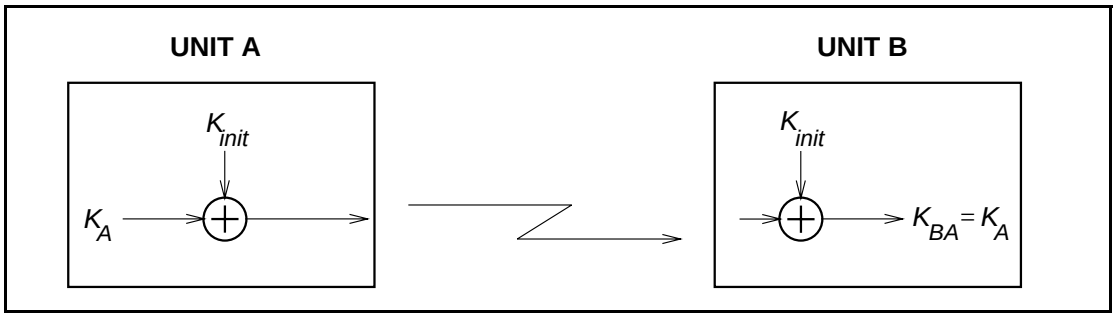


Figure 3.1: Generation of unit key. When the unit key has been exchanged, the initialization key is discarded in both devices.

3.2.4 Generation of a combination key

To use a combination key, it is first generated during the initialization procedure. The combination key is the combination of two numbers generated in device A and B, respectively. First, each device shall generate a random number, LK_RAND_A and LK_RAND_B. Then, utilizing E_{21} with the random number and their own BD_ADDRs, the two random numbers

$$LK\_K_A = E_{21}(LK\_RAND_A, BD\_ADDR_A), \quad \text{(EQ 1)}$$

and

$$LK\_K_B = E_{21}(LK\_RAND_B, BD\_ADDR_B), \quad \text{(EQ 2)}$$

shall be created in device A and device B, respectively. These numbers constitute the devices' contribution to the combination key that is to be created. Then, the two random numbers LK_RAND_A and LK_RAND_B shall be exchanged securely by XORing with the current link key, K. Thus, device A shall send $K \oplus LK\_RAND_A$ to device B, while device B shall send $K \oplus LK\_RAND_B$ to device A. If this is done during the initialization phase the link key $K = K_{init}$.

When the random numbers LK_RAND_A and LK_RAND_B have been mutually exchanged, each device shall recalculate the other device's contribution to the combination key. This is possible since each device knows the Bluetooth device address of the other device. Thus, device A shall calculate (EQ 2) on page 781 and device B shall calculate (EQ 1) on page 781. After this, both devices shall combine the two numbers to generate the 128-bit link key. The combining operation is a simple bitwise modulo-2 addition (i.e. XOR). The result shall be stored in device A as the link key K_{AB} and in device B as the link key K_{BA}. When both devices have derived the new combination key, a mutual authentication procedure shall be initiated to confirm the success of the transaction. The old link key shall be discarded after a successful exchange of a new combination key. The message flow between master and slave and the principle for creating the combination key is depicted in Figure 3.2 on page 782.

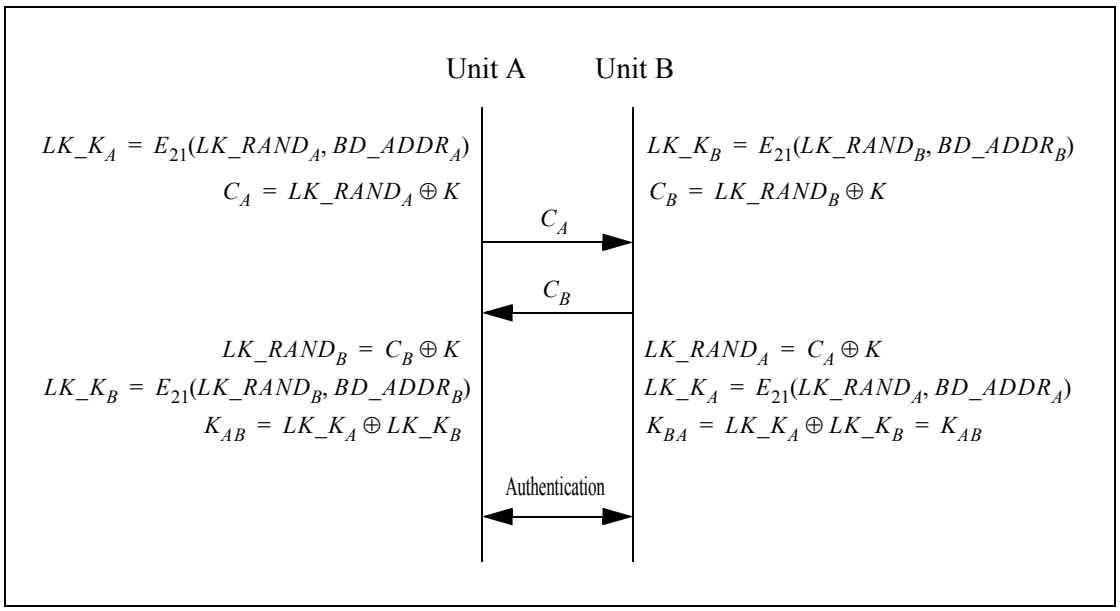


Figure 3.2: Generating a combination key. The old link key (K) is discarded after the exchange of a new combination key has succeeded

3.2.5 Generating the encryption key

The encryption key, K_C, is derived by algorithm E_3 from the current link key, a 96-bit Ciphering OFfset number (COF), and a 128-bit random number. The COF is determined in one of two ways. If the current link key is a master key, then COF shall be derived from the master BD_ADDR. Otherwise the value of COF shall be set to the value of ACO as computed during the authentication procedure. Therefore:[1]

1. $x \cup y$ denotes the concatenation of x and y.

$$COF = \begin{cases} BD\_ADDR \cup BD\_ADDR, & \text{if link key is a master key} \\ ACO, & \text{otherwise.} \end{cases} \quad \text{(EQ 3)}$$

There is an explicit call of E_3 when the LM activates encryption. Consequently, the encryption key is automatically changed each time the device enters encryption mode. The details of the key generating function E_3 can be found in Section 6.4 on page 807.

3.2.6 Point-to-multipoint configuration

It is possible for the master to use separate encryption keys for each slave in a point-to-multipoint configuration with ciphering activated. Then, if the application requires more than one slave to listen to the same payload, each slave must be addressed individually. This can cause unwanted capacity loss for the piconet. Moreover, a slave might not be capable of switching between two or more encryption keys in real time (e.g., after looking at the LT_ADDR in the header). Thus, the master cannot use different encryption keys for broadcast messages and individually addressed traffic. Therefore, the master may tell several slave devices to use a common link key (and, hence, indirectly also to use a common encryption key) and may then broadcast the information encrypted. For many applications, this key is only of temporary interest. In the following discussion, this key is denoted by K_{master}.

The transfer of necessary parameters shall be protected by the routine described in Section 3.2.8 on page 784. After the confirmation of successful reception in each slave, the master shall issue a command to the slaves to replace their respective current link key by the new (temporary) master key. Before encryption can be activated, the master shall also generate and distribute a common EN_RAND to all participating slaves. Using this random number and the newly derived master key, each slave shall generate a new encryption key.

Note that the master must negotiate the encryption key length to use individually with each slave that will use the master key. If the master has already negotiated with some of these slaves, it has knowledge of the sizes that can be accepted. There may be situations where the permitted key lengths of some devices are incompatible. In that case, the master must exclude the limiting device from the group.

When all slaves have received the necessary data, the master can communicate information on the piconet securely using the encryption key derived from the new temporary link key. Each slave in possession of the master key can eavesdrop on all encrypted traffic, not only the traffic intended for itself. The master may tell all participants to fall back to their old link keys simultaneously.

3.2.7 Modifying the link keys

A link key based on a unit key can be changed. The unit key is created once during first use. Typically, the link key should be changed rather than the unit key, as several devices may share the same unit key as link key (e.g. a printer whose unit key is distributed to all users using the printer's unit key as link key). Changing the unit key will require re-initialization of all devices connecting. Changing the unit key can be justified in some circumstances, e.g. to deny access to all previously allowed devices.

If the key change concerns combination keys, then the procedure is straightforward. The change procedure is identical to the procedure described in Figure 3.2 on page 782, using the current value of the combination key as link key. This procedure can be carried out at any time after the authentication and encryption start. Since the combination key corresponds to a single link, it can be modified each time this link is established. This will improve the security of the system since then old keys lose their validity after each session.

Starting up an entirely new initialization procedure is also possible. In that case, user interaction is necessary since a PIN will be required in the authentication and encryption procedures.

3.2.8 Generating a master key

The key-change routines described so far are semi-permanent. To create the master link key, which can replace the current link key during a session (see Section 3.2.6 on page 783), other means are needed. First, the master shall create a new link key from two 128-bit random numbers, RAND1 and RAND2. This shall be done by

$$K_{master} = E_{22}(\text{RAND1}, \text{RAND2}, 16). \quad \text{(EQ 4)}$$

This key is a 128-bit random number. The reason for using the output of E_{22} and not directly choosing a random number as the key, is to avoid possible problems with degraded randomness due to a poor implementation of the random number generator within the device.

Then, a third random number, RAND, shall be transmitted to the slave. Using E_{22} with the current link key and RAND as inputs, both the master and the slave shall compute a 128-bit overlay. The master shall send the bitwise XOR of the overlay and the new link key to the slave. The slave, who knows the overlay, shall recalculate K_{master}. To confirm the success of this transaction, the devices shall perform a mutual authentication procedure using the new link key. This procedure shall then be repeated for each slave that receives the new link key. The ACO values from the authentications shall not replace the current ACO, as this ACO is needed to (re)compute a ciphering key when the master falls back to the previous (non-temporary) link key.

The master activates encryption by an LM command. Before activating encryption, the master shall ensure that all slaves receive the same random number, EN_RAND, since the encryption key is derived through the means of E_3 individually in all participating devices. Each slave shall compute a new encryption key as follows:

$$K_C = E_3(K_{master}, \text{EN\_RAND}, \text{COF}) \quad \text{(EQ 5)}$$

where the value of COF shall be derived from the master's BD_ADDR as specified by equation (EQ 3) on page 783. The details of the encryption key generating function are described in Section 6.4 on page 807. The message flow between the master and the slave when generating the master key is depicted in Figure 3.3. Note that in this case the ACO produced during the authentication is not used when computing the ciphering key.

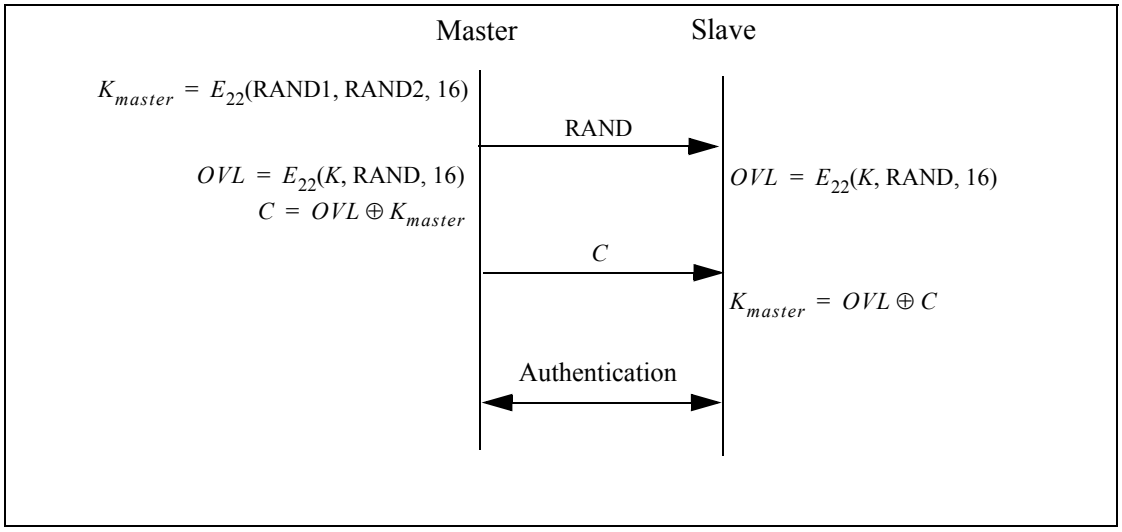


Figure 3.3: Master link key distribution and computation of the corresponding encryption key.

4 ENCRYPTION

User information can be protected by encryption of the packet payload; the access code and the packet header shall never be encrypted. The encryption of the payload shall be carried out with a stream cipher called E_0 that shall be re-synchronized for every payload. The overall principle is shown in Figure 4.1 on page 787.

The stream cipher system E_0 shall consist of three parts:

- the first part performs initialization (generation of the payload key). The payload key generator shall combine the input bits in an appropriate order and shall shift them into the four LFSRs used in the key stream generator.
- the second part generates the key stream bits and shall use a method derived from the summation stream cipher generator attributable to Massey and Rueppel. The second part is the main part of the cipher system, as it will also be used for initialization.
- the third part performs encryption and decryption.

The Massey and Rueppel method has been thoroughly investigated, and there exist good estimates of its strength with respect to presently known methods for cryptanalysis. Although the summation generator has weaknesses that can be used in correlation attacks, the high re-synchronization frequency will disrupt such attacks.

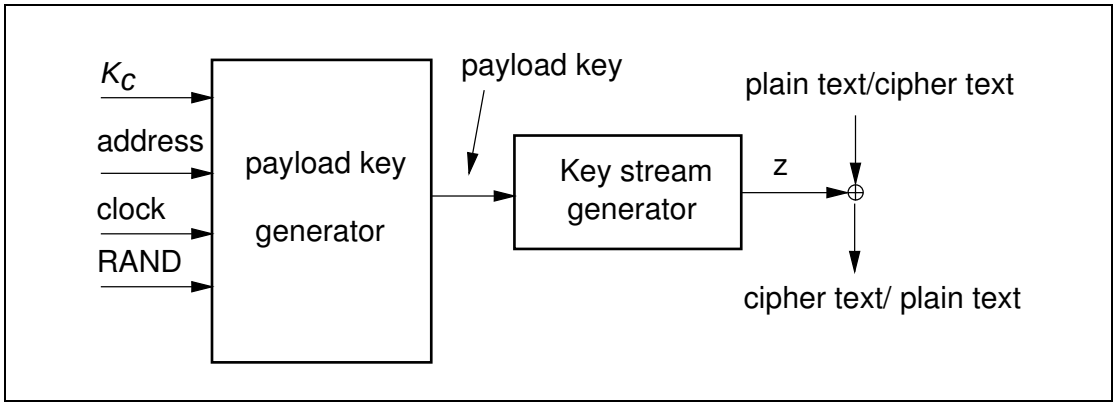


Figure 4.1: Stream ciphering for Bluetooth with E_0.

4.1 ENCRYPTION KEY SIZE NEGOTIATION

Each device implementing the baseband specification shall have a parameter defining the maximal allowed key length, L_{max}, $1 \leq L_{max} \leq 16$ (number of octets in the key). For each application using encryption, a number L_{min} shall be defined indicating the smallest acceptable key size for that particular application. Before generating the encryption key, the devices involved shall negotiate to decide the key size to use.

The master shall send a suggested value, $L_{sug}^{(M)}$, to the slave. Initially, the suggested value shall be set to $L_{max}^{(M)}$. If $L_{min}^{(S)} \leq L_{sug}^{(M)}$, and, the slave supports the suggested length, the slave shall acknowledge and this value shall be the length of the encryption key for this link. However, if both conditions are not fulfilled, the slave shall send a new proposal, $L_{sug}^{(S)} < L_{sug}^{(M)}$, to the master. This value shall be the largest among all supported lengths less than the previous master suggestion. Then, the master shall perform the corresponding test on the slave suggestion. This procedure shall be repeated until a key length agreement is reached, or, one device aborts the negotiation. An abort may be caused by lack of support for L_{sug} and all smaller key lengths, or if $L_{sug} < L_{min}$ in one of the devices. In case of an abort link encryption can not be employed.

The possibility of a failure in setting up a secure link is an unavoidable consequence of letting the application decide whether to accept or reject a suggested key size. However, this is a necessary precaution. Otherwise a fraudulent device could enforce a weak protection on a link by claiming a small maximum key size.

4.2 ENCRYPTION OF BROADCAST MESSAGES

There may be three settings for the baseband regarding encryption:

1. No encryption.
   This is the default setting. No messages are encrypted
2. Point-to-point only encryption.
   Broadcast messages are not encrypted. This may be enabled either during the connection establishment procedure or after the connection has been established.
3. Point-to-point and broadcast encryption.
   All messages are encrypted. This may be enabled after the connection has been established only. This setting should not be enabled unless all affected links share the same master link key as well as the same EN_RAND value, both used in generating the encryption key.

4.3 ENCRYPTION CONCEPT

Broadcast traffic	Individually addressed traffic
No encryption	No encryption
No encryption	Encryption, K_{master}
Encryption, K_{master}	Encryption, K_{master}

Table 4.1: Possible encryption modes for a slave in possession of a master key.

For the encryption routine, a stream cipher algorithm is used in which ciphering bits are bit-wise modulo-2 added to the data stream to be sent over the air interface. The payload is ciphered after the CRC bits are appended, but, prior to the FEC encoding.

Each packet payload shall be ciphered separately. The cipher algorithm E_0 uses the master Bluetooth device address (BD_ADDR), 26 bits of the master real-time clock (CLK $_{26\text{-}1}$) and the encryption key K_C as input, see Figure 4.2 on page 790 (where it is assumed that device A is the master).

The encryption key K_C is derived from the current link key, COF, and a random number, EN_RAND_A (see Section 6.4 on page 807). The random number shall be issued by the master before entering encryption mode. Note that EN_RAND_A is publicly known since it is transmitted as plain text over the air.

Within the E_0 algorithm, the encryption key K_C is modified into another key denoted K'_C. The maximum effective size of this key shall be factory preset and may be set to any multiple of eight between one and sixteen (8-128 bits). The procedure for deriving the key is described in Section 4.5 on page 793.

The real-time clock is incremented for each slot. The E_0 algorithm shall be re-initialized at the start of each new packet (i.e. for Master-to-Slave as well as for Slave-to-Master transmission). By using $CLK_{26\text{-}1}$ at least one bit is changed between two transmissions. Thus, a new keystream is generated after each re-initialization. For packets covering more than a single slot, the Bluetooth clock as found in the first slot shall be used for the entire packet.

The encryption algorithm E_0 generates a binary keystream, K_{cipher}, which shall be modulo-2 added to the data to be encrypted. The cipher is symmetric; decryption shall be performed in exactly the same way using the same key as used for encryption.

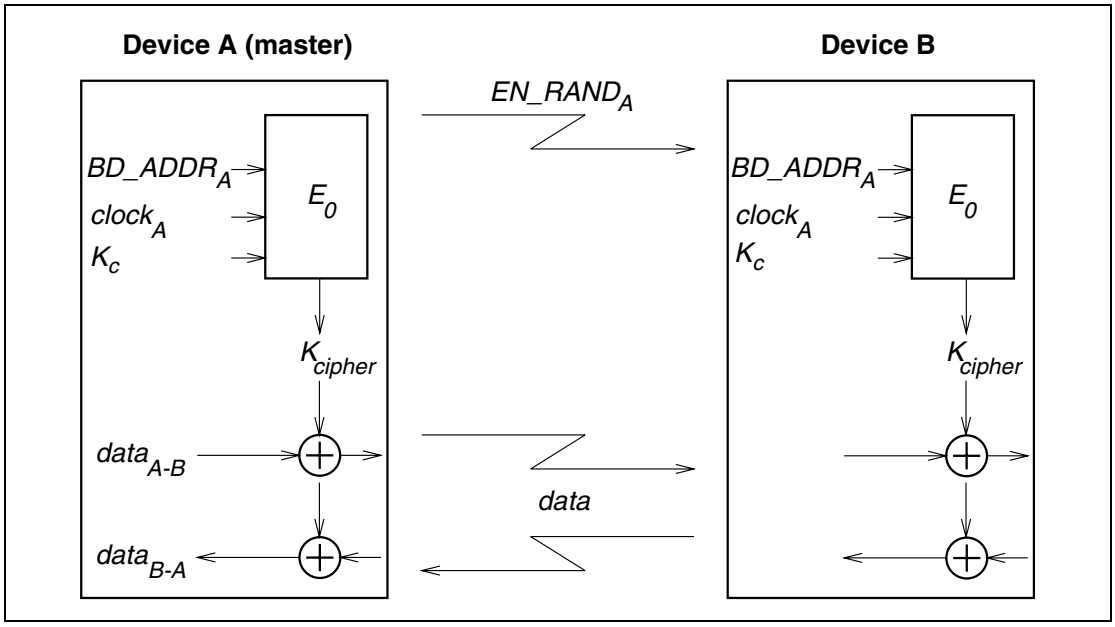


Figure 4.2: Functional description of the encryption procedure

4.4 ENCRYPTION ALGORITHM

The system uses linear feedback shift registers (LFSRs) whose output is combined by a simple finite state machine (called the summation combiner) with 16 states. The output of this state machine is the key stream sequence, or, during initialization phase, the randomized initial start value. The algorithm uses an encryption key K_C, a 48-bit Bluetooth address, the master clock bits $CLK_{26\text{-}1}$, and a 128-bit RAND value. Figure 4.3 on page 791 shows the setup.

There are four LFSRs ($LFSR_1$,...,$LFSR_4$) of lengths $L_1 = 25$, $L_2 = 31$, $L_3 = 33$, and, $L_4 = 39$, with feedback polynomials as specified in Table 4.2 on page 791. The total length of the registers is 128. These polynomials are all primitive. The Hamming weight of all the feedback polynomials is chosen to be five – a reasonable trade-off between reducing the number of required XOR gates in the hardware implementation and obtaining good statistical properties of the generated sequences.

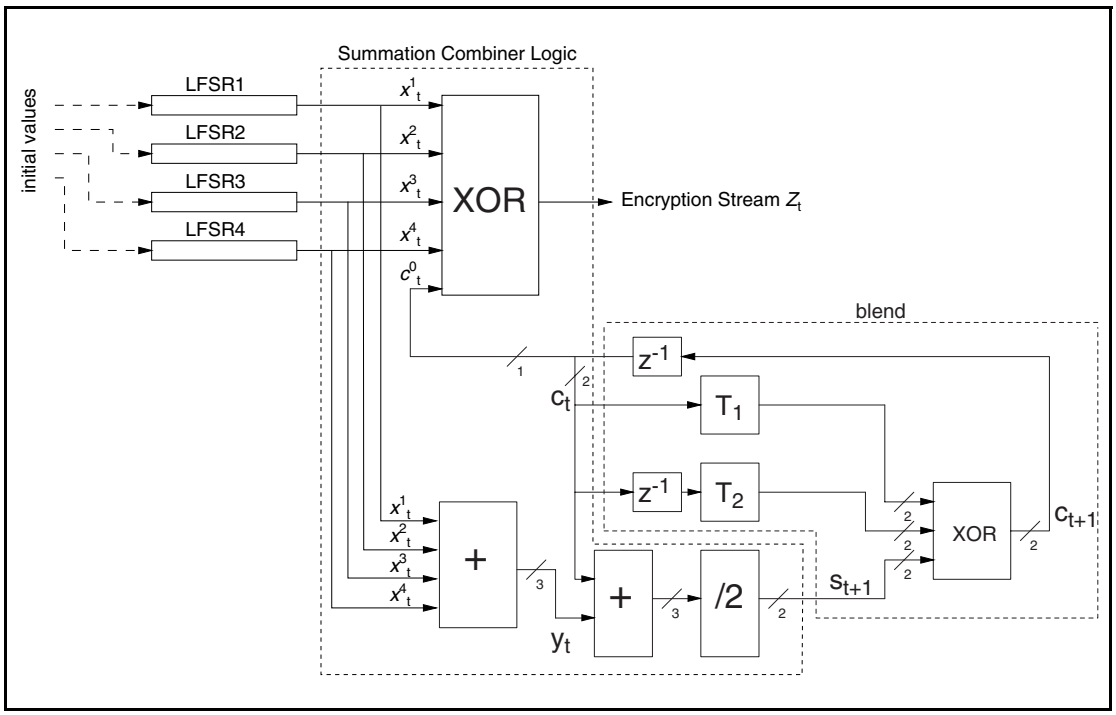


Figure 4.3: Concept of the encryption engine.

i	L_i	**feedback** $f_i(t)$	**weight**
1	25	$t^{25} + t^{20} + t^{12} + t^8 + 1$	5
2	31	$t^{31} + t^{24} + t^{16} + t^{12} + 1$	5
3	33	$t^{33} + t^{28} + t^{24} + t^4 + 1$	5
4	39	$t^{39} + t^{36} + t^{28} + t^4 + 1$	5

Table 4.2: The four primitive feedback polynomials.

Let x_t^i denote the t^{th} symbol of LFSR_i. The value y_t is derived from the four-tuple $x_t^1, \ldots, x_t^4$ using the following equation:

$$y_t = \sum_{i=1}^{4} x_t^i, \qquad \text{(EQ 6)}$$

where the sum is over the integers. Thus y_t can take the values 0,1,2,3, or 4. The output of the summation generator is obtained by the following equations:

$$z_t = x_t^1 \oplus x_t^2 \oplus x_t^3 \oplus x_t^4 \oplus c_t^0 \in \{0, 1\}, \qquad \text{(EQ 7)}$$

$$s_{t+1} = (s^1_{t+1}, s^0_{t+1}) = \left\lfloor \frac{y_t + c_t}{2} \right\rfloor \in \{0, 1, 2, 3\}, \quad \text{(EQ 8)}$$

$$c_{t+1} = (c^1_{t+1}, c^0_{t+1}) = s_{t+1} \oplus T_1[c_t] \oplus T_2[c_{t-1}], \quad \text{(EQ 9)}$$

where $T_1[.]$ and $T_2[.]$ are two different linear bijections over GF(4). Suppose GF(4) is generated by the irreducible polynomial $x^2 + x + 1$, and let α be a zero of this polynomial in GF(4). The mappings T_1 and T_2 are now defined as:

$$T_1: \mathrm{GF}(4) \rightarrow \mathrm{GF}(4)$$
$$x \mapsto x$$
$$T_2: \mathrm{GF}(4) \rightarrow \mathrm{GF}(4)$$
$$x \mapsto (\alpha + 1)x.$$

The elements of GF(4) can be written as binary vectors. This is summarized in Table 4.3.

x	$T_1[x]$	$T_2[x]$
00	00	00
01	01	11
10	10	01
11	11	10

Table 4.3: The mappings T_1 and T_2.

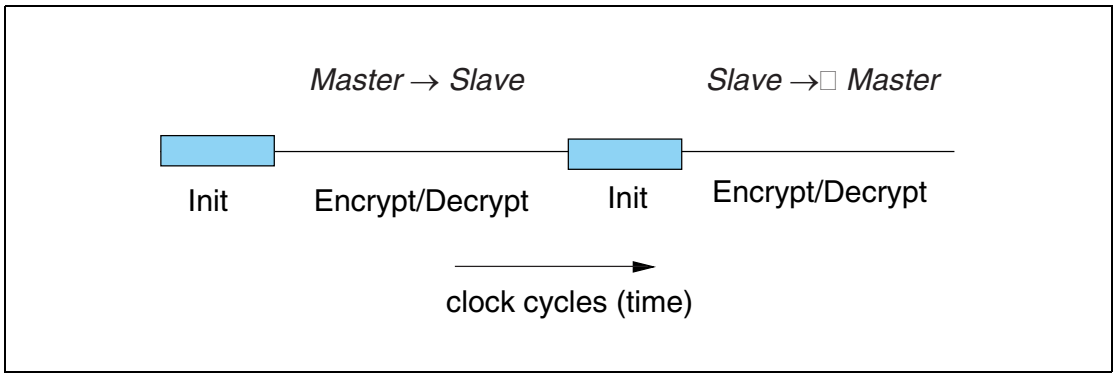
Since the mappings are linear, they can be implemented using XOR gates; i.e.

$$T_1: \quad (x_1, x_0) \mapsto (x_1, x_0),$$
$$T_2: \quad (x_1, x_0) \mapsto (x_0, x_1 \oplus x_0).$$

4.4.1 The operation of the cipher

Figure 4.4 on page 792 gives an overview of the operation in time. The encryption algorithm shall run through the initialization phase before the start of transmission or reception of a new packet. Thus, for multislot packets the cipher is initialized using the clock value of the first slot in the multislot sequence.

Figure 4.4: Overview of the operation of the encryption engine. Between each start of a packet (TX or RX), the LFSRs are re-initialized.



4.5 LFSR INITIALIZATION

The key stream generator is loaded with an initial value for the four LFSRs (in total 128 bits) and the 4 bits that specify the values of c_0 and c_{-1}. The 132 bit initial value is derived from four inputs by using the key stream generator. The input parameters are the key K_C, a 128-bit random number RAND, a 48-bit Bluetooth device address, and the 26 master clock bits $CLK_{26\text{-}1}$.

The effective length of the encryption key may vary between 8 and 128 bits. Note that the actual key length as obtained from E_3 is 128 bits. Then, within E_0, the key length may be reduced by a modulo operation between K_C and a polynomial of desired degree. After reduction, the result is encoded with a block code in order to distribute the starting states more uniformly. The operation shall be as defined in (EQ 10) on page 794.

When the encryption key has been created the LFSRs are loaded with their initial values. Then, 200 stream cipher bits are created by operating the generator. Of these bits, the last 128 are fed back into the key stream generator as an initial value of the four LFSRs. The values of c_t and c_{t-1} are kept. From this point on, when clocked the generator produces the encryption (decryption) sequence which is bitwise XORed to the transmitted (received) payload data.

In the following, octet i of a binary sequence X is $X[i]$. Bit 0 of X is the LSB. Then, the LSB of $X[i]$ corresponds to bit $8i$ of the sequence X, the MSB of $X[i]$ is bit $8i+7$ of X. For instance, bit 24 of the Bluetooth device address is the LSB of BD_ADDR[3].

The details of the initialization shall be as follows:

1. Create the encryption key to use from the 128-bit secret key K_C and the 128-bit publicly known EN_RAND. Let L, $1 \le L \le 16$, be the

effective key length in number of octets. The resulting encryption key is K'_C:

$$K'_C(x) = g_2^{(L)}(x)(K_C(x) \mod g_1^{(L)}(x)), \quad \text{(EQ 10)}$$

where $\deg(g_1^{(L)}(x)) = 8L$ and $\deg(g_2^{(L)}(x)) \leq 128 - 8L$. The polynomials are defined in Table 4.4.

2. Shift the 3 inputs K'_C, the Bluetooth device address, the clock, and the six-bit constant 111001 into the LFSRs. In total, 208 bits are shifted in:

   a) Open all switches shown in Figure 4.5 on page 795;

   b) Arrange inputs bits as shown in Figure 4.5; Set the content of all shift register elements to zero. Set $t = 0$.

   c) Start shifting bits into the LFSRs. The rightmost bit at each level of Figure 4.5 is the first bit to enter the corresponding LFSR.

   d) When the first input bit at level i reaches the rightmost position of $LFSR_i$, close the switch of this LFSR.

   e) At $t = 39$ (when the switch of $LFSR_4$ is closed), reset both blend registers $c_{39} = c_{39-1} = 0$; Up to this point, the content of c_t and c_{t-1} has been of no concern. However, their content will now be used in computing the output sequence.

   f) From now on output symbols are generated. The remaining input bits are continuously shifted into their corresponding shift registers. When the last bit has been shifted in, the shift register is clocked with input = 0;

      Note: When finished, $LFSR_1$ has effectively clocked 30 times with feedback closed, $LFSR_2$ has clocked 24 times, $LFSR_3$ has clocked 22 times, and $LFSR_4$ has effectively clocked 16 times with feedback closed.

3. To mix initial data, continue to clock until 200 symbols have been produced with all switches closed ($t = 239$);

4. Keep blend registers c_t and c_{t-1}, make a parallel load of the last 128 generated bits into the LFSRs according to Figure 4.6 at $t = 240$;

After the parallel load in item 4, the blend register contents shall be updated for each subsequent clock.

L	deg	$g_1^{(L)}$	deg	$g_2^{(L)}$
1	[8]	00000000 00000000 00000000 0000011d	[119]	00e275a0 abd218d4 cf928b9b bf6cb08f
2	[16]	00000000 00000000 00000000 0001003f	[112]	0001e3f6 3d7659b3 7f18c258 cff6efef
3	[24]	00000000 00000000 00000000 010000db	[104]	000001be f66c6c3a b1030a5a 1919808b
4	[32]	00000000 00000000 00000001 000000af	[96]	00000001 6ab89969 de17467f d3736ad9

Table 4.4: Polynomials used when creating K'_C. [1]

L	deg	$g_1^{(L)}$	deg	$g_2^{(L)}$
5	[40]	00000000 00000000 00000100 00000039	[88]	00000000 01630632 91da50ec 55715247
6	[48]	00000000 00000000 00010000 00000291	[77]	00000000 00002c93 52aa6cc0 54468311
7	[56]	00000000 00000000 01000000 00000095	[71]	00000000 000000b3 f7fffce2 79f3a073
8	[64]	00000000 00000001 00000000 0000001b	[63]	00000000 00000000 a1ab815b c7ec8025
9	[72]	00000000 00000100 00000000 00000609	[49]	00000000 00000000 0002c980 11d8b04d
10	[80]	00000000 00010000 00000000 00000215	[42]	00000000 00000000 0000058e 24f9a4bb
11	[88]	00000000 01000000 00000000 0000013b	[35]	00000000 00000000 0000000c a76024d7
12	[96]	00000001 00000000 00000000 000000dd	[28]	00000000 00000000 00000000 1c9c26b9
13	[104]	00000100 00000000 00000000 0000049d	[21]	00000000 00000000 00000000 0026d9e3
14	[112]	00010000 00000000 00000000 0000014f	[14]	00000000 00000000 00000000 00004377
15	[120]	01000000 00000000 00000000 000000e7	[7]	00000000 00000000 00000000 00000089
16	[128]	1 00000000 00000000 00000000 00000000	[0]	00000000 00000000 00000000 00000001

Table 4.4: Polynomials used when creating K'_c. . [1]

1. All polynomials are in hexadecimal notation. The LSB is in the rightmost position.

In Figure 4.5, all bits are shifted into the LFSRs, starting with the least significant bit (LSB). For instance, from the third octet of the address, BD_ADDR[2], first BD_ADDR_{16} is entered, followed by BD_ADDR_{17}, etc. Furthermore, CL_0 corresponds to CLK_1,..., CL_{25} corresponds to CLK_{26}.

Note that the output symbols $x_t^i, i = 1, \ldots, 4$ are taken from the positions 24, 24, 32, and 32 for $LFSR_1$, $LFSR_2$,$LFSR_3$, and $LFSR_4$, respectively (counting the leftmost position as number 1).

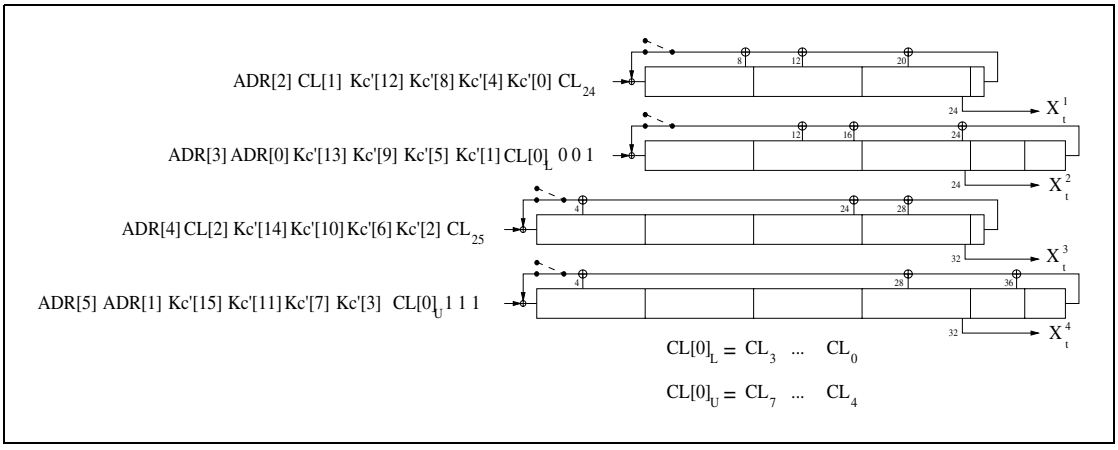


Figure 4.5: Arranging the input to the LFSRs.

In Figure 4.6, the 128 binary output symbols Z_0,..., Z_{127} are arranged in octets denoted Z[0],..., Z[15]. The LSB of Z[0] corresponds to the first of these symbols, the MSB of Z[15] is the last output from the generator. These bits shall be loaded into the LFSRs according to the figure. It is a parallel load and no update of the blend registers is done. The first output symbol is generated at the same time. The octets shall be written into the registers with the LSB in the leftmost position (i.e. the opposite of before). For example, Z_{24} is loaded into position 1 of $LFSR_4$.

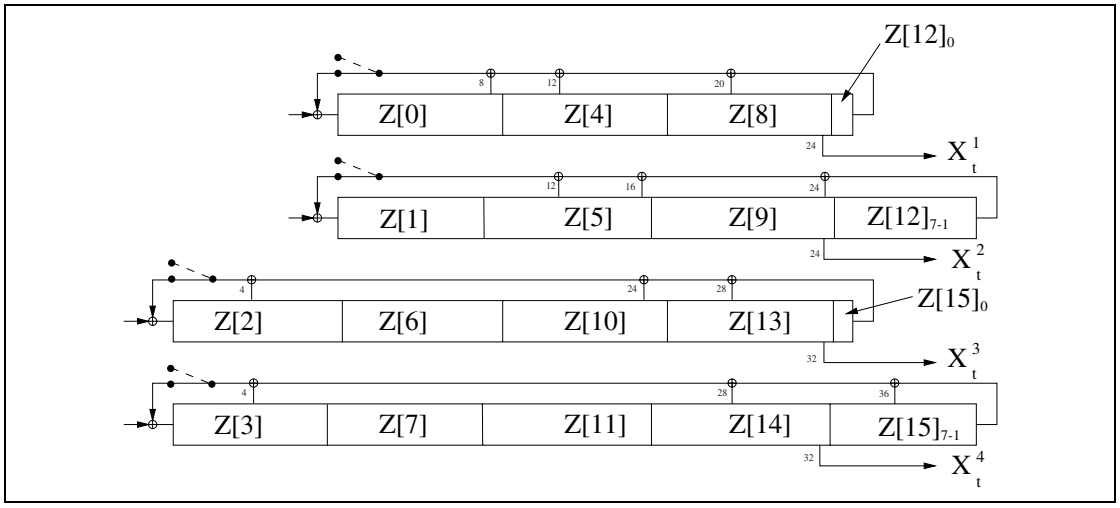


Figure 4.6: Distribution of the 128 last generated output symbols within the LFSRs.

4.6 KEY STREAM SEQUENCE

When the initialization is finished, the output from the summation combiner is used for encryption/decryption. The first bit to use shall be the one produced at the parallel load, i.e. at $t = 240$. The circuit shall be run for the entire length of the current payload. Then, before the reverse direction is started, the entire initialization process shall be repeated with updated values on the input parameters.

Sample data of the encryption output sequence can be found in [Part G] Section 1 on page 673. A necessary, but not sufficient, condition for all Bluetooth compliant implementations of encryption is to produce these encryption streams for identical initialization values.

5 AUTHENTICATION

Authentication uses a challenge-response scheme in which a claimant's knowledge of a secret key is checked through a 2-move protocol using symmetric secret keys. The latter implies that a correct claimant/verifier pair share the same secret key, for example K. In the challenge-response scheme the verifier challenges the claimant to authenticate a random input (the challenge), denoted by AU_RAND_A, with an authentication code, denoted by E_1, and return the result SRES to the verifier, see Figure 5.1 on page 797. This figure also shows that the input to E_1 consists of the tuple AU_RAND_A and the Bluetooth device address (BD_ADDR) of the claimant. The use of this address prevents a simple reflection attack[1]. The secret K shared by devices A and B is the current link key.

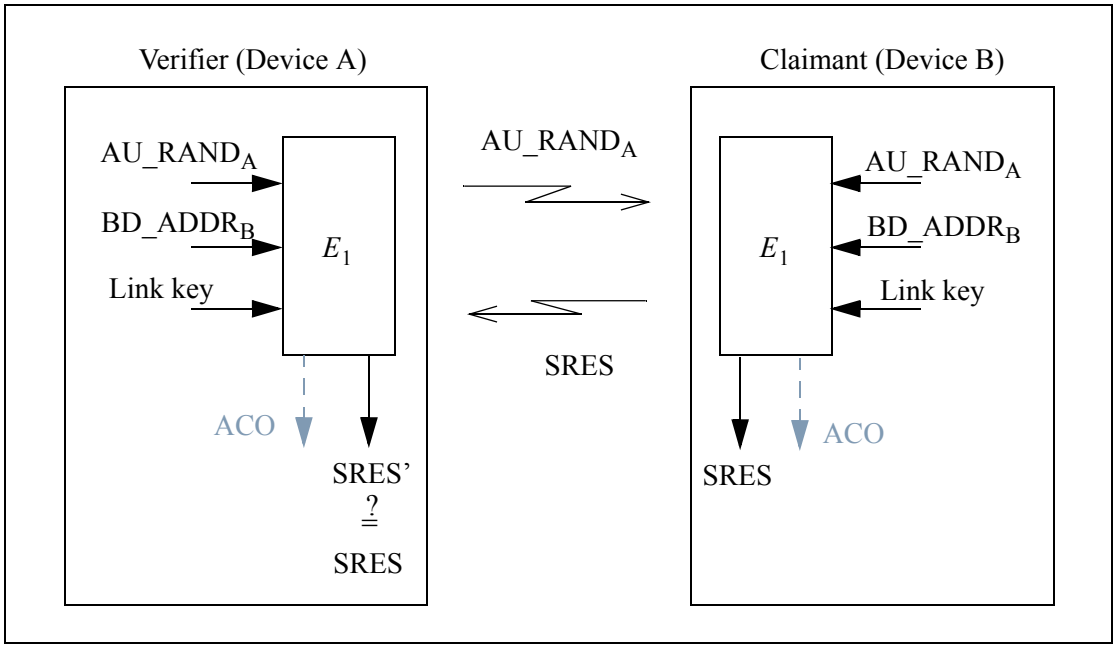


Figure 5.1: Challenge-response for the Bluetooth.

The challenge-response scheme for symmetric keys is depicted in Figure 5.2 on page 798.

1. The reflection attack actually forms no threat because all service requests are dealt with on a FIFO bases. When preemption is introduced, this attack is potentially dangerous.

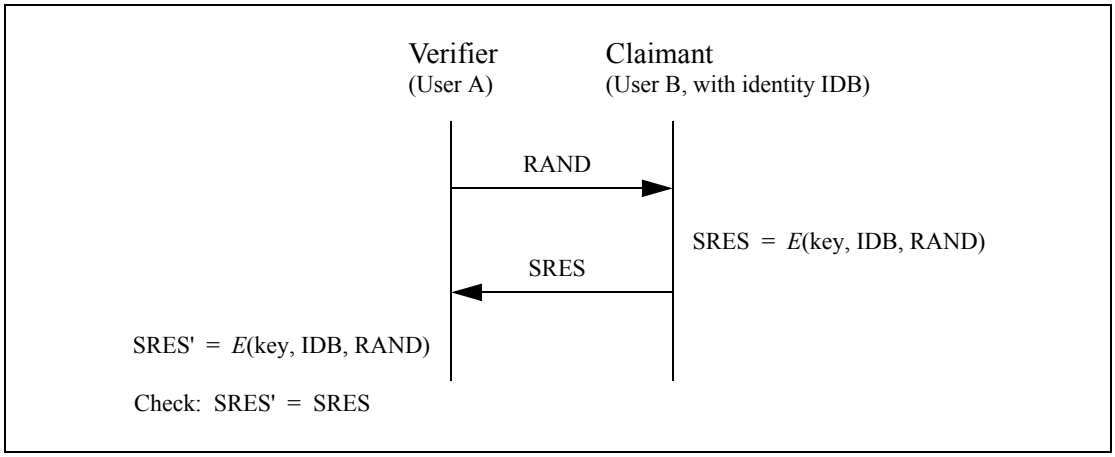


Figure 5.2: Challenge-response for symmetric key systems.

The verifier is not required to be the master. The application indicates which device has to be authenticated. Some applications only require a one-way authentication. However, some peer-to-peer communications, should use a mutual authentication in which each device is subsequently the challenger (verifier) in two authentication procedures. The LM shall process authentication preferences from the application to determine in which direction(s) the authentication(s) takes place. For mutual authentication with the devices of Figure 5.1 on page 797, after device A has successfully authenticated device B, device B could authenticate device A by sending an AU_RAND_B (different from the AU_RAND_A that device A issued) to device A, and deriving the SRES and SRES' from the new AU_RAND_B, the address of device A, and the link key K_{AB}.

If an authentication is successful the value of ACO as produced by E_1 shall be retained.

5.1 REPEATED ATTEMPTS

When the authentication attempt fails, a waiting interval shall pass before the verifier will initiate a new authentication attempt to the same claimant, or before it will respond to an authentication attempt initiated by a device claiming the same identity as the failed device. For each subsequent authentication failure with the same Bluetooth device address, the waiting interval shall be increased exponentially. That is, after each failure, the waiting interval before a new attempt can be made, could be for example, twice as long as the waiting interval prior to the previous attempt[1]. The waiting interval shall be limited to a maximum. The maximum waiting interval depends on the implementation. The waiting time shall exponentially decrease to a minimum when no new failed attempts are made during a certain time period. This procedure prevents an intruder from repeating the authentication procedure with a large number of different keys.

To make the system less vulnerable to denial-of-service attacks, the devices should keep a list of individual waiting intervals for each device it has established contact with. The size of this list may be restricted to only contain the N devices with which the most recent contacts have been made. The number N may vary for different devices depending on available memory size and user environment.

1. Another appropriate value larger than 1 may be used.

6 THE AUTHENTICATION AND KEY-GENERATING FUNCTIONS

This section describes the algorithms used for authentication and key generation.

6.1 THE AUTHENTICATION FUNCTION E_1

The authentication function E_1 is a computationally secure authentication code. E_1 uses the encryption function SAFER+. The algorithm is an enhanced version of an existing 64-bit block cipher SAFER-SK128, and it is freely available. In the following discussion, the block cipher will be denoted as the function A_r which maps using a 128-bit key, a 128-bit input to a 128-bit output, i.e.

$$A_r\colon \{0,1\}^{128} \times \{0,1\}^{128} \rightarrow \{0,1\}^{128}$$
$$(k \times x) \mapsto t. \qquad \text{(EQ 11)}$$

The details of A_r are given in the next section. The function E_1 is constructed using A_r as follows

$$E_1\colon \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{48} \rightarrow \{0,1\}^{32} \times \{0,1\}^{96}$$
$$(K, \text{RAND}, \text{address}) \mapsto (\text{SRES}, \text{ACO}), \qquad \text{(EQ 12)}$$

where $\text{SRES} = Hash(K, \text{RAND}, \text{address}, 6)[0, \ldots, 3]$, where $Hash$ is a keyed hash function defined as[1],

$$Hash\colon \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{128}$$
$$(K, I_1, I_2, L) \mapsto A'_r([\tilde{K}], [E(I_2, L) +_{16} (A_r(K, I_1) \oplus_{16} I_1)]), \qquad \text{(EQ 13)}$$

and where

$$E\colon \{0,1\}^{8 \times L} \times \{6, 12\} \rightarrow \{0,1\}^{8 \times 16}$$
$$(X[0, \ldots, L-1], L) \mapsto (X[i(\text{mod L})] \text{ for } i = 0...15), \qquad \text{(EQ 14)}$$

is an expansion of the L octet word X into a 128-bit word. The function A_r is evaluated twice for each evaluation of E_1. The key $\tilde{K}$ for the second use of A_r (actually A'_r) is offset from K as follows[2]

1. The operator $+_{16}$ denotes bytewise addition mod 256 of the 16 octets, and the operator $\oplus_{16}$ denotes bytewise XORing of the 16 octets.
2. The constants are the largest primes below 257 for which 10 is a primitive root.

$$\begin{aligned}
\tilde{K}[0] &= (K[0] + 233) \mod 256, & \tilde{K}[1] &= K[1] \oplus 229, \\
\tilde{K}[2] &= (K[2] + 223) \mod 256, & \tilde{K}[3] &= K[3] \oplus 193, \\
\tilde{K}[4] &= (K[4] + 179) \mod 256, & \tilde{K}[5] &= K[5] \oplus 167, \\
\tilde{K}[6] &= (K[6] + 149) \mod 256, & \tilde{K}[7] &= K[7] \oplus 131, \\
\tilde{K}\{8\} &= K[8] \oplus 233, & \tilde{K}[9] &= (K[9] + 229) \mod 256, \\
\tilde{K}[10] &= K[10] \oplus 223, & \tilde{K}[11] &= (K[11] + 193) \mod 256, \\
\tilde{K}[12] &= K[12] \oplus 179, & \tilde{K}[13] &= (K[13] + 167) \mod 256, \\
\tilde{K}[14] &= K[14] \oplus 149, & \tilde{K}[15] &= (K[15] + 131) \mod 256.
\end{aligned} \qquad \text{(EQ 15)}$$

A data flowchart of the computation of E_1 is shown in Figure 6.1 on page 802. E_1 is also used to deliver the parameter ACO (Authenticated Ciphering Offset) that is used in the generation of the ciphering key by E_3, see equations (EQ 3) on page 783 and (EQ 23) on page 807. The value of ACO is formed by the octets 4 through 15 of the output of the hash function defined in (EQ 13) on page 801, i.e.

$$\text{ACO} = Hash(K, \text{RAND}, \text{address}, 6)[4, \ldots, 15]. \qquad \text{(EQ 16)}$$

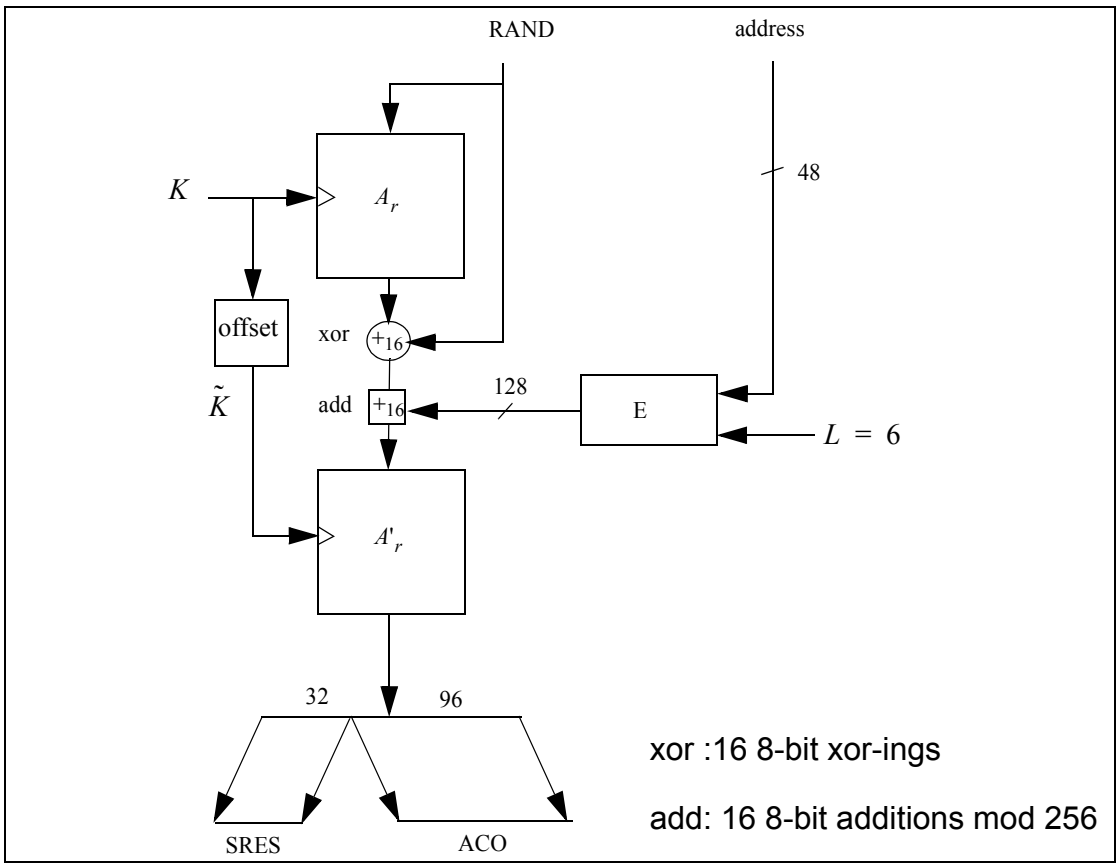


Figure 6.1: Flow of data for the computation of E_1.

6.2 THE FUNCTIONS A_r AND A'_r

The function A_r is identical to SAFER+. It consists of a set of 8 layers, (each layer is called a round) and a parallel mechanism for generating the sub keys $K_p[j]$, $p = 1, 2, \ldots, 17$, which are the round keys to be used in each round. The function will produce a 128-bit result from a 128-bit random input string and a 128-bit key. Besides the function A_r, a slightly modified version referred to as A'_r is used in which the input of round 1 is added to the input of round 3. This is done to make the modified version non-invertible and prevents the use of A'_r (especially in E_{2x}) as an encryption function. See Figure 6.2 on page 804 for details.

6.2.1 The round computations

The computations in each round are a composition of encryption with a round key, substitution, encryption with the next round key, and, finally, a Pseudo Hadamard Transform (PHT). The computations in a round shall be as shown in Figure 6.2 on page 804. The sub keys for round $r, r = 1, 2, \ldots, 8$ are denoted $K_{2r-1}[j]$, $K_{2r}[j]$, $j = 0, 1, \ldots, 15$. After the last round $K_{17}[j]$ is applied identically to all previous odd numbered keys.

6.2.2 The substitution boxes “e” and “l”

In Figure 6.2 on page 804 two boxes are shown, marked “e” and “l”. These boxes implement the same substitutions as are used in SAFER+; i.e. they implement

$$\begin{aligned} e, l &: \{0, \ldots, 255\} \rightarrow \{0, \ldots, 255\}, \\ e &: i \mapsto (45^i \pmod{257}) \pmod{256}, \\ l &: i \mapsto j \text{ s.t. } i = e(j). \end{aligned}$$

Their role, as in the SAFER+ algorithm, is to introduce non-linearity.

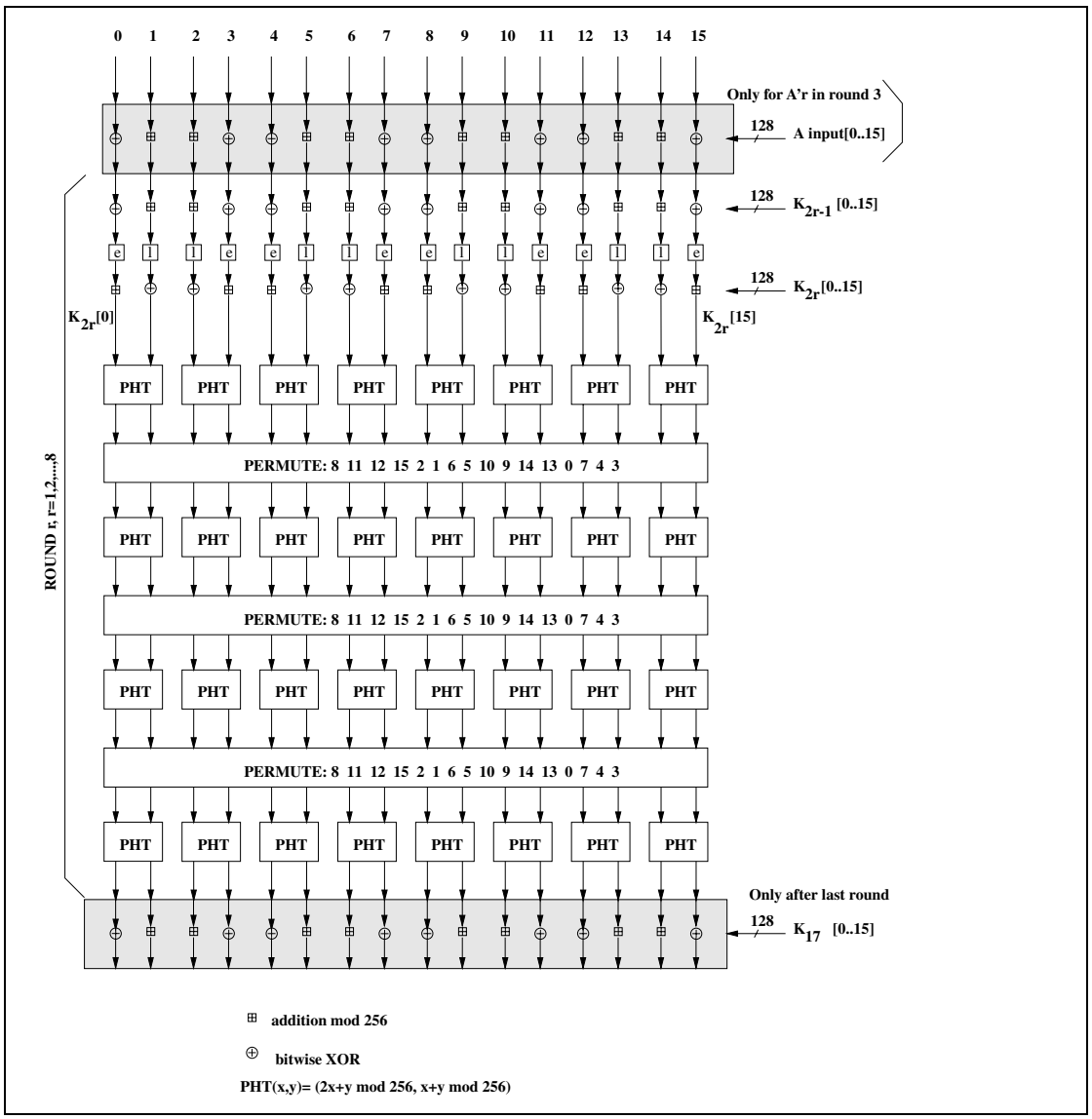


Figure 6.2: One round in A_r and A'_r. The permutation boxes show how input byte indices are mapped onto output byte indices. Thus, position 8 is mapped on position 0 (leftmost), position 11 is mapped on position 1, etc.

6.2.3 Key scheduling

In each round, 2 batches of 16 octet-wide keys are needed. These round keys are derived as specified by the key scheduling in SAFER+. Figure 6.3 on page 805 gives an overview of how the round keys $K_p[j]$ are determined. The bias vectors B_2, B_3, ..., B_{17} shall be computed according to following equation:

$$B_p[i] = ((45^{(45^{17p+i+1} \bmod 257)} \bmod 257) \bmod 256), \text{ for } i = 0, \ldots, 15. \qquad \text{(EQ 17)}$$

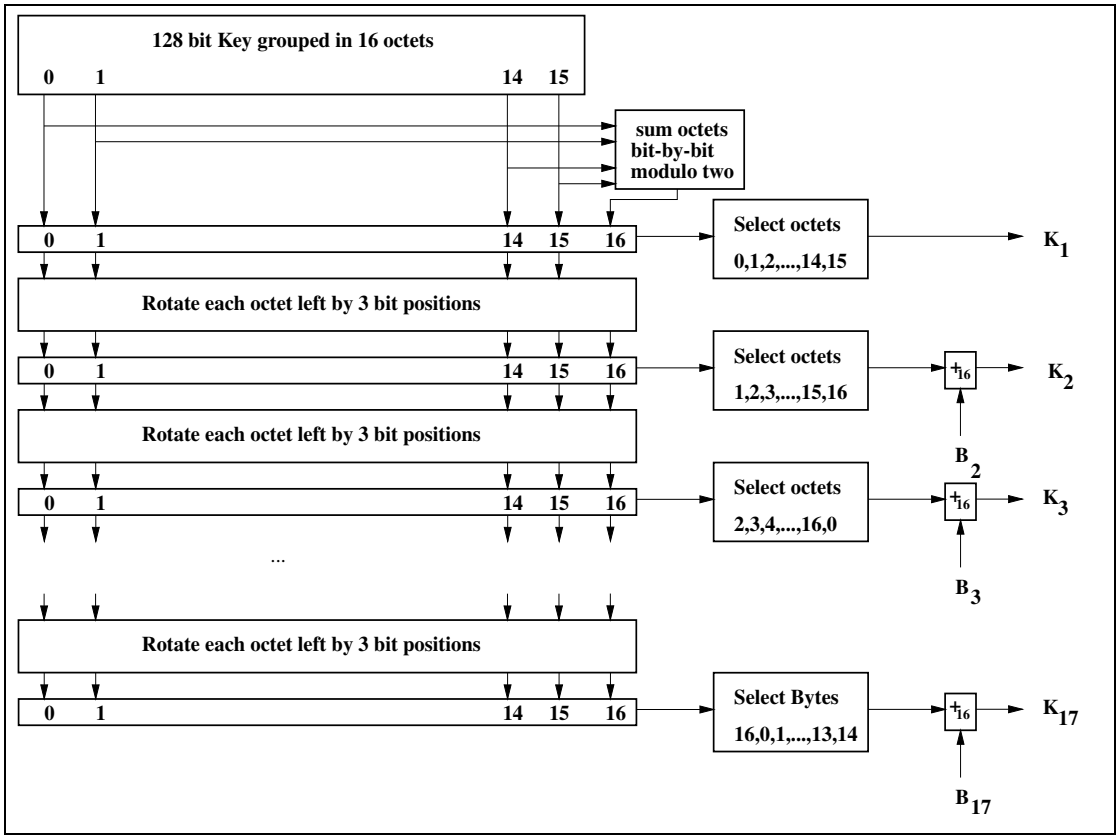


Figure 6.3: Key scheduling in A_r.

6.3 E_2-KEY GENERATION FUNCTION FOR AUTHENTICATION

The key used for authentication shall be derived through the procedure that is shown in Figure 6.4 on page 807. The figure shows two modes of operation for the algorithm. In the first mode, E_{21} produces a 128-bit kink key, K, using a 128-bit RAND value and a 48-bit address. This mode shall be utilized when creating unit keys and combination keys. In the second mode, E_{22} produces a 128-bit link key, K, using a 128-bit RAND value and an L octet user PIN. The second mode shall be used to create the initialization key, and also when a master key is to be generated.

When the initialization key is generated, the PIN is augmented with the BD_ADDR, see Section 3.2.1 on page 780 for which address to use. The augmentation shall always start with the least significant octet of the address immediately following the most significant octet of the PIN. Since the maximum length of the PIN used in the algorithm cannot exceed 16 octets, it is possible that not all octets of BD_ADDR will be used.

This key generating algorithm again exploits the cryptographic function. E_2 for mode 1 (denoted E_{21}) is computed according to following equations:

$$\begin{aligned} E_{21}\colon\ \{0,1\}^{128} \times \{0,1\}^{48} &\rightarrow \{0,1\}^{128} \\ (\mathrm{RAND}, \mathrm{address}) &\mapsto A'_r(X, Y) \end{aligned} \qquad \text{(EQ 18)}$$

where (for mode 1)

$$\begin{cases} X = \mathrm{RAND}[0 \ldots 14] \cup (\mathrm{RAND}[15] \oplus 6) \\ Y = \bigcup_{i=0}^{15} \mathrm{address}[i \pmod 6] \end{cases} \qquad \text{(EQ 19)}$$

Let L be the number of octets in the user PIN. The augmenting is defined by

$$\mathrm{PIN}' = \begin{cases} \mathrm{PIN}[0 \ldots L-1] \cup \mathrm{BD\_ADDR}[0 \ldots \min\{5, 15-L\}], & L < 16, \\ \mathrm{PIN}[0 \ldots L-1], & L = 16, \end{cases} \qquad \text{(EQ 20)}$$

Then, in mode 2, E_2 (denoted E_{22}) is

$$\begin{aligned} E_{22}\colon\ \{0,1\}^{8L'} \times \{0,1\}^{128} \times \{1, 2, \ldots, 16\} &\rightarrow \{0,1\}^{128} \\ (\mathrm{PIN}', \mathrm{RAND}, \mathrm{L}') &\mapsto A'_r(X, Y) \end{aligned} \qquad \text{(EQ 21)}$$

where

$$\begin{cases} X = \bigcup_{i=0}^{15} \mathrm{PIN}'[i \pmod{L'}], \\ Y = \mathrm{RAND}[0 \ldots 14] \cup (\mathrm{RAND}[15] \oplus L'), \end{cases} \qquad \text{(EQ 22)}$$

and $L' = \min\{16, L+6\}$ is the number of octets in PIN'.

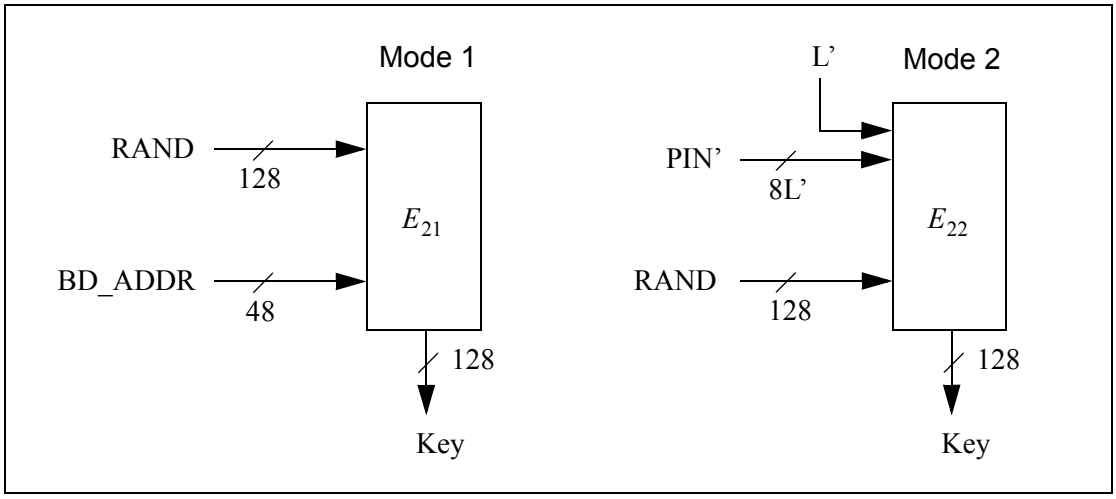


Figure 6.4: Key generating algorithm E_2 and its two modes. Mode 1 is used for unit and combination keys, while mode 2 is used for K_{init} and K_{master}.

6.4 E_3-KEY GENERATION FUNCTION FOR ENCRYPTION

The ciphering key K_C used by E_0 shall be generated by E_3. The function E_3 is constructed using A'_r as follows

$$E_3: \{0,1\}^{128} \times \{0,1\}^{128} \times \{0,1\}^{96} \rightarrow \{0,1\}^{128}$$
$$(K, \text{RAND}, \text{COF}) \mapsto Hash(K, \text{RAND}, \text{COF}, 12) \qquad \text{(EQ 23)}$$

where $Hash$ is the hash function as defined by (EQ 13) on page 801. The key length produced is 128 bits. However, before use within E_0, the encryption key K_C is shortened to the correct encryption key length, as described in Section 4.5 on page 793. A block scheme of E_3 is depicted in Figure 6.5.

The value of COF is determined as specified by equation (EQ 3) on page 783.

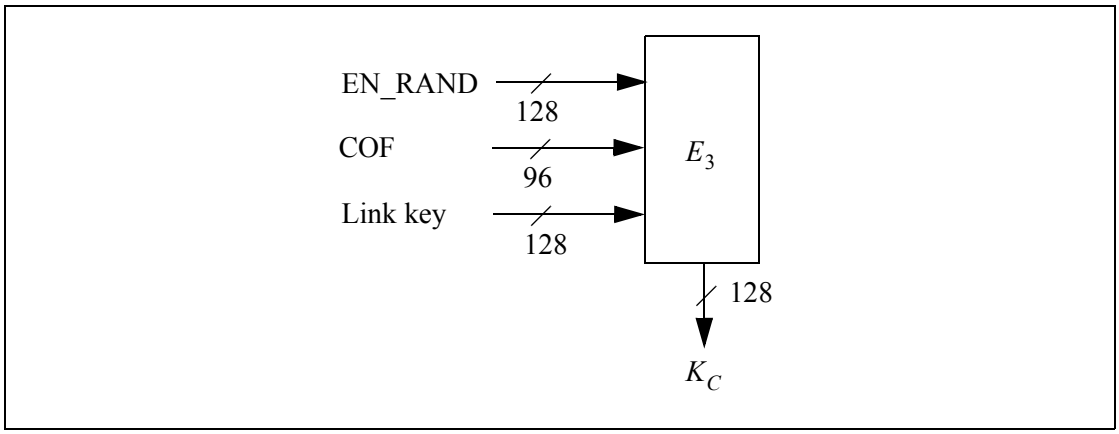


Figure 6.5: Generation of the encryption key.

7 LIST OF FIGURES

8 LIST OF TABLES

Specification of the Bluetooth System

Wireless connections made easy

Core System Package

[Host volume]

Covered Core Package version:
2.0 + EDR
Current Master TOC issued:
4 November 2004

Bluetooth®

Revision History

The Revision History is shown in the “Appendix” on page 51[vol. 0].

Contributors

The persons who contributed to this specification are listed in the “Appendix” on page 51[vol. 0].

Web Site

This specification can also be found on the official Bluetooth web site:
http://www.bluetooth.com

Disclaimer and Copyright Notice

Part A

Logical Link Control and Adaptation Protocol

Core System Package [Host volume]

Part A

LOGICAL LINK CONTROL AND ADAPTATION PROTOCOL SPECIFICATION

The Bluetooth logical link control and adaptation protocol (L2CAP) supports higher level protocol multiplexing, packet segmentation and reassembly, and the conveying of quality of service information. The protocol state machine, packet format and composition are described in this document.

CONTENTS

1 INTRODUCTION

This section of the Bluetooth Specification defines the Logical Link Control and Adaptation Layer Protocol, referred to as L2CAP. L2CAP is layered over the Link Controller Protocol and resides in the data link layer as shown in Figure 1.1. L2CAP provides connection-oriented and connectionless data services to upper layer protocols with protocol multiplexing capability, segmentation and reassembly operation, and group abstractions. L2CAP permits higher level protocols and applications to transmit and receive upper layer data packets (L2CAP Service Data Units, SDU) up to 64 kilobytes in length. L2CAP also permits per-channel flow control and retransmission via the Flow Control and Retransmission Modes.

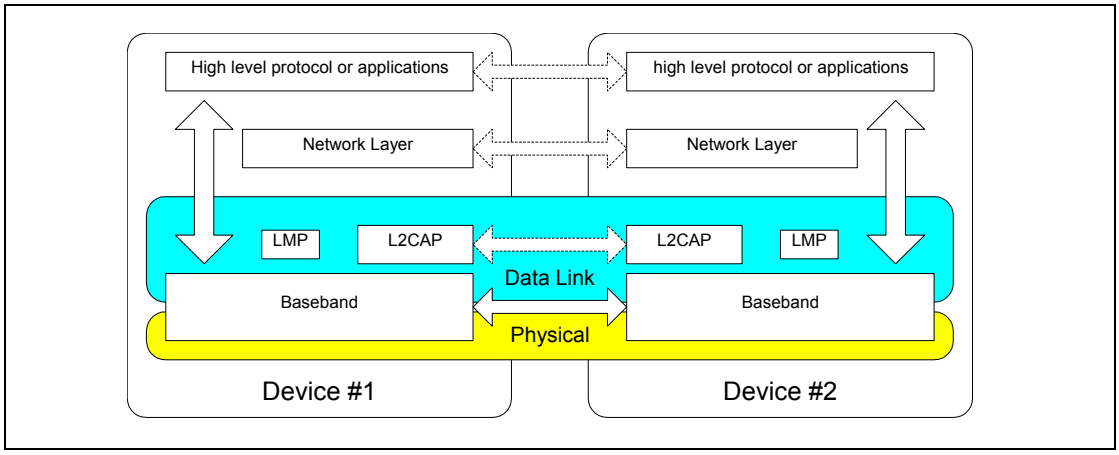


Figure 1.1: L2CAP within protocol layers

The L2CAP layer provides logical channels, named L2CAP channels, which are mapped to L2CAP logical links supported by an ACL logical transport, see baseband specification [vol.3, part B] Section 4.4 on page 98.

1.1 L2CAP FEATURES

The functional requirements for L2CAP include protocol/channel multiplexing, segmentation and reassembly (SAR), per-channel flow control, error control and group management. Figure 1.2 on page 22 illustrates how L2CAP data flows fit into the Bluetooth Protocol Stack. L2CAP lies above the Link Controller Protocol and interfaces with other communication protocols such as the Bluetooth Service Discovery Protocol (SDP), RFCOMM, Telephony Control (TCS) and Bluetooth Network Encapsulation Protocol (BNEP). Voice-quality channels for audio and telephony applications and synchronous transparent connections are usually run over synchronous logical transports, see [vol.3, part B] Section 4.3 on page 98. Packetized audio data, such as IP Telephony, may be sent using communication protocols running over L2CAP.

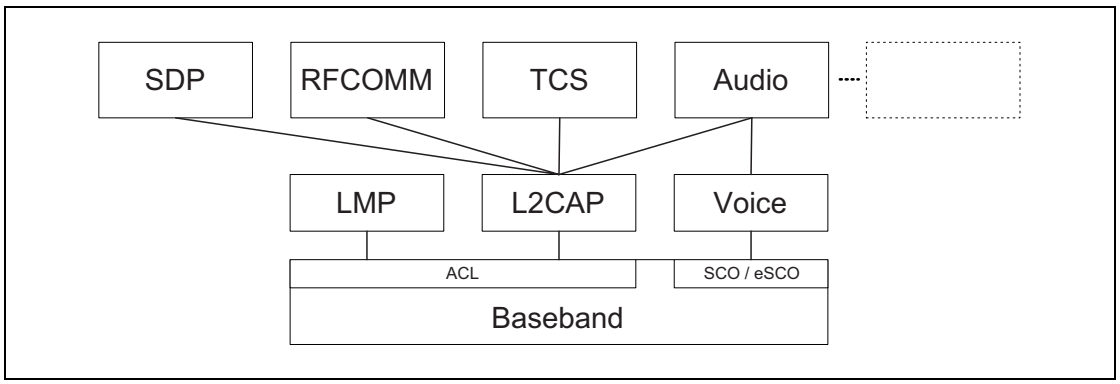


Figure 1.2: L2CAP data flows in Bluetooth Protocol Architecture

Figure 1.3 on page 22 breaks down L2CAP into its architectural components. The Channel Manager provides the control plane functionality and is responsible for all internal signalling, L2CAP peer-to-peer signalling and signalling with higher and lower layers. It performs the state machine functionality described in Section 6 on page 67 and uses message formats described in Section 4 on page 41, Section 5 on page 57. The Retransmission and Flow Control block provides per-channel flow control and optional retransmission for applications that require it. The Resource Manager is responsible for providing a frame relay service to the Channel Manager, the Retransmission and Flow Control block and those application data streams that do not require Retransmission and Flow Control services. It is responsible for coordinating the transmission and reception of packets related to multiple L2CAP channels over the facilities offered at the lower layer interface.

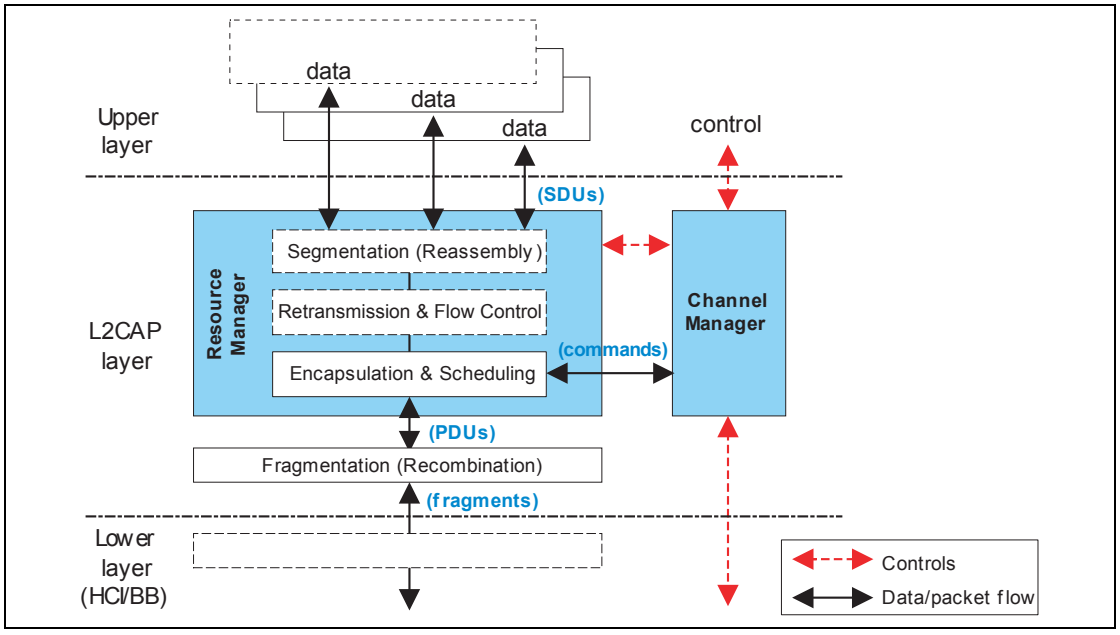


Figure 1.3: L2CAP architectural blocks

- *Protocol/channel multiplexing*

  L2CAP supports multiplexing because the Baseband Protocol does not support any 'type' field identifying the higher layer protocol being multiplexed above it.

  During channel setup, protocol multiplexing capability is used to route the connection request to the correct upper layer protocol.

  For data transfer, logical channel multiplexing is needed to distinguish between multiple upper layer entities. There may be more than one upper layer entity using the same protocol.

- *Segmentation and reassembly*

  With the frame relay service offered by the Resource Manager, the length of transport frames is controlled by the individual applications running over L2CAP. Many multiplexed applications are better served if L2CAP has control over the PDU length. This provides the following benefits:

  a) Segmentation will allow the interleaving of application data units in order to satisfy latency requirements.

  b) Memory and buffer management is easier when L2CAP controls the packet size.

  c) Error correction by retransmission can be made more efficient.

  d) The amount of data that is destroyed when an L2CAP PDU is corrupted or lost can be made smaller than the application's data unit.

  e) The application is decoupled from the segmentation required to map the application packets into the lower layer packets.

- *Flow control per L2CAP channel*

  When several data streams run over the same L2CAP logical link, using separate L2CAP channels, each channel may require individual flow control. Also L2CAP provides flow control services to profiles or applications that need flow control and can avoid having to implement it. Due to the delays between the L2CAP layers, stop-and-go flow control as employed in the baseband is not sufficient. A window based flow control scheme is provided. The use of flow control is an optional aspect of the L2CAP protocol.

- *Error control and retransmissions*

  Some applications require a residual error rate much smaller than the baseband can deliver. L2CAP includes optional error checks and retransmissions of L2CAP PDUs. The error checking in L2CAP protects against errors due to the baseband falsely accepting packet headers and due to failures of the HEC or CRC error checks on the baseband packets. Retransmission Mode also protects against loss of packets due to flush on the same logical transport. The error control works in conjunction with flow control in the sense that the flow control mechanism will throttle retransmissions as well as first transmissions. The use of error control and retransmission procedures is optional.

- *Fragmentation and Recombination*

  The lower layers have limited transmission capabilities and may require fragment sizes different from those created by L2CAP segmentation. Therefore layers below L2CAP may further fragment and recombine L2CAP PDUs to create fragments which fit each layer capabilities. During transmission of an L2CAP PDU, many different levels of fragmentation and recombination may occur in both peer devices.

  The HCI driver or controller may fragment L2CAP PDUs to honor packet size constraints of a host controller interface transport scheme. This results in HCI data packet payloads carrying start and continuation fragments of the L2CAP PDU. Similarly the link controller may fragment L2CAP PDUs to map them into baseband packets. This results in baseband packet payloads carrying start and continuation fragments of the L2CAP PDU.

  Each layer of the protocol stack may pass on different sized fragments of L2CAP PDUs, and the size of fragments created by a layer may be different in each peer device. However the PDU is fragmented within the stack, the receiving L2CAP entity still recombines the fragments to obtain the original L2CAP PDU.

- *Quality of Service*

  The L2CAP connection establishment process allows the exchange of information regarding the quality of service (QoS) expected between two Bluetooth devices. Each L2CAP implementation monitors the resources used by the protocol and ensures that QoS contracts are honored.

1.2 ASSUMPTIONS

The protocol is designed based on the following assumptions:

1. The ACL logical transport and L2CAP logical link between two devices is set up using the Link Manager Protocol. The baseband provides orderly delivery of data packets, although there might be individual packet corruption and duplicates. No more than 1 unicast ACL logical transport exists between any two devices.
2. The baseband always provides the impression of full-duplex communication channels. This does not imply that all L2CAP communications are bi-directional. Multicasts and unidirectional traffic (e.g., video) do not require duplex channels.
3. The L2CAP layer provides a channel with a degree of reliability based on the mechanisms available at the baseband layer and with optional additional packet segmentation and error detection that can be enabled in the enhanced L2CAP layer. The baseband performs data integrity checks and resends data until it has been successfully acknowledged or a timeout occurs. Because acknowledgements may be lost, timeouts may occur even after the data has been successfully sent. The link controller protocol uses a 1-bit sequence number. Note that the use of baseband broadcast packets is prohibited if reliability is required, since all broadcasts start the first segment of an L2CAP packet with the same sequence bit.
4. Some applications will expect independent flow control, independence from the effects of other traffic and, in some cases, better error control than the baseband provides. The Flow and Error Control block provides two modes. Retransmission Mode offers segmentation, flow control and L2CAP PDU retransmissions. Flow control mode offers just the segmentation and flow control functions. If Basic L2CAP mode is chosen, the Flow and Error Control block is not used.

1.3 SCOPE

The following features are outside the scope of L2CAP's responsibilities:

- L2CAP does not transport audio or transparent synchronous data designated for SCO or eSCO logical transports.
- L2CAP does not support a reliable multicast channel. See Section 3.2 on page 34.
- L2CAP does not support the concept of a global group name.

1.4 TERMINOLOGY

The following formal definitions apply:

Term	Description
Upper layer	The system layer above the L2CAP layer, which exchanges data with L2CAP in the form of SDUs. The upper layer may be represented by an application or higher protocol entity known as the Service Level Protocol. The interface of the L2CAP layer with the upper layer is not specified.
Lower layer	The system layer below the L2CAP layer, which exchanges data with the L2CAP layer in the form of PDUs, or fragments of PDUs. The lower layer is mainly represented within the Bluetooth Controller, however a Host Controller Interface (HCI) may be involved, such that an HCI host driver could also be seen as the lower layer. Except for the HCI functional specification (in case HCI is involved) the interface between L2CAP and the lower layer is not specified.
L2CAP channel	The logical connection between two endpoints in peer devices, characterized by their Channel Identifiers (CID), which is multiplexed on the L2CAP logical link, which is supported by an ACL logical transport, see [vol.3, part B] Section 4.4 on page 98
SDU, or L2CAP SDU	Service Data Unit: a packet of data that L2CAP exchanges with the upper layer and transports transparently over an L2CAP channel using the procedures specified here. The term SDU is associated with data originating from upper layer entities only, i.e. does not include any protocol information generated by L2CAP procedures.
Segment, or SDU segment	A part of an SDU, as resulting from the Segmentation procedure. An SDU may be split into one or more segments. Note: this term is relevant only to the Retransmission Mode and Flow Control Mode, not to the Basic L2CAP Mode.
Segmentation	A procedure used in the L2CAP Retransmission and Flow Control Modes, resulting in an SDU being split into one or more smaller units, called Segments, as appropriate for the transport over an L2CAP channel. Note: this term is relevant only to the Retransmission Mode and Flow Control Mode, not to the Basic L2CAP Mode.
Reassembly	The reverse procedure corresponding to Segmentation, resulting in an SDU being re-established from the segments received over an L2CAP channel, for use by the upper layer. Note that the interface between the L2CAP and the upper layer is not specified; therefore, reassembly may actually occur within an upper layer entity although it is conceptually part of the L2CAP layer. Note: this term is relevant only to the Retransmission Mode and Flow Control Mode, not to the Basic L2CAP Mode.
PDU, or L2CAP PDU	Protocol Data Unit a packet of data containing L2CAP protocol information fields, control information, and/or upper layer information data. A PDU is always started by a Basic L2CAP header. Types of PDUs are: B-frames, I-frames, S-frames, C-frames and G-frames.

Table 1.1: Terminology

Term	Description
Basic L2CAP header	Minimum L2CAP protocol information that is present in the beginning of each PDU: a length field and a field containing the Channel Identifier (CID)
Basic information frame (B-frame)	A B-frame is a PDU used in the Basic L2CAP mode for L2CAP data packets. It contains a complete SDU as its payload, encapsulated by a basic L2CAP header.
Information frame (I-frame)	An I-frame is a PDU used in the Retransmission Mode and the Flow Control Mode. It contains an SDU segment and additional protocol information, encapsulated by a basic L2CAP header
Supervisory frame (S-frame)	An S-frame is a PDU used in the Retransmission Mode and the Flow Control Mode. It contains protocol information only, encapsulated by a basic L2CAP header, and no SDU data.
Control frame (C-frame)	A C-frame is a PDU that contains L2CAP signalling messages exchanged between the peer L2CAP entities. C-frames are exclusively used on the L2CAP signalling channel.
Group frame (G-frame)	G-frame is a PDU exclusively used on Connectionless L2CAP channels in the Basic L2CAP mode. It contains a complete SDU as its payload, encapsulated by a specific header.
Fragment	A part of a PDU, as resulting from a fragmentation operation. Fragments are used only in the delivery of data to and from the lower layer. They are not used for peer-to-peer transportation. A fragment may be a Start or Continuation Fragment with respect to the L2CAP PDU. A fragment does not contain any protocol information beyond the PDU; the distinction of start and continuation fragments is transported by lower layer protocol provisions. Note: Start Fragments always begin with the Basic L2CAP header of a PDU.
Fragmentation	A procedure used to split L2CAP PDUs to smaller parts, named fragments, appropriate for delivery to the lower layer transport. Although described within the L2CAP layer, fragmentation may actually occur in an HCI host driver, and/or in the Controller, to accommodate the L2CAP PDU transport to HCI data packet or baseband packet sizes. Fragmentation of PDUs may be applied in all L2CAP modes. Note: in version 1.1, Fragmentation and Recombination was referred to as “Segmentation and Reassembly“.
Recombination	The reverse procedure corresponding to fragmentation, resulting in an L2CAP PDU re-established from fragments. In the receive path, full or partial recombination operations may occur in the Controller and/or the Host, and the location of recombination does not necessarily correspond to where fragmentations occurs on the transmit side.
Maximum Transmission Unit (MTU)	The maximum size of payload data, in octets, that the upper layer entity is capable of accepting, i.e. the MTU corresponds to the maximum SDU size.

Table 1.1: Terminology

Term	Description
Maximum PDU payload Size (MPS)	The maximum size of payload data in octets that the L2CAP layer entity is capable of accepting, i.e. the MPS corresponds to the maximum PDU payload size. Note: in the absence of segmentation, or in the Basic L2CAP Mode, the Maximum Transmission Unit is the equivalent to the Maximum PDU payload Size and shall be made equal in the configuration parameters.
Signalling MTU (MTU_{sig})	The maximum size of command information that the L2CAP layer entity is capable of accepting. The MTU_{sig}, refers to the signalling channel only and corresponds to the maximum size of a C-frame, excluding the Basic L2CAP header. The MTU_{sig} value of a peer is discovered when a C-frame that is too large is rejected by the peer.
Connectionless MTU (MTU_{cnl})	The maximum size of the connection packet information that the L2CAP layer entity is capable of accepting. The MTU_{cnl} refers to the connectionless channel only and corresponds to the maximum G-frame, excluding the Basic L2CAP header. The MTU_{cnl} of a peer can be discovered by sending an Information Request.
MaxTransmit	In Retransmission mode, MaxTransmit controls the number of transmissions of a PDU that L2CAP is allowed to try before assuming that the PDU (and the link) is lost. The minimum value is 1 (only 1 transmission permitted). Note: Setting MaxTransmit to 1 prohibits PDU retransmissions. Failure of a single PDU will cause the link to drop. By comparison, in Flow Control mode, failure of a single PDU will not necessarily cause the link to drop.

Table 1.1: Terminology

2 GENERAL OPERATION

L2CAP is based around the concept of *'channels'*. Each one of the endpoints of an L2CAP channel is referred to by a *channel identifier (CID).*

2.1 CHANNEL IDENTIFIERS

A channel identifier (CID) is the local name representing a logical channel endpoint on the device. The null identifier (0x0000) is an illegal identifier and shall never be used as a destination endpoint. Identifiers from 0x0001 to 0x003F are reserved for specific L2CAP functions. Implementations are free to manage the remaining CIDs in a manner best suited for that particular implementation, with the provision that two simultaneously active L2CAP channels shall not share the same CID. Table 2.1 on page 29 summarizes the definition and partitioning of the CID name space.

CID assignment is relative to a particular device and a device can assign CIDs independently from other devices (unless it needs to use any of the reserved CIDs shown in the table below). Thus, even if the same CID value has been assigned to (remote) channel endpoints by several remote devices connected to a single local device, the local device can still uniquely associate each remote CID with a different device.

CID	Description
0x0000	Null identifier
0x0001	Signalling channel
0x0002	Connectionless reception channel
0x0003-0x003F	Reserved
0x0040-0xFFFF	Dynamically allocated

Table 2.1: CID name space

2.2 OPERATION BETWEEN DEVICES

Figure 2.1 on page 30 illustrates the use of CIDs in a communication between corresponding peer L2CAP entities in separate devices. The connection-oriented data channels represent a connection between two devices, where a CID identifies each endpoint of the channel. The connectionless channels restrict data flow to a single direction. These channels are used to support a channel 'group' where the CID on the source represents one or more remote devices. There are also a number of CIDs reserved for special purposes. The signalling channel is one example of a reserved channel. This channel is used to create and establish connection-oriented data channels and to negotiate changes in the characteristics of connection oriented and connectionless channels. Support for a signalling channel within an L2CAP entity is mandatory.

Note: it is assumed that an L2CAP signalling channel is available immediately when an ACL logical transport is established between two devices, and L2CAP traffic is enabled on the L2CAP logical link.
Another CID is reserved for all incoming connectionless data traffic. In the example below, a CID is used to represent a group consisting of device #3 and #4. Traffic sent from this channel ID is directed to the remote channel reserved for connectionless data traffic.

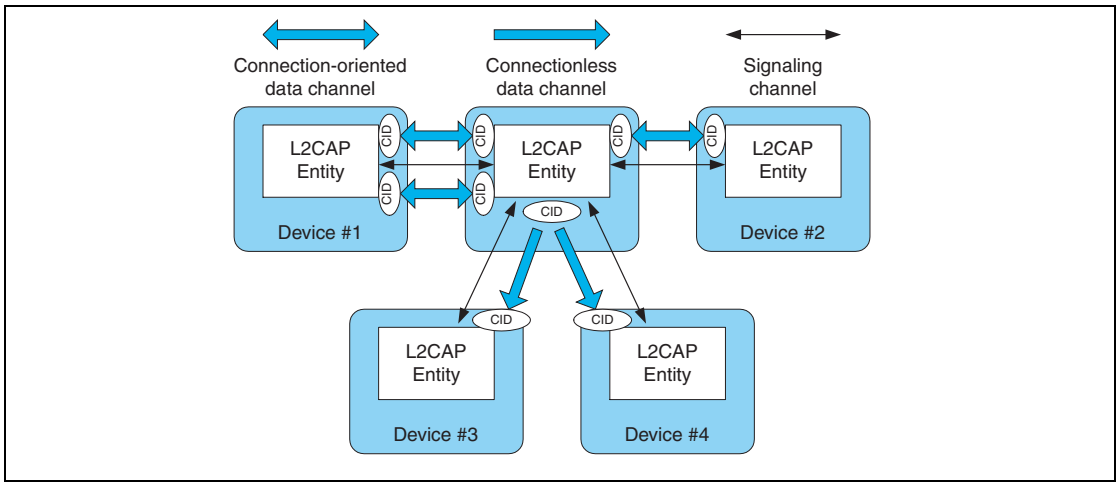


Figure 2.1: Channels between devices

Table 2.2 on page 30 describes the various channels and their source and destination identifiers. A CID is allocated to identify the local endpoint and shall be in the range 0x0040 to 0xFFFF. Section 6 on page 67 describes the state machine associated with each connection-oriented channel. Section 3.1 on page 33 describes the packet format associated with bi-directional channels and Section 3.2 on page 34 describes the packet format associated with uni-directional channels.

Channel Type	Local CID (sending)	Remote CID (receiving)
Connection-oriented	Dynamically allocated	Dynamically allocated
Connectionless data	Dynamically allocated	0x0002 (fixed)
Signalling	0x0001 (fixed)	0x0001 (fixed)

Table 2.2: Types of Channel Identifiers

2.3 OPERATION BETWEEN LAYERS

L2CAP implementations should follow the general architecture described below. L2CAP implementations transfer data between upper layer protocols and the lower layer protocol. This document lists a number of services that should be exported by any L2CAP implementation. Each implementation shall also support a set of signalling commands for use between L2CAP implementations. L2CAP implementations should also be prepared to accept certain types of events from lower layers and generate events to upper layers. How these events are passed between layers is implementation specific.

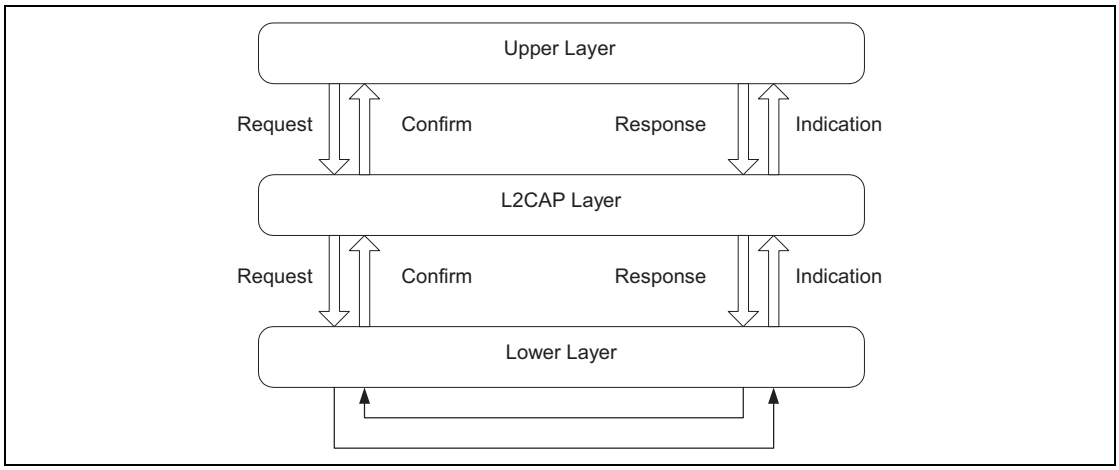


Figure 2.2: L2CAP transaction model.

2.4 MODES OF OPERATION

L2CAP may operate in one of three different modes as selected for each L2CAP channel by an upper layer.

The modes are:

- Basic L2CAP Mode (equivalent to L2CAP specification in Bluetooth v1.1) [1]
- Flow Control Mode
- Retransmission Mode

The modes are enabled using the configuration procedure. The Basic L2CAP Mode is the default mode, which is used when no other mode is agreed.

In Flow Control and Retransmission modes, PDUs exchanged with a peer entity are numbered and acknowledged. The sequence numbers in the PDUs are used to control buffering, and a TxWindow size is used to limit the required buffer space and/or provide a method for flow control. In addition to the window size, the Token Bucket size parameter of the flow specification can be used to

1. Specification of the Bluetooth System v1.1 (Feb 22nd 2001): volume 1, part D.

dimension the buffers; in particular on channels that do not use flow and error control.

In Flow Control Mode no retransmissions take place, but missing PDUs are detected and can be reported as lost.

In Retransmission Mode a timer is used to ensure that all PDUs are delivered to the peer, by retransmitting PDUs as needed. A go-back-n repeat mechanism is used to simplify the protocol and limit the buffering requirements.

3 DATA PACKET FORMAT

L2CAP is packet-based but follows a communication model based on *channels*. A channel represents a data flow between L2CAP entities in remote devices. Channels may be connection-oriented or connectionless.
All packet fields shall use Little Endian byte order.

3.1 CONNECTION-ORIENTED CHANNEL IN BASIC L2CAP MODE

Figure 3.1 on page 33 illustrates the format of the L2CAP PDU within a connection-oriented channel. In basic L2CAP mode, the L2CAP PDU on a connection-oriented channel is also referred to as a "B-frame".

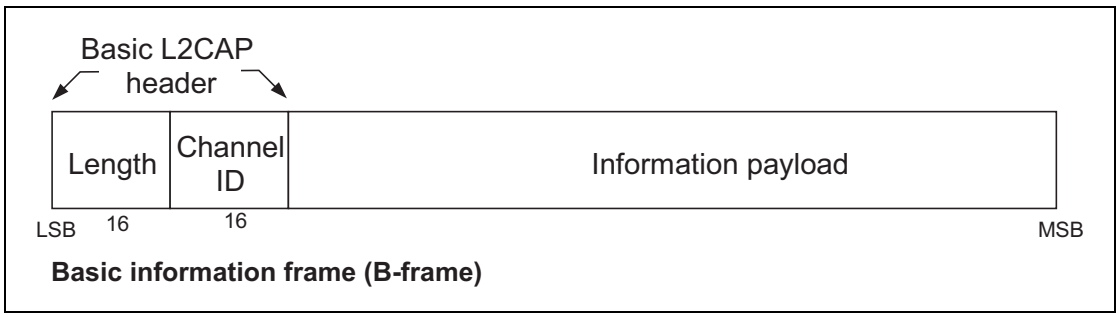


Figure 3.1: PDU format in Basic L2CAP mode on connection-oriented channels (field sizes in bits)

The fields shown are:

- *Length: 2 octets (16 bits)*

  Length indicates the size of the information payload in octets, excluding the length of the L2CAP header. The length of an information payload can be up to 65535 octets. The Length field is used for recombination and serves as a simple integrity check of the recombined L2CAP packet on the receiving end.

- *Channel ID: 2 octets*

  The channel ID (CID) identifies the destination channel endpoint of the packet.

- *Information payload: 0 to 65535 octets*

  This contains the payload received from the upper layer protocol (outgoing packet), or delivered to the upper layer protocol (incoming packet). The MTU is determined during channel configuration (see Section 5.1 on page 57). The minimum supported MTU for the signalling PDUs (MTU_{sig}) is 48 octets (see Section 4 on page 41).

3.2 CONNECTIONLESS DATA CHANNEL IN BASIC L2CAP MODE

Figure 3.2 illustrates the L2CAP PDU format within a connectionless data channel. Here, the L2CAP PDU is also referred to as a "G-frame".

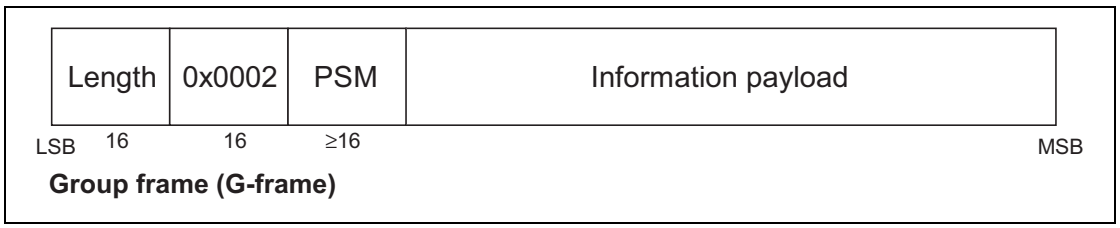


Figure 3.2: L2CAP PDU format in Basic L2CAP mode on Connectionless channel

The fields shown are:

- *Length: 2 octets*

  Length indicates the size of information payload plus the PSM field in octets.

- *Channel ID: 2 octets*

  Channel ID (0x0002) reserved for connectionless traffic.

- *Protocol/Service Multiplexer (PSM): 2 octets (minimum)*

  For information on the PSM field see Section 4.2 on page 44.

- *Information payload: 0 to 65533 octets*

  The payload information to be distributed to all members of the piconet. Implementations shall support a connectionless MTU (MTU_{cnl}) of 48 octets on connectionless channels. Devices may also explicitly change to a larger or smaller connectionless MTU (MTU_{cnl}).

  Note: the maximum size of the Information payload field decreases accordingly if the PSM field is extended beyond the two octet minimum.

3.3 CONNECTION-ORIENTED CHANNEL IN RETRANSMISSION/FLOW CONTROL MODES

To support flow control and retransmissions, L2CAP PDU types with protocol elements in addition to the Basic L2CAP header are defined. The information frames (I-frames) are used for information transfer between L2CAP entities. The supervisory frames (S-frames) are used to acknowledge I-frames and request retransmission of I-frames.

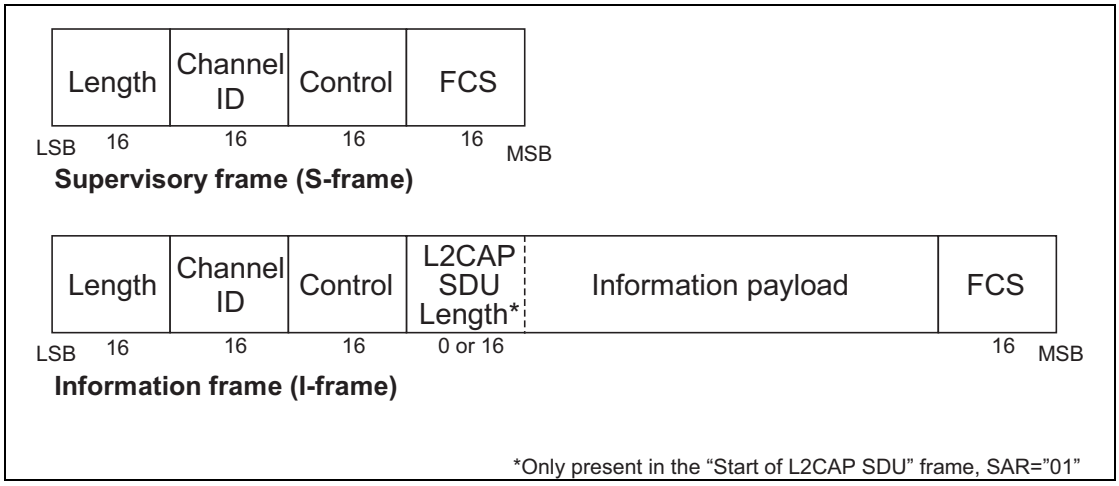


Figure 3.3: L2CAP PDU formats in Flow Control and Retransmission Modes

3.3.1 L2CAP header fields

- *Length: 2 octets*

  The first two octets in the L2CAP PDU contain the length of the entire L2CAP PDU in octets, excluding the Length and CID field.

  For I-frames and S-frames, the Length field therefore includes the octet lengths of the Control, L2CAP SDU Length (when present), Information octets and frame check sequence (FCS) fields.

  If the L2CAP SDU length field is present then the maximum number of Information octets in one I-frame is 65529 octets. If the L2CAP SDU length field is not present then the maximum number of Information octets in one I-frame is 65531 octets.

- *Channel ID: 2 octets*

  This field contains the Channel Identification (CID).

3.3.2 Control field (2 octets)

The Control field identifies the type of frame. The control field will contain sequence numbers where applicable. Its coding is shown in Table 3.1 on page 36. There are two different frame types, Information frame types and Supervisory frame types. Information and Supervisory frames types are distinguished by the rightmost bit in the Control field, as shown in Table 3.1 on page 36.

- *Information frame format (I-frame)*

  The I-frames are used to transfer information between L2CAP entities. Each I-frame has a TxSeq(Send sequence number), ReqSeq(Receive sequence number) which may or may not acknowledge additional I-frames received by the data link layer entity, and a retransmission bit (R bit) that affects whether I-frames are retransmitted.

  The SAR field in the I-frame is used for segmentation and reassembly control. The L2CAP SDU Length field specifies the length of an SDU, including the aggregate length across all segments if segmented.

- *Supervisory frame format (S-frame)*

  S-frames are used to acknowledge I-frames and request retransmission of I-frames. Each S-frame has an ReqSeq sequence number which may acknowledge additional I-frames received by the data link layer entity, and a retransmission bit (R bit) that affects whether I-frames are retransmitted.

  Defined types of S-frames are RR (Receiver Ready) and REJ (Reject).

Frame type	**16**	**15**	**14**	**13**	**12**	**11**	**10**	**9**	**8**	**7**	**6**	**5**	**4**	**3**	**2**	**1**
I	SAR		ReqSeq						R	TxSeq						0
S	X	X	ReqSeq						R	X	X	X	S		0	1

X denotes reserved bits. Shall be coded 0.

Table 3.1: Control Field formats

- *Send Sequence Number - TxSeq (6 bits)*

  The send sequence number is used to number each I-frame, to enable sequencing and retransmission.

- *Receive Sequence Number - ReqSeq (6 bits)*

  The receive sequence number is used by the receiver side to acknowledge I-frames, and in the REJ frame to request the retransmission of an I-frame with a specific send sequence number.

- *Retransmission Disable Bit - R (1 bit)*

  The Retransmission Disable bit is used to implement Flow Control. The receiver sets the bit when its internal receive buffer is full, this happens when one or more I-frames have been received but the SDU reassembly function has not yet pulled all the frames received. When the sender receives a frame with the Retransmission Disable bit set it shall disable the RetransmissionTimer, this causes the sender to stop retransmitting I-frames.

  R=0: Normal operation. Sender uses the RetransmissionTimer to control retransmission of I-frames. Sender does not use the MonitorTimer.

  R=1: Receiver side requests sender to postpone retransmission of I-frames. Sender monitors signalling with the MonitorTimer. Sender does not use the RetransmissionTimer.

  The functions of ReqSeq and R are independent.

- *Segmentation and Reassembly - SAR (2 bits)*

  The SAR bits define whether an L2CAP SDU is segmented. For segmented SDUs, the SAR bits also define which part of an SDU is in this I-frame, thus allowing one L2CAP SDU to span several I-frames.

  An I-frame with SAR=”Start of L2CAP SDU” also contains a length indicator, specifying the number of information octets in the total L2CAP SDU. The encoding of the SAR bits is shown in Table 3.2.

00	Unsegmented L2CAP SDU
01	Start of L2CAP SDU
10	End of L2CAP SDU
11	Continuation of L2CAP SDU

Table 3.2: SAR control element format.

- *Supervisory function - S (2 bits)*

  The S-bits mark the type of S-frame. There are two types defined: RR (Receiver Ready) and REJ (Reject). The encoding is shown in Table 3.3.

00	RR - Receiver Ready
01	REJ - Reject
10	Reserved
11	Reserved

Table 3.3: S control element format: type of S-frame.

3.3.3 L2CAP SDU length field (2 octets)

When a SDU spans more than one I-frame, the first I-frame in the sequence shall be identified by SAR=01=”Start of L2CAP SDU”. The L2CAP SDU Length field shall specify the total number of octets in the SDU. The L2CAP SDU Length field shall be present in I-frames with SAR=01 (Start of L2CAP SDU), and shall not be used in any other I-frames. When the SDU is unsegmented (SAR=00), L2CAP SDU Length field is not needed and shall not be present.

3.3.4 Information payload field (0 to 65531 octets)

The information payload field consists of an integral number of octets. The maximum number of octets in this field is the same as the negotiated value of the MPS configuration parameter. The maximum number of octets in this field is also limited by the range of the basic L2CAP header length field. For I-frames without an SDU length field, this limits the maximum number of octets in the field to 65531. For I-frames with an SDU length field, SAR=01, this limits maximum number of octets in the field to 65529. Thus, even if an MPS of 65531 has been negotiated, the range of the basic L2CAP header length field will restrict the number of octets in this field when an SDU length field is present to 65529.

3.3.5 Frame check sequence (2 octets)

The Frame Check Sequence (FCS) is 2 octets. The FCS is constructed using the generator polynomial $g(D) = D^{16} + D^{15} + D^2 + 1$ (see Figure 3.4). The 16 bit LFSR is initially loaded with the value 0x0000, as depicted in Figure 3.5. The switch S is set in position 1 while data is shifted in, LSB first for each octet. After the last bit has entered the LFSR, the switch is set in position 2, and the register contents are transmitted from right to left (i.e. starting with position 15, then position 14, etc.). The FCS covers the Basic L2CAP header, Control, L2CAP-SDU length and Information payload fields, if present, as shown in Figure 3.3 on page 35.

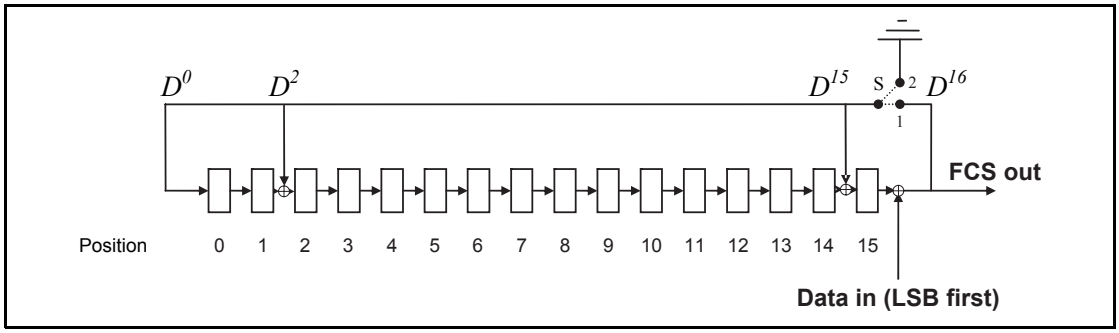


Figure 3.4: The LFSR circuit generating the FCS.

Position	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
LFSR	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 3.5: Initial state of the FCS generating circuit.

Examples of FCS calculation, $g(D) = D^{16} + D^{15} + D^2 + 1$:

1. ***I Frame***
   Length = 14
   Control = 0x0002 (SAR=0, ReqSeq=0, R=0, TxSeq=1)
   Information Payload = 00 01 02 03 04 05 06 07 08 09 (10 octets, hexadecimal notation)
   ==> FCS = 0x6138
   ==> Data to Send = 0E 00 40 00 02 00 00 01 02 03 04 05 06 07 08 09 38 61 (hexadecimal notation)
2. ***RR Frame***
   Length = 4
   Control = 0x0101 (ReqSeq=1, R=0, S=0)
   ==> FCS = 0x14D4
   ==> Data to Send = 04 00 40 00 01 01 D4 14 (hexadecimal notation)

3.3.6 Invalid frame detection

A received PDU shall be regarded as invalid, if one of the following conditions occurs:

1. Contains an unknown CID.
2. Contains an FCS error.
3. Contains a length greater than the maximum PDU payload size (MPS).
4. I-frame that has fewer than 8 octets.
5. I-frame with SAR=01 (Start of L2CAP SDU) that has fewer than 10 octets.
6. I-frame with SAR bits that do not correspond to a normal sequence of either unsegmented or start, continuation, end for the given CID.
7. S-frame where the length field is not equal to 4.

These error conditions may be used for error reporting.

4 SIGNALLING PACKET FORMATS

This section describes the signalling commands passed between two L2CAP entities on peer devices. All signalling commands are sent to the signalling channel with CID 0x0001. This signalling channel is available as soon as an ACL logical transport is set up and L2CAP traffic is enabled on the L2CAP logical link. Figure 4.1 on page 41 illustrates the general format of L2CAP PDUs containing signalling commands (C-frames). Multiple commands may be sent in a single C-frame. Commands take the form of Requests and Responses. All L2CAP implementations shall support the reception of C-frames with a payload length that does not exceed the signaling MTU. The minimum supported payload length for the C-frame (MTU_{sig}) is 48 octets. L2CAP implementations should not use C-frames that exceed the MTU_{sig} of the peer device. If they ever do, the peer device shall send a Command Reject containing the supported MTU_{sig}. Implementations must be able to handle the reception of multiple commands in an L2CAP packet.

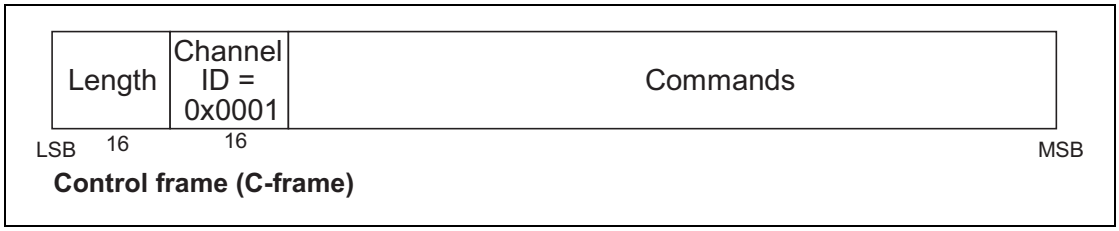


Figure 4.1: L2CAP PDU format on the signalling channel

Figure 4.2 displays the general format of all signalling commands.

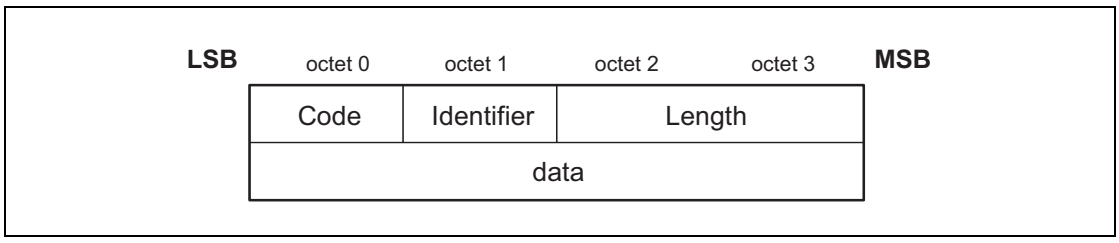


Figure 4.2: Command format

The fields shown are:

- *Code (1 octet)*

  The Code field is one octet long and identifies the type of command. When a packet is received with an unknown Code field, a Command Reject packet (defined in Section 4.1 on page 43) is sent in response.

  Table 4.1 on page 42 lists the codes defined by this document. All codes are specified with the most significant bit in the left-most position.

Code	Description
0x00	RESERVED
0x01	Command reject
0x02	Connection request
0x03	Connection response
0x04	Configure request
0x05	Configure response
0x06	Disconnection request
0x07	Disconnection response
0x08	Echo request
0x09	Echo response
0x0A	Information request
0x0B	Information response

Table 4.1: Signalling Command Codes

- *Identifier (1 octet)*

  The Identifier field is one octet long and matches responses with requests. The requesting device sets this field and the responding device uses the same value in its response. Between any two devices a different Identifier shall be used for each successive command. Following the original transmission of an Identifier in a command, the Identifier may be recycled if all other Identifiers have subsequently been used.

  RTX and ERTX timers are used to determine when the remote end point is not responding to signalling requests. On the expiration of a RTX or ERTX timer, the same identifier shall be used if a duplicate Request is re-sent as stated in Section 6.2 on page 75.

  A device receiving a duplicate request should reply with a duplicate response. A command response with an invalid identifier is silently discarded. Signaling identifier 0x00 is an illegal identifier and shall never be used in any command.

- *Length (2 octets)*

  The Length field is two octets long and indicates the size in octets of the data field of the command only, i.e., it does not cover the Code, Identifier, and Length fields.

- *Data (0 or more octets)*

  The Data field is variable in length. The Code field determines the format of the Data field. The length field determines the length of the data field.

4.1 COMMAND REJECT (CODE 0x01)

A Command Reject packet shall be sent in response to a command packet with an unknown command code or when sending the corresponding response is inappropriate. Figure 4.3 on page 43 displays the format of the packet. The identifier shall match the identifier of the command packet being rejected. Implementations shall always send these packets in response to unidentified signalling packets. Command Reject packets should not be sent in response to an identified Response packet.

When multiple commands are included in an L2CAP packet and the packet exceeds the signalling MTU (MTU_{sig}) of the receiver, a single Command Reject packet shall be sent in response. The identifier shall match the first Request command in the L2CAP packet. If only Responses are recognized, the packet shall be silently discarded.

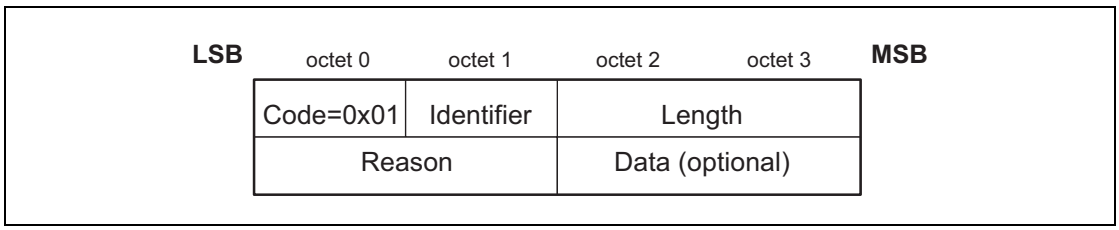


Figure 4.3: Command Reject packet

Figure 4.3 shows the format of the Command Reject packet. The data fields are:

- *Reason (2 octets)*

  The Reason field describes why the Request packet was rejected, and is set to one of the Reason codes in Table 4.2.

Reason value	Description
0x0000	Command not understood
0x0001	Signalling MTU exceeded
0x0002	Invalid CID in request
Other	Reserved

Table 4.2: Reason Code Descriptions

- *Data (0 or more octets)*

  The length and content of the Data field depends on the Reason code. If the Reason code is 0x0000, “Command not understood”, no Data field is used. If the Reason code is 0x0001, “Signalling MTU Exceeded”, the 2-octet Data field represents the maximum signalling MTU the sender of this packet can accept.

  If a command refers to an invalid channel then the Reason code 0x0002 will be returned. Typically a channel is invalid because it does not exist. The data field shall be 4 octets containing the local (first) and remote (second) channel endpoints (relative to the sender of the Command Reject) of the disputed channel. The remote endpoint is the source CID from the rejected command. The local endpoint is the destination CID from the rejected command. If the rejected command contains only one of the channel endpoints, the other one shall be replaced by the null CID 0x0000.

Reason value	Data Length	Data value
0x0000	0 octets	N/A
0x0001	2 octets	Actual MTU_{sig}
0x0002	4 octets	Requested CID

Table 4.3: Reason Data values

4.2 CONNECTION REQUEST (CODE 0x02)

Connection request packets are sent to create an L2CAP channel between two devices. The L2CAP channel shall be established before configuration begins. Figure 4.4 illustrates a Connection Request packet.

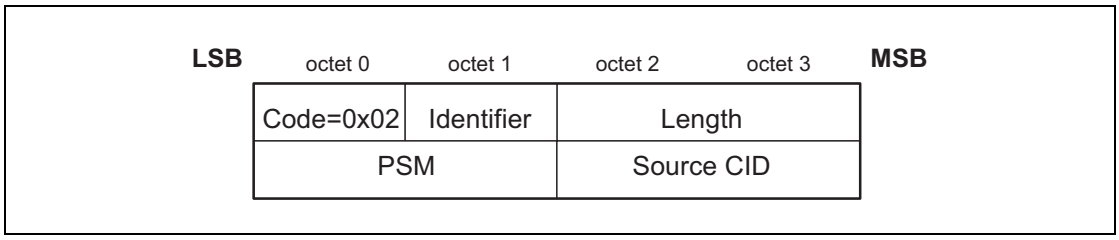


Figure 4.4: Connection Request Packet

The data fields are:

- *Protocol/Service Multiplexor - PSM (2 octets (minimum))*

  The PSM field is at least two octets in length. The structure of the PSM field is based on the ISO 3309 extension mechanism for address fields. All PSM values shall be ODD, that is, the least significant bit of the least significant octet must be ’1’. Also, all PSM values shall have the least significant bit of the most significant octet equal to ’0’. This allows the PSM field to be extended beyond 16 bits. PSM values are separated into two ranges. Values in the first range are assigned by the Bluetooth SIG and indicate protocols. The second range of values are dynamically allocated and used in conjunction with the Service Discovery Protocol (SDP). The dynamically assigned values may be used to support multiple implementations of a particular protocol.

PSM value	Description
0x0001	Service Discovery Protocol
0x0003	RFCOMM
0x0005	Telephony Control Protocol
0x0007	TCS cordless
0x000F	BNEP
0x0011	HID Control
0x0013	HID Interrupt
0x0015	UPnP (ESDP)
0x0017	AVCTP
0x0019	AVDTP
Other < 1000	RESERVED
[0x1001-0xFFFF]	DYNAMICALLY ASSIGNED

Table 4.4: Defined PSM Values[1]

1. The most recent PSM assignments can be found in the Assigned Numbers database on the Bluetooth member web site.

- *Source CID - SCID (2 octets)*

  The source CID is two octets in length and represents a channel endpoint on the device sending the request. Once the channel has been configured, data packets flowing to the sender of the request shall be sent to this CID. Thus, the Source CID represents the channel endpoint on the device sending the request and receiving the response.

4.3 CONNECTION RESPONSE (CODE 0x03)

When a device receives a Connection Request packet, it shall send a Connection Response packet. The format of the connection response packet is shown in Figure 4.5.

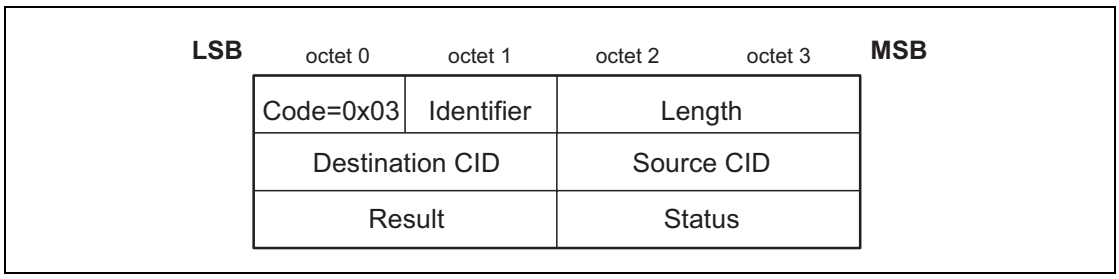


Figure 4.5: Connection Response Packet

The data fields are:

- *Destination Channel Identifier - DCID (2 octets)*

  This field contains the channel endpoint on the device sending this Response packet. Thus, the Destination CID represents the channel endpoint on the device receiving the request and sending the response.

- *Source Channel Identifier - SCID (2 octets)*

  This field contains the channel endpoint on the device receiving this Response packet. This is copied from the SCID field of the connection request packet.

- *Result (2 octets)*

  The result field indicates the outcome of the connection request. The result value of 0x0000 indicates success while a non-zero value indicates the connection request failed or is pending. A logical channel is established on the receipt of a successful result. Table 4.5 on page 46 defines values for this field. The DCID and SCID fields shall be ignored when the result field indicates the connection was refused.

Value	Description
0x0000	Connection successful.
0x0001	Connection pending
0x0002	Connection refused – PSM not supported.
0x0003	Connection refused – security block.
0x0004	Connection refused – no resources available.
Other	Reserved.

Table 4.5: Result values

- *Status (2 octets)*

  Only defined for Result = Pending. Indicates the status of the connection. The status is set to one of the values shown in Table 4.6 on page 47.

Value	Description
0x0000	No further information available
0x0001	Authentication pending
0x0002	Authorization pending
Other	Reserved

Table 4.6: Status values

4.4 CONFIGURATION REQUEST (CODE 0x04)

Configuration Request packets are sent to establish an initial logical link transmission contract between two L2CAP entities and also to re-negotiate this contract whenever appropriate. During a re-negotiation session, all data traffic on the channel should be suspended pending the outcome of the negotiation. Each configuration parameter in a Configuration Request shall be related exclusively to either the outgoing or the incoming data traffic but not both of them. In Section 5 on page 57, the various configuration parameters and their relation to the outgoing or incoming data traffic are shown. If an L2CAP entity receives a Configuration Request while it is waiting for a response it shall not block sending the Configuration Response, otherwise the configuration process may deadlock.

If no parameters need to be negotiated then no options shall be inserted and the continuation flag (C) shall be set to zero. L2CAP entities in remote devices shall negotiate all parameters defined in this document whenever the default values are not acceptable. Any missing configuration parameters are assumed to have their most recently explicitly or implicitly accepted values. Even if all default values are acceptable, a Configuration Request packet with no options shall be sent. Implicitly accepted values are default values for the configuration parameters that have not been explicitly negotiated for the specific channel under configuration.

Each configuration parameter is one-directional. The configuration parameters describe the non default parameters the device sending the Configuration Request will accept. The configuration request can not request a change in the parameters the device receiving the request will accept.

If a device needs to establish the value of a configuration parameter the remote device will accept, then it must wait for a configuration request containing that configuration parameter to be sent from the remote device.

See Section 7.1 on page 79 for details of the configuration procedure.

Figure 4.6 defines the format of the Configuration Request packet.

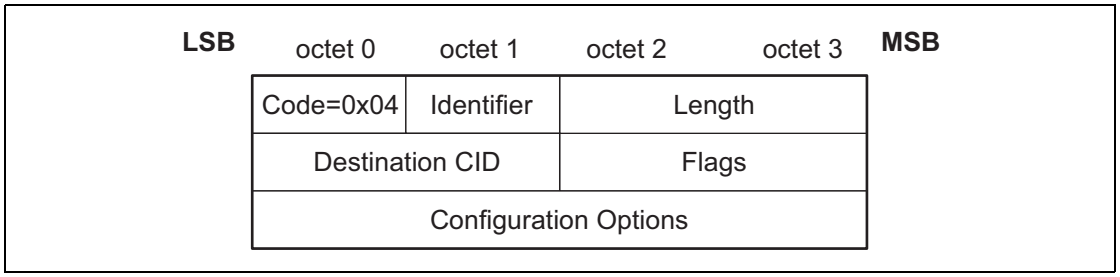


Figure 4.6: Configuration Request Packet

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field contains the channel endpoint on the device receiving this Request packet.

- *Flags (2 octets)*

  Figure 4.7 shows the two-octet Flags field. Note the most significant bit is shown on the left.

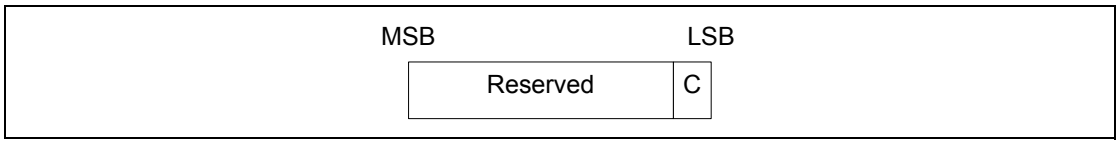


Figure 4.7: Configuration Request Flags field format

Only one flag is defined, the Continuation flag (C).

When all configuration options cannot fit into a Configuration Request with length that does not exceed the receiver's MTU_{sig}, the options shall be passed in multiple configuration command packets. If all options fit into the receiver's MTU_{sig}, then they shall be sent in a single configuration request with the continuation flag set to zero. Each Configuration Request shall contain an integral number of options - partially formed options shall not be sent in a packet. Each Request shall be tagged with a different Identifier and shall be matched with a Response with the same Identifier.

When used in the Configuration Request, the continuation flag indicates the responder should expect to receive multiple request packets. The responder shall reply to each Configuration Request packet. The responder may reply to each Configuration Request with a Configuration Response containing the same option(s) present in the Request (except for those error conditions more appropriate for a Command Reject), or the responder may reply with a "Success" Configuration Response packet containing no options, delaying those options until the full Request has been received. The Configuration Request packet with the continuation flag cleared shall be treated as the Configuration Request event in the channel state machine.

When used in the Configuration Response, the continuation flag shall be set to one if the flag is set to one in the Request. If the continuation flag is set to one in the Response when the matching Request has the flag set to zero, it indicates the responder has additional options to send to the requestor. In this situation, the requestor shall send null-option Configuration Requests (with continuation flag set to zero) to the responder until the responder replies with a Configuration Response where the continuation flag is set to zero. The Configuration Response packet with the continuation flag set to zero shall be treated as the Configuration Response event in the channel state machine.

The result of the configuration transaction is the union of all the result values. All the result values must succeed for the configuration transaction to succeed.

Other flags are reserved and shall be set to zero. L2CAP implementations shall ignore these bits.

- *Configuration Options*

  A list of the parameters and their values to be negotiated shall be provided in the Configuration Options field. These are defined in Section 5 on page 57. A Configuration Request may contain no options (referred to as an empty or null configuration request) and can be used to request a response. For an empty configuration request the length field is set to 0x0004.

4.5 CONFIGURATION RESPONSE (CODE 0X05)

Configuration Response packets shall be sent in reply to Configuration Request packets except when the error condition is covered by a Command Reject response. Each configuration parameter value (if any is present) in a Configuration Response reflects an 'adjustment' to a configuration parameter value that has been sent (or, in case of default values, implied) in the corresponding Configuration Request. For example, if a configuration request relates to traffic flowing from device A to device B, the sender of the configuration response may adjust this value for the same traffic flowing from device A to device B, but the response can not adjust the value in the reverse direction.

The options sent in the Response depend on the value in the Result field. Figure 4.8 on page 50 defines the format of the Configuration Response packet. See also Section 7.1 on page 79 for details of the configuration process.

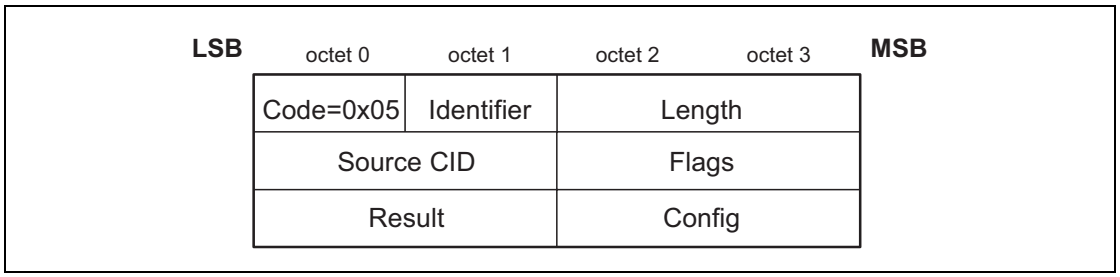


Figure 4.8: Configuration Response Packet

The data fields are:

- *Source CID - SCID (2 octets)*

  This field contains the channel endpoint on the device receiving this Response packet. The device receiving the Response shall check that the Identifier field matches the same field in the corresponding configuration request command and the SCID matches its local CID paired with the original DCID.

- *Flags (2 octets)*

  Figure 4.9 displays the two-octet Flags field. Note the most significant bit is shown on the left.

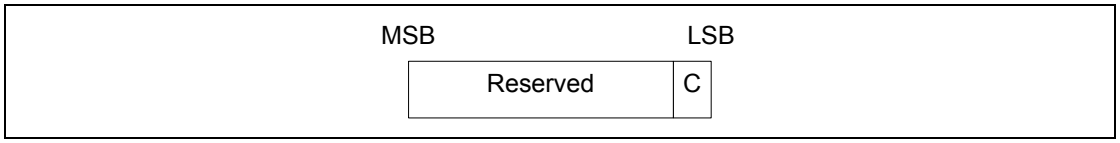


Figure 4.9: Configuration Response Flags field format

Only one flag is defined, the Continuation flag (C).

More configuration responses will follow when C is set to one. This flag indicates that the parameters included in the response are a partial subset of parameters being sent by the device sending the Response packet.

The other flag bits are reserved and shall be set to zero. L2CAP implementations shall ignore these bits.

- *Result (2 octets)*

  The Result field indicates whether or not the Request was acceptable. See Table 4.7 on page 51 for possible result codes.

Result	Description
0x0000	Success
0x0001	Failure – unacceptable parameters
0x0002	Failure – rejected (no reason provided)
0x0003	Failure – unknown options
Other	RESERVED

Table 4.7: Configuration Response Result codes

- *Configuration Options*

  This field contains the list of parameters being configured. These are defined in Section 5 on page 57. On a successful result, these parameters contain the return values for any wild card parameter values (see Section 5.3 on page 60) contained in the request.

  On an unacceptable parameters failure (Result=0x0001) the rejected parameters shall be sent in the response with the values that would have been accepted if sent in the original request. Any missing configuration parameters are assumed to have their most recently accepted values and they too shall be included in the Configuration Response if they need to be changed.

  Each configuration parameter is one-directional. The configuration parameters describe the non default parameters the device sending the Configuration Request will accept. The configuration request can not request a change in the parameters the device receiving the request will accept.

  If a device needs to establish the value of a configuration parameter the remote device will accept, then it must wait for a configuration request containing that configuration parameter to be sent from the remote device.

  On an unknown option failure (Result=0x0003), the option types not understood by the recipient of the Request shall be included in the Response unless they are hints. Hints are those options in the Request that are skipped if not understood (see Section 5 on page 57). Hints shall not be included in the Response and shall not be the sole cause for rejecting the Request.

  The decision on the amount of time (or messages) spent arbitrating the channel parameters before terminating the negotiation is implementation specific.

4.6 DISCONNECTION REQUEST (CODE 0x06)

Terminating an L2CAP channel requires that a disconnection request be sent and acknowledged by a disconnection response. Figure 4.10 on page 52 shows a disconnection request. The receiver shall ensure that both source and destination CIDs match before initiating a channel disconnection.

Once a Disconnection Request is issued, all incoming data in transit on this L2CAP channel shall be discarded and any new additional outgoing data shall be discarded. Once a disconnection request for a channel has been received, all data queued to be sent out on that channel shall be discarded.

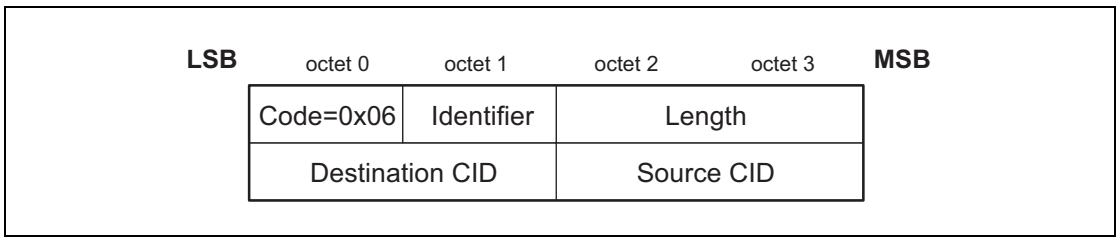


Figure 4.10: Disconnection Request Packet

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field specifies the endpoint of the channel to be disconnected on the device receiving this request.

- *Source CID - SCID (2 octets)*

  This field specifies the endpoint of the channel to be disconnected on the device sending this request.

The SCID and DCID are relative to the sender of this request and shall match those of the channel to be disconnected. If the DCID is not recognized by the receiver of this message, a CommandReject message with ’invalid CID’ result code shall be sent in response. If the receiver finds a DCID match but the SCID fails to find the same match, the request should be silently discarded.

4.7 DISCONNECTION RESPONSE (CODE 0x07)

Disconnection responses shall be sent in response to each valid disconnection request.

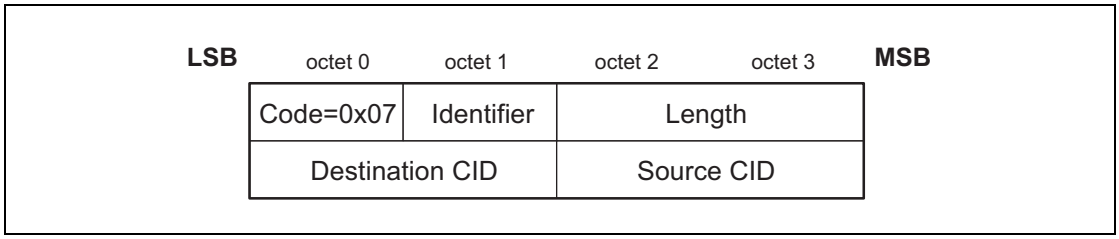


Figure 4.11: Disconnection Response Packet

The data fields are:

- *Destination CID - DCID (2 octets)*

  This field identifies the channel endpoint on the device sending the response.

- *Source CID - SCID (2 octets)*

  This field identifies the channel endpoint on the device receiving the response.

  The DCID and the SCID (which are relative to the sender of the request), and the Identifier fields shall match those of the corresponding disconnection request command. If the CIDs do not match, the response should be silently discarded at the receiver.

4.8 ECHO REQUEST (CODE 0x08)

Echo requests are used to request a response from a remote L2CAP entity. These requests may be used for testing the link or for passing vendor specific information using the optional data field. L2CAP entities shall respond to a valid Echo Request packet with an Echo Response packet. The Data field is optional and implementation specific. L2CAP entities should ignore the contents of this field if present.

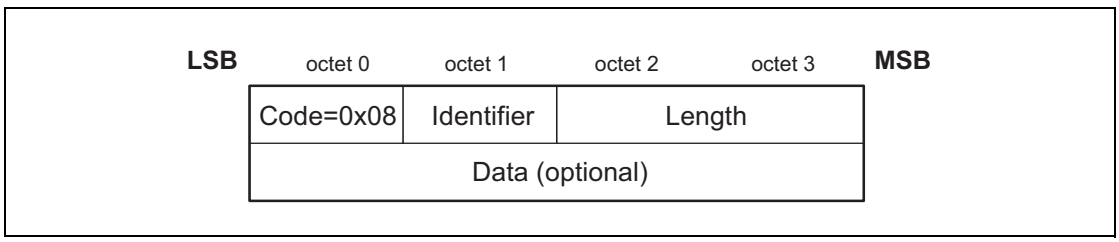


Figure 4.12: Echo Request Packet

4.9 ECHO RESPONSE (CODE 0x09)

An Echo response shall be sent upon receiving a valid Echo Request. The identifier in the response shall match the identifier sent in the Request. The optional and implementation specific data field may contain the contents of the data field in the Request, different data, or no data at all.

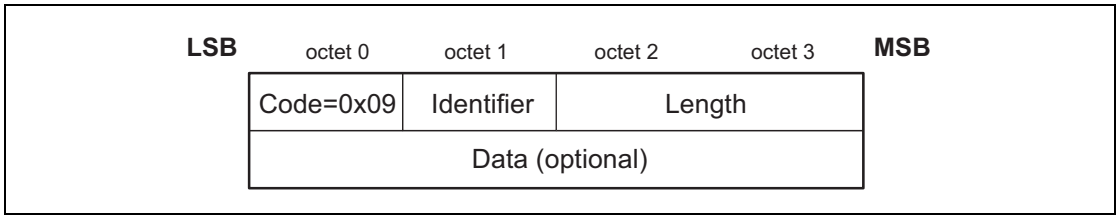


Figure 4.13: Echo Response Packet

4.10 INFORMATION REQUEST (CODE 0X0A)

Information requests are used to request implementation specific information from a remote L2CAP entity. L2CAP implementations shall respond to a valid Information Request with an Information Response. It is optional to send Information Requests.

An L2CAP implementation shall only use optional features or attribute ranges for which the remote L2CAP entity has indicated support through an Information Response. Until an Information Response which indicates support for optional features or ranges has been received only mandatory features and ranges shall be used.

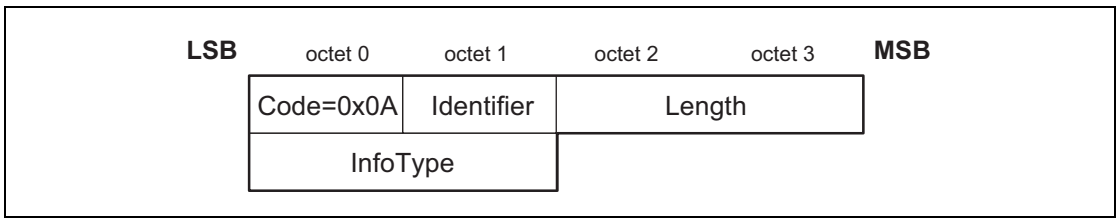


Figure 4.14: Information Request Packet

The data fields are:

- *InfoType (2 octets)*

  The InfoType defines the type of implementation specific information being requested. See Section 4.11 on page 55 for details on the type of information requested.

Value	Description
0x0001	Connectionless MTU
0x0002	Extended features supported
Other	Reserved

Table 4.8: InfoType definitions

4.11 INFORMATION RESPONSE (CODE 0X0B)

An information response shall be sent upon receiving a valid Information Request. The identifier in the response shall match the identifier sent in the Request. The data field shall contain the value associated with the InfoType field sent in the Request, or shall be empty if the InfoType is not supported.

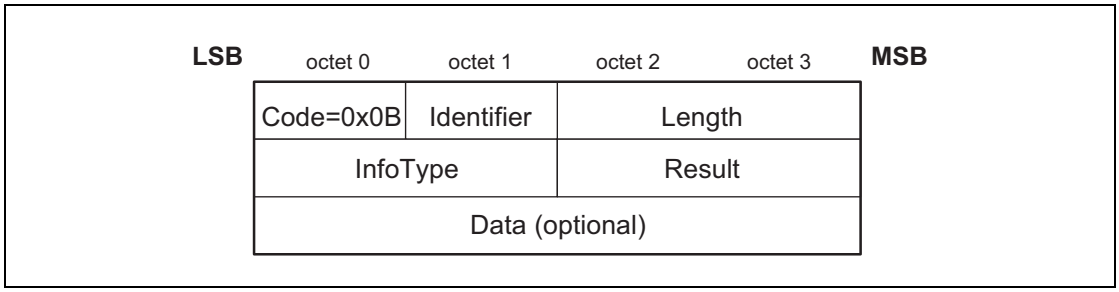


Figure 4.15: Information Response Packet

The data fields are:

- *InfoType (2 octets)*

  The InfoType defines the type of implementation specific information that was requested. This value shall be copied from the InfoType field in the Information Request.

- *Result (2 octets)*

  The Result contains information about the success of the request. If result is "Success", the data field contains the information as specified in Table 4.10 on page 56. If result is "Not supported", no data shall be returned.

Value	Description
0x0000	Success
0x0001	Not supported
Other	Reserved

Table 4.9: Information Response Result values

- *Data (0 or more octets)*

  The contents of the Data field depends on the InfoType.
  For InfoType = 0x0001 the data field contains the remote entity's 2-octet acceptable connectionless MTU. The default value is defined in Section 3.2 on page 34.

  For InfoType = 0x0002, the data field contains the 4 octet L2CAP extended feature mask. The feature mask refers to the extended features that the L2CAP entity sending the Information Response supports. The feature bits contained in the L2CAP feature mask are specified in Section 4.12 on page 56.

Note: L2CAP entities of versions prior to version 1.2, receiving an Information Request with InfoType = 0x0002 for an L2CAP feature discovery, will return an Information Response with result code "Not supported". L2CAP entities at version 1.2 or later that have an all zero extended features mask may return an Information Response with result code "Not supported".

InfoType	Data	Data Length (octets)
0x0001	Connectionless MTU	2
0x0002	Extended feature mask	4

Table 4.10: Information Response Data fields

4.12 EXTENDED FEATURE MASK

The features are represented as a bit mask in the Information Response data field (see Section 4.11 on page 55). For each feature a single bit is specified which shall be set to 1 if the feature is supported and set to 0 otherwise. All unknown, reserved, or unassigned feature bits shall be set to 0.

The feature mask shown in Table 4.11 on page 56 consists of 4 octets (numbered octet 0 ... 3), with bit numbers 0 ... 7 each. Within the Information Response packet data field, bit 0 of octet 0 is aligned leftmost, bit 7 of octet 3 is aligned rightmost.

Note: the L2CAP feature mask is a new concept introduced in Bluetooth v1.2 and thus contains new features introduced after Bluetooth v1.1.

No.	Supported feature	Octet	Bit
0	Flow control mode	0	0
1	Retransmission mode	0	1
2	Bi-directional QoS[1]	0	2
31	Reserved for feature mask extension	3	7

Table 4.11: Extended feature mask.

1. Peer side supports upper layer control of the Link Manager's Bi-directional QoS, see Section 5.3 on page 60 for more details.

5 CONFIGURATION PARAMETER OPTIONS

Options are a mechanism to extend the configuration parameters. Options shall be transmitted as information elements containing an option type, an option length, and one or more option data fields. Figure 5.1 illustrates the format of an option.

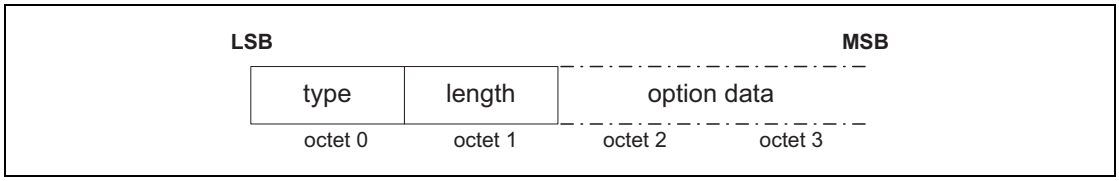


Figure 5.1: Configuration option format

The configuration option fields are:

- *Type (1 octet)*

  The option type field defines the parameters being configured. The most significant bit of the type determines the action taken if the option is not recognized.

  0 - option must be recognized; if the option is not recognized then refuse the configuration request

  1 - option is a hint; if the option is not recognized then skip the option and continue processing

- *Length (1 octet)*

  The length field defines the number of octets in the option data. Thus an option type without option data has a length of 0.

- *Option data*

  The contents of this field are dependent on the option type.

5.1 MAXIMUM TRANSMISSION UNIT (MTU)

This option specifies the maximum SDU size the sender of this option is capable of accepting for a channel. The type is 0x01, and the payload length is 2 octets, carrying the two-octet MTU size value as the only information element (see Figure 5.2 on page 58). Unlike the B-Frame length field, the I-Frame length field may be greater then the configured MTU because it includes the octet lengths of the Control, L2CAP SDU Length (when present), and frame check sequence fields as well as the Information octets.

MTU is not a negotiated value, it is an informational parameter that each device can specify independently. It indicates to the remote device that the local device can receive, in this channel, an MTU larger than the minimum required. All L2CAP implementations shall support a minimum MTU of 48 octets, however some protocols and profiles explicitly require support for a

larger MTU. The minimum MTU for a channel is the larger of the L2CAP minimum 48 octet MTU and any MTU explicitly required by the protocols and profiles using that channel. (Note: the MTU is only affected by the profile directly using the channel. For example, if a service discovery transaction is initiated by a non service discovery profile, that profile does not affect the MTU of the L2CAP channel used for service discovery).

The following rules shall be used when responding to a configuration request specifying the MTU for a channel:

- A request specifying any MTU greater than or equal to the minimum MTU for the channel shall be accepted.
- A request specifying an MTU smaller than the minimum MTU for the channel may be rejected.

The signalling described in Section 4.5 on page 50 may be used to reject an MTU smaller than the minimum MTU for a channel. The "failure-unacceptable parameters" result sent to reject the MTU shall include the proposed value of MTU that the remote device intends to transmit. It is implementation specific whether the local device continues the configuration process or disconnects the channel.

If the remote device sends a positive configuration response it shall include the actual MTU to be used on this channel for traffic flowing into the local device. This is the minimum of the MTU in the configuration request and the outgoing MTU capability of the device sending the configuration response. The new agreed value (the default value in a future re-configuration) is the value specified in the request.

The MTU to be used on this channel for the traffic flowing in the opposite direction will be established when the remote device sends its own Configuration Request as explained in Section 4.4 on page 47.

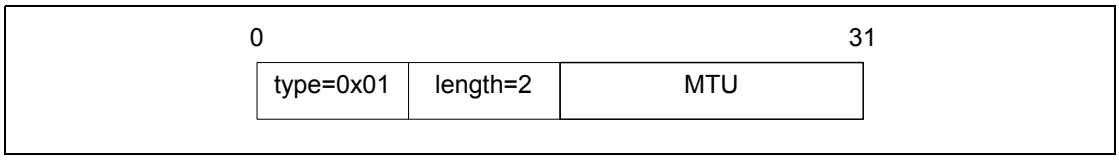


Figure 5.2: MTU Option Format

The option data field is:

- Maximum Transmission Unit - MTU (2 octets)

  The MTU field is the maximum SDU size, in octets, that the originator of the Request can accept for this channel. The MTU is asymmetric and the sender of the Request shall specify the MTU it can receive on this channel if it differs from the default value. L2CAP implementations shall support a minimum MTU size of 48 octets. The default value is 672 octets[1].

5.2 FLUSH TIMEOUT OPTION

This option is used to inform the recipient of the Flush Timeout the sender is going to use. The Flush Timeout is defined in the Baseband specification “Flushing payloads” on page 150[vol. 3]. The type is 0x02 and the payload size is 2 octets.

If the remote device returns a negative response to this option and the local device cannot honor the proposed value, then it shall either continue the configuration process by sending a new request with the original value, or disconnect the channel. The flush timeout applies to all channels on the same ACL logical transport and other channels on the same ACL logical transport may therefore have other values.

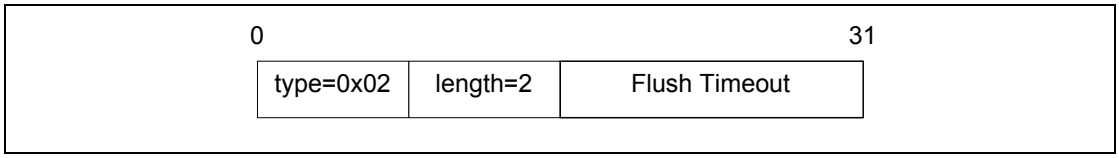


Figure 5.3: Flush Timeout option format.

The option data field is:

- *Flush Timeout*

  This value is the Flush Timeout in milliseconds. This is an asymmetric value and the sender of the Request shall specify its flush timeout value if it differs from the default value of 0xFFFF.

  Possible values are:

  0x0001 - no retransmissions at the baseband level should be performed since the minimum polling interval is 1.25 ms.

  0x0002 to 0xFFFE - Flush Timeout used by the baseband.

  0xFFFF - an infinite amount of retransmissions. This is also referred to as a ’reliable channel’. In this case, the baseband shall continue retransmissions until physical link loss is declared by link manager timeouts.

1. The default MTU was selected based on the payload carried by two baseband DH5 packets (2*341=682) minus the baseband ACL headers (2*2=4) and L2CAP header (6).

5.3 QUALITY OF SERVICE (QOS) OPTION

This option specifies a flow specification similar to RFC 1363[1]. Although the RFC flow specification addresses only the transmit characteristics, the Bluetooth QoS interface can handle the two directions (Tx and Rx) in the negotiation as described below.

If no QoS configuration parameter is negotiated the link shall assume the default parameters. The QoS option is type 0x03.

In a configuration request, this option describes the outgoing traffic flow from the device sending the request. In a positive Configuration Response, this option describes the incoming traffic flow agreement to the device sending the response. In a negative Configuration Response, this option describes the preferred incoming traffic flow to the device sending the response.

L2CAP implementations are only required to support 'Best Effort' service, support for any other service type is optional. Best Effort does not require any guarantees. If no QoS option is placed in the request, Best Effort shall be assumed. If any QoS guarantees are required then a QoS configuration request shall be sent.

The remote device's Configuration Response contains information that depends on the value of the result field (see Section 4.5 on page 50). If the request was for Guaranteed Service, the response shall include specific values for any wild card parameters (see Token Rate and Token Bucket Size descriptions) contained in the request. If the result is "Failure – unacceptable parameters", the response shall include a list of outgoing flow specification parameters and parameter values that would make a new Connection Request from the local device acceptable by the remote device. Both explicitly referenced in a Configuration Request or implied configuration parameters can be included in a Configuration Response. Recall that any missing configuration parameters from a Configuration Request are assumed to have their most recently accepted values.

If a configuration request contains any QoS option parameters set to "do not care" then the configuration response shall set the same parameters to "do not care". This rule applies for both Best Effort and Guaranteed Service.

1. Internet Engineering Task Force, "A Proposed Flow Specification", RFC 1363, September 1992.

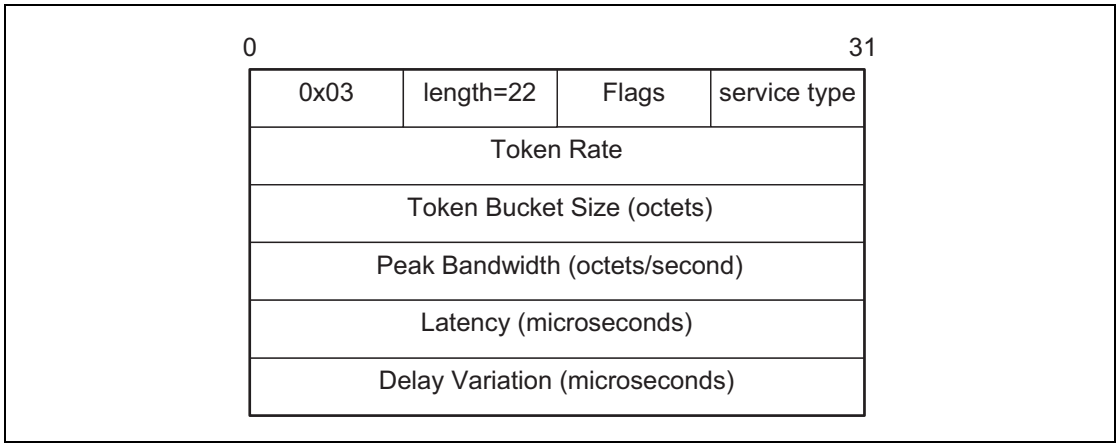


Figure 5.4: Quality of Service (QoS) option format containing Flow Specification.

The option data fields are:

- *Flags (1 octet)*

  Reserved for future use and shall be set to 0 and ignored by the receiver.

- *Service Type (1 octet)*

  This field indicates the level of service required. Table 5.1 on page 61 defines the different services available. The default value is ‘Best effort’.

  If ’Best effort’ is selected, the remaining parameters should be treated as optional by the remote device. The remote device may choose to ignore the fields, try to satisfy the parameters but provide no response (QoS option omitted in the Response message), or respond with the settings it will try to meet.

  If ’No traffic’ is selected, the remainder of the fields shall be ignored because there is no data being sent across the channel in the outgoing direction.

Value	Description
0x00	No traffic
0x01	Best effort (Default)
0x02	Guaranteed
Other	Reserved

Table 5.1: Service type definitions

- *Token Rate (4 octets)*

  The value of this field represents the average data rate with which the application transmits data. The application may send data at this rate continuously. On a short time scale the application may send data in excess of the average data rate, dependent on the specified Token Bucket Size and Peak Bandwidth (see below). The Token Bucket Size and Peak Bandwidth allow the application to transmit data in a 'bursty' fashion.

The Token Rate signalled between two L2CAP peers is the data transmitted by the application and shall exclude the L2CAP protocol overhead. The Token Rate signalled over the interface between L2CAP and the Link Manager shall include the L2CAP protocol overhead. Furthermore the Token Rate value signalled over this interface may also include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

The Token Rate is the rate with which traffic credits are provided. Credits can be accumulated up to the Token Bucket Size. Traffic credits are consumed when data is transmitted by the application. When traffic is transmitted, and there are insufficient credits available, the traffic is non-conformant. The Quality of Service guarantees are only provided for conformant traffic. For non-conformant traffic there may not be sufficient resources such as bandwidth and buffer space. Furthermore non-conformant traffic may violate the QoS guarantees of other traffic flows.

The Token Rate is specified in octets per second. The value 0x00000000 indicates no token rate is specified. This is the default value and means “do not care”. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token rate available. The meaning of this value depends on the service type. For best effort, the value is a hint that the application wants as much bandwidth as possible. For Guaranteed service the value represents the maximum bandwidth available at the time of the request.

- *Token Bucket Size (4 octets)*

  The Token Bucket Size specifies a limit on the 'burstiness' with which the application may transmit data. The application may offer a burst of data equal to the Token Bucket Size instantaneously, limited by the Peak Bandwidth (see below). The Token Bucket Size is specified in octets.

  The Token Bucket Size signalled between two L2CAP peers is the data transmitted by the application and shall exclude the L2CAP protocol overhead. The Token Bucket Size signalled over the interface between L2CAP and Link Manager shall include the L2CAP protocol overhead. Furthermore the Token Bucket Size value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The value of 0x00000000 means that no token bucket is needed; this is the default value. When the Guaranteed service is selected, the default value shall not be used. The value 0xFFFFFFFF is a wild card matching the maximum token bucket available. The meaning of this value depends on the service type. For best effort, the value indicates the application wants a bucket as big as possible. For Guaranteed service the value represents the maximum L2CAP SDU size.

  The Token Bucket Size is a property of the traffic carried over the L2CAP channel. The Maximum Transmission Unit (MTU) is a property of an L2CAP implementation. For the Guaranteed service the Token Bucket Size shall be smaller or equal to the MTU.

- *Peak Bandwidth (4 octets)*

  The value of this field, expressed in octets per second, limits how fast packets from applications may be sent back-to-back. Some systems can take advantage of this information, resulting in more efficient resource allocation.

  The Peak Bandwidth signalled between two L2CAP peers specifies the data transmitted by the application and shall exclude the L2CAP protocol overhead. The Peak Bandwidth signalled over the interface between L2CAP and Link Manager shall include the L2CAP protocol overhead. Furthermore the Peak Bandwidth value over this interface may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The value of 0x00000000 means "don't care". This states that the device has no preference on incoming maximum bandwidth, and is the default value. When the Guaranteed service is selected, the default value shall not be used.

- *Access Latency (4 octets)*

  The value of this field is the maximum acceptable delay of an L2CAP packet to the air-interface. The precise interpretation of this number depends on over which interface this flow parameter is signalled. When signalled between two L2CAP peers, the Access Latency is the maximum acceptable delay between the instant when the L2CAP SDU is received from the upper layer and the start of the L2CAP SDU transmission over the air. When signalled over the interface between L2CAP and the Link Manager, it is the maximum delay between the instant the first fragment of an L2CAP PDU is stored in the Host Controller buffer and the initial transmission of the L2CAP packet on the air.

  Thus the Access Latency value may be different when signalled between L2CAP and the Link Manager to account for any queuing delay at the L2CAP transmit side. Furthermore the Access Latency value may include the aggregation of multiple L2CAP channels onto the same ACL logical transport.

  The Access Latency is expressed in microseconds. The value 0xFFFFFFFF means “do not care” and is the default value. When the Guaranteed service is selected, the default value shall not be used.

- *Delay Variation (4 octets)*

  The value of this field is the difference, in microseconds, between the maximum and minimum possible delay of an L2CAP SDU between two L2CAP peers. The Delay Variation is a purely informational parameter. The value 0xFFFFFFFF means “do not care” and is the default value.

5.4 RETRANSMISSION AND FLOW CONTROL OPTION

This option specifies whether retransmission and flow control is used. If the feature is used incoming parameters are specified by this option.

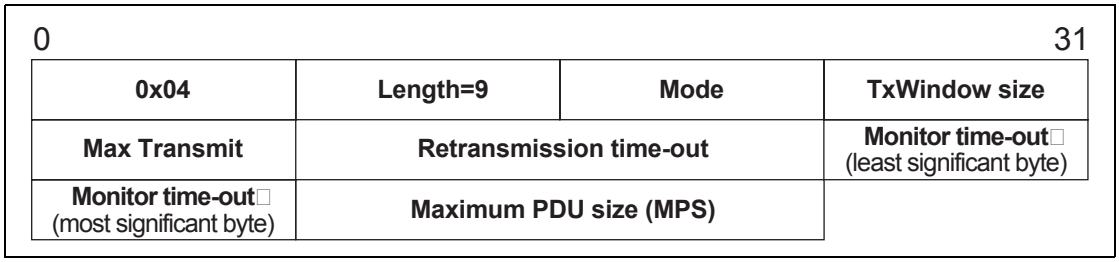


Figure 5.5: Retransmission and Flow Control option format.

The option data fields are:

- *Mode (1 octet)*

  The field contain the requested mode of the link. Possible values are shown in Table 5.2 on page 64.

Value	Description
0x00	Basic L2CAP mode
0x01	Retransmission Mode
0x02	Flow control mode
Other values	Reserved for future use

Table 5.2: Mode definitions.

  The Basic L2CAP mode is the default. If Basic L2CAP mode is requested then all other parameters shall be ignored.

  Retransmission mode should be enabled if a reliable channel has been requested, or if the L2CAP Flush Time-Out is long enough to contain the round-trip delay of a retransmission request.

- *TxWindow size (1 octet)*

  This field specifies the size of the transmission window for flow control mode and retransmission mode. The range is 1 to 32.

  This parameter should be negotiated to reflect the buffer sizes allocated for the connection on both sides. In general, the Tx Window size should be made as large as possible to maximize channel utilization. Tx Window size also controls the delay on flow control action. The transmitting device can send as many PDUs fit within the window.

- *MaxTransmit (1 octet)*

  This field controls the number of transmissions of a single I-frame that L2CAP is allowed to try in Retransmission mode. The minimum value is 1 (one transmission is permitted).

  MaxTransmit controls the number of retransmissions that L2CAP is allowed to try in Retransmission mode before accepting that a packet and the link is lost. Lower values might be appropriate for services requiring low latency. Higher values will be suitable for a link requiring robust operation. A value of 1 means that no retransmissions will be made but also means that the link will be dropped as soon as a packet is lost. MaxTransmit shall not be set to zero.

- *Retransmission time-out (2 octets)*

  This is the value in milliseconds of the retransmission time-out (this value is used to initialize the RetransmissionTimer, see below).

  The purpose of this timer in retransmission mode is to activate a retransmission in some exceptional cases. In such cases, any delay requirements on the channel may be broken, so the value of the timer should be set high enough to avoid unnecessary retransmissions due to delayed acknowledgements. Suitable values could be 100's of milliseconds and up.

  The purpose of this timer in flow control mode is to supervise I-frame transmissions. If an acknowledgement for an I-frame is not received within the time specified by the RetransmissionTimer value, either because the I-frame has been lost or the acknowledgement has been lost, the timeout will cause the transmitting side to continue transmissions. Suitable values are implementation dependent.

- *Monitor time-out (2 octets)*

  This is the value in milliseconds of the interval at which S-frames should be transmitted on the return channel when no frames are received on the forward channel. (this value is used to initialize the MonitorTimer, see below).

  This timer ensures that lost acknowledgements are retransmitted. Its main use is to recover Retransmission Disable Bit changes in lost frames when no data is being sent. The timer shall be started immediately upon transitioning to the open state. It shall remain active as long as the connection is in the open state and the retransmission timer is not active. Upon expiration of the Monitor timer an S-frame shall be sent and the timer shall be restarted. If the monitor timer is already active when an S-frame is sent, the timer shall be restarted. An idle connection will have periodic monitor traffic sent in both directions. The value for this time-out should also be set to 100's of milliseconds or higher.

- *Maximum PDU payload Size - MPS (2 octets)*

  The maximum size of payload data in octets that the L2CAP layer entity is capable of accepting, i.e. the MPS corresponds to the maximum PDU payload size.

The settings are configured separately for the two directions of an L2CAP connection. For example, an L2CAP connection can be configured as Flow Control

mode in one direction and Retransmission mode in the other direction. If Basic L2CAP mode is configured in one direction and Retransmission mode or Flow control mode is configured in the other direction on the same L2CAP channel then the channel shall not be used.
Note: this asymmetric configuration only occurs during configuration.

6 STATE MACHINE

This section is informative. The state machine may not represent all possible scenarios.

6.1 GENERAL RULES FOR THE STATE MACHINE:

- It is implementation specific, and outside the scope of this specification, how the transmissions are triggered.
- "Ignore" means that the signal can be silently discarded.

The following states have been defined to clarify the protocol; the actual number of states and naming in a given implementation is outside the scope of this specification:

- CLOSED – channel not connected.
- WAIT_CONNECT – a connection request has been received, but only a connection response with indication “pending” can be sent.
- WAIT_CONNECT_RSP – a connection request has been sent, pending a positive connect response.
- CONFIG – the different options are being negotiated for both sides; this state comprises a number of substates, see Section 6.1.3 on page 69
- OPEN – user data transfer state.
- WAIT_DISCONNECT – a disconnect request has been sent, pending a disconnect response.

Below the L2CAP_Data message corresponds to one of the PDU formats used on connection-oriented data channels as described in section 3, including PDUs containing B-frames, I-frames, S-frames.

Some state transitions and actions are triggered only by internal events effecting one of the L2CAP entity implementations, not by preceding L2CAP signalling messages. It is implementation-specific and out of the scope of this specification, how these internal events are realized; just for the clarity of specifying the state machine, the following abstract internal events are used in the state event tables, as far as needed:

- *OpenChannel_Req* – a local L2CAP entity is requested to set up a new connection-oriented channel.
- *OpenChannel_Rsp* – a local L2CAP entity is requested to finally accept or refuse a pending connection request.
- *ConfigureChannel_Req* – a local L2CAP entity is requested to initiate an outgoing configuration request.
- *CloseChannel_Req* – a local L2CAP entity is requested to close a channel.
- *SendData_Req* – a local L2CAP entity is requested to transmit an SDU.
- *ReconfigureChannel_Req* – a local L2CAP entity is requested to reconfigure the parameters of a connection-oriented channel.

There is a single state machine for each L2CAP connection-oriented channel that is active. A state machine is created for each new L2CAP_ConnectReq received. The state machine always starts in the CLOSED state.

To simplify the state event tables, the RTX and ERTX timers, as well as the handling of request retransmissions are described in Section 6.2 on page 75 and not included in the state tables.

L2CAP messages not bound to a specific data channel and thus not impacting a channel state (e.g. L2CAP_InformationReq, L2CAP_EchoReq) are not covered in this section.

The following states and transitions are illustrated in Figure 6.1 on page 77.

6.1.1 CLOSED state

Event	Condition	Action	Next State
OpenChannel_req	-	Send L2CAP_ConnectReq	WAIT_CONNECT_RSP
L2CAP_ConnectReq	Normal, connection is possible	Send L2CAP_ConnectRsp (success)	CONFIG (substate WAIT_CONFIG)
L2CAP_ConnectReq	Need to indicate pending	Send L2CAP_ConnectRsp (pending)	WAIT_CONNECT
L2CAP_ConnectReq	No resource, not approved, etc.	Send L2CAP_ConnectRsp (refused)	CLOSED
L2CAP_ConnectRsp	-	Ignore	CLOSED
L2CAP_ConfigReq	-	Send L2CAP_CommandReject (with reason Invalid CID)	CLOSED
L2CAP_ConfigRsp	-	Ignore	CLOSED
L2CAP_DisconnectReq	-	Send L2CAP_DisconnectRsp	CLOSED
L2CAP_DisconnectRsp	-	Ignore	CLOSED
L2CAP_Data	-	Ignore	CLOSED

Table 6.1: CLOSED state event table.

Notes:

- The L2CAP_ConnectReq message is not mentioned in any of the other states apart from the CLOSED state, as it triggers the establishment of a new channel, thus the branch into a new instance of the state machine.

6.1.2 WAIT_CONNECT_RSP state

Event	Condition	Action	Next State
L2CAP_ConnectRsp	Success indicated in result	Send L2CAP_ConfigReq	CONFIG (substate WAIT_CONFIG)
L2CAP_ConnectRsp	Result pending	-	WAIT_CONNECT_RSP
L2CAP_ConnectRsp	Remote side refuses connection	-	CLOSED
L2CAP_ConfigReq	-	Send L2CAP_CommandReject (with reason Invalid CID)	WAIT_CONNECT_RSP
L2CAP_ConfigRsp	-	Ignore	WAIT_CONNECT_RSP
L2CAP_DisconnectRsp	-	Ignore	WAIT_CONNECT_RSP
L2CAP_Data	-	Ignore	WAIT_CONNECT_RSP

Table 6.2: WAIT_CONNECT_RSP state event table.

Notes:

- An L2CAP_DisconnectReq message is not included here, since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.

6.1.3 WAIT_CONNECT state

Event	Condition	Action	Next State
OpenChannel_Rsp	Pending connection request is finally acceptable	Send L2CAP_Connect_Rsp (success)	CONFIG (substate WAIT_CONFIG)
OpenChannel_Rsp	Pending connection request is finally refused	Send L2CAP_Connect_Rsp (refused)	CLOSED
L2CAP_ConnectRsp	-	Ignore	WAIT_CONNECT
L2CAP_ConfigRsp	-	Ignore	WAIT_CONNECT
L2CAP_DisconnectRsp	-	Ignore	WAIT_CONNECT
L2CAP_Data	-	Ignore	WAIT_CONNECT

Table 6.3: WAIT_CONNECT state event table.

Notes:

- An L2CAP_DisconnectReq or L2CAP_ConfigReq message is not included here, since the Source and Destination CIDs are not available yet to relate it correctly to the state machine of a specific channel.

6.1.4 CONFIG state

As it is also described in Section 7.1 on page 79, both L2CAP entities initiate a configuration request during the configuration process. This means that each device adopts an initiator role for the outgoing configuration request, and an acceptor role for the incoming configuration request. Configurations in both directions may occur sequentially, but can also occur in parallel.

The following substates are distinguished within the CONFIG state:

- WAIT_CONFIG – a device has sent or received a connection response, but has neither initiated a configuration request yet, nor received a configuration request with acceptable parameters.
- WAIT_SEND_CONFIG – for the initiator path, a configuration request has not yet been initiated, while for the response path, a request with acceptable options has been received.
- WAIT_CONFIG_REQ_RSP – for the initiator path, a request has been sent but a positive response has not yet been received, and for the acceptor path, a request with acceptable options has not yet been received.
- WAIT_CONFIG_RSP – the acceptor path is complete after having responded to acceptable options, but for the initiator path, a positive response on the recent request has not yet been received.
- WAIT_CONFIG_REQ – the initiator path is complete after having received a positive response, but for the acceptor path, a request with acceptable options has not yet been received.

According to Section 6.1.1 on page 68 and Section 6.1.2 on page 69, the CONFIG state is entered via WAIT_CONFIG substate from either the CLOSED state, the WAIT_CONNECT state, or the WAIT_CONNECT_RSP state. The CONFIG state is left for the OPEN state if both the initiator and acceptor paths complete successfully.

For better overview, separate tables are given: Table 6.4 shows the success transitions; therein, transitions on one of the minimum paths (no previous non-success transitions) are shaded. Table 6.5 on page 71 shows the non-success transitions within the configuration process, and Table 6.6 on page 72 shows further transition cause by events not belonging to the configuration process itself. The following configuration states and transitions are illustrated in Figure 6.2 on page 78.

Previous state	Event	Condition	Action	Next State
WAIT_CONFIG	*ConfigureChannel_Req*	-	Send L2CAP_ConfigReq	WAIT_CONFIG_REQ_RSP
WAIT_CONFIG	L2CAP_ConfigReq	Options acceptable	Send L2CAP_ConfigRsp (success)	WAIT_SEND_CONFIG

Table 6.4: CONFIG state/substates event table: success transitions within configuration process.

Previous state	Event	Condition	Action	Next State
WAIT_CONFIG_REQ_RSP	L2CAP_ConfigReq	Options acceptable	Send L2CAP_ConfigRsp (success)	WAIT_CONFIG_RSP
WAIT_CONFIG_REQ_RSP	L2CAP_ConfigRsp	Remote side accepts options	- (continue waiting for configuration request)	WAIT_CONFIG_REQ
WAIT_CONFIG_REQ	L2CAP_ConfigReq	Options acceptable	Send L2CAP_ConfigRsp (success)	OPEN
WAIT_SEND_CONFIG	*ConfigureChannel_Req*	-	Send L2CAP_ConfigReq	WAIT_CONFIG_RSP
WAIT_CONFIG_RSP	L2CAP_ConfigRsp	Remote side accepts options	-	OPEN

Table 6.4: CONFIG state/substates event table: success transitions within configuration process.

Previous state	Event	Condition	Action	Next State
WAIT_CONFIG	L2CAP_ConfigReq	Options not acceptable	Send L2CAP_ConfigRsp (fail)	WAIT_CONFIG
WAIT_CONFIG	L2CAP_ConfigRsp	-	Ignore	WAIT_CONFIG
WAIT_SEND_CONFIG	L2CAP_ConfigRsp	-	Ignore	WAIT_SEND_CONFIG
WAIT_CONFIG_REQ_RSP	L2CAP_ConfigReq	Options not acceptable	Send L2CAP_ConfigRsp (fail)	WAIT_CONFIG_REQ_RSP
WAIT_CONFIG_REQ_RSP	L2CAP_ConfigRsp	Remote side rejects options	Send L2CAP_ConfigReq (new options)	WAIT_CONFIG_REQ_RSP
WAIT_CONFIG_REQ	L2CAP_ConfigReq	Options not acceptable	Send L2CAP_ConfigRsp (fail)	WAIT_CONFIG_REQ
WAIT_CONFIG_REQ	L2CAP_ConfigRsp	-	Ignore	WAIT_CONFIG_REQ
WAIT_CONFIG_RSP	L2CAP_ConfigRsp	Remote side rejects options	Send L2CAP_ConfigReq (new options)	WAIT_CONFIG_RSP

Table 6.5: CONFIG state/substates event table: non-success transitions within configuration process.

Previous state	Event	Condition	Action	Next State
CONFIG (any substate)	*CloseChannel_Req*	Any internal reason to stop	Send L2CAP_ Disconnect Req	WAIT_ DISCONNECT
CONFIG (any substate)	L2CAP_Disconnect Req	-	Send L2CAP_ Disconnect Rsp	CLOSED
CONFIG (any substate)	L2CAP_Disconnect Rsp	-	Ignore	CONFIG (remain in sub-state)
CONFIG (any substate)	L2CAP_Data	-	Process the PDU	CONFIG (remain in sub-state)

Table 6.6: CONFIG state/substates event table: events not related to configuration process.

Notes:

- Receiving data PDUs (L2CAP_Data) in CONFIG state should be relevant only in case of a transition to a reconfiguration procedure (from OPEN state). Discarding the received data is allowed only in Retransmission Mode. Discarding an S-frame is allowed but not recommended. If a S-frame is discarded, the monitor timer will cause a new S-frame to be sent after a time out.
- Indicating a failure in a configuration response does not necessarily imply a failure of the overall configuration procedure; instead, based on the information received in the negative response, a modified configuration request may be triggered.

6.1.5 OPEN state

Event	Condition	Action	Next State
SendData_req	-	Send L2CAP_Data packet according to configured mode	OPEN
ReconfigureChannel_req	-	Complete outgoing SDU Send L2CAP_ConfigReq	CONFIG (substate WAIT_CONFIG_RSP)
CloseChannel_req	-	Send L2CAP_DisconnectReq	WAIT_DISCONNECT
L2CAP_ConnectRsp	-	Ignore	OPEN
L2CAP_ConfigReq	Incoming config. options acceptable	Complete outgoing SDU Send L2CAP_ConfigRsp (ok)	CONFIG (substate WAIT_CONFIG_REQ)
L2CAP_ConfigReq	Incoming config. options not acceptable	Complete outgoing SDU Send L2CAP_ConfigRsp (fail)	OPEN
L2CAP_DisconnectReq	-	Send L2CAP_DisconnectRsp	CLOSED
L2CAP_DisconnectRsp	-	Ignore	OPEN
L2CAP_Data	-	Process the PDU	OPEN

Table 6.7: OPEN state event table.

Note: The outgoing SDU shall be completed from the view of the remote entity. Therefore all PDUs forming the SDU shall have been reliably transmitted by the local entity and acknowledged by the remote entity, before entering the configuration state.

6.1.6 WAIT_DISCONNECT state

Event	Condition	Action	Next State
L2CAP_ConnectRsp	-	Ignore	WAIT_DISCONNECT
L2CAP_ConfigReq	-	Send L2CAP_CommandReject with reason Invalid CID	WAIT_DISCONNECT
L2CAP_ConfigRsp	-	Ignore	WAIT_DISCONNECT
L2CAP_DisconnectReq	-	Send L2CAP_DisconnectRsp	CLOSED
L2CAP_DisconnectRsp	-	-	CLOSED

Table 6.8: WAIT_DISCONNECT state event table.

Event	Condition	Action	Next State
L2CAP_Data	-	Ignore	WAIT_DISCONNECT

Table 6.8: WAIT_DISCONNECT state event table.

6.2 TIMERS EVENTS

6.2.1 RTX

The Response Timeout eXpired (RTX) timer is used to terminate the channel when the remote endpoint is unresponsive to signalling requests. This timer is started when a signalling request (see Section 7 on page 79) is sent to the remote device. This timer is disabled when the response is received. If the initial timer expires, a duplicate Request message may be sent or the channel identified in the request may be disconnected. If a duplicate Request message is sent, the RTX timeout value shall be reset to a new value at least double the previous value. When retransmitting the Request message, the context of the same state shall be assumed as with the original transmission. If a Request message is received that is identified as a duplicate (retransmission), it shall be processed in the context of the same state which applied when the original Request message was received.

Implementations have the responsibility to decide on the maximum number of Request retransmissions performed at the L2CAP level before terminating the channel identified by the Requests. The one exception is the signalling CID that should never be terminated. The decision should be based on the flush timeout of the signalling link. The longer the flush timeout, the more retransmissions may be performed at the physical layer and the reliability of the channel improves, requiring fewer retransmissions at the L2CAP level.

For example, if the flush timeout is infinite, no retransmissions should be performed at the L2CAP level. When terminating the channel, it is not necessary to send a L2CAP_DisconnectReq and enter WAIT_DISCONNECT state. Channels can be transitioned directly to the CLOSED state.

The value of this timer is implementation-dependent but the minimum initial value is 1 second and the maximum initial value is 60 seconds. One RTX timer shall exist for each outstanding signalling request, including each Echo Request. The timer disappears on the final expiration, when the response is received, or the physical link is lost. The maximum elapsed time between the initial start of this timer and the initiation of channel termination (if no response is received) is 60 seconds.

6.2.2 ERTX

The Extended Response Timeout eXpired (ERTX) timer is used in place of the RTX timer when it is suspected the remote endpoint is performing additional processing of a request signal. This timer is started when the remote endpoint responds that a request is pending, e.g., when an L2CAP_ConnectRsp event with a "connect pending" result (0x0001) is received. This timer is disabled when the formal response is received or the physical link is lost. If the initial timer expires, a duplicate Request may be sent or the channel may be disconnected.

If a duplicate Request is sent, the particular ERTX timer disappears, replaced by a new RTX timer and the whole timing procedure restarts as described previously for the RTX timer.

The value of this timer is implementation-dependent but the minimum initial value is 60 seconds and the maximum initial value is 300 seconds. Similar to RTX, there MUST be at least one ERTX timer for each outstanding request that received a Pending response. There should be at most one (RTX or ERTX) associated with each outstanding request. The maximum elapsed time between the initial start of this timer and the initiation of channel termination (if no response is received) is 300 seconds. When terminating the channel, it is not necessary to send a L2CAP_DisconnectReq and enter WAIT_DISCONNECT state. Channels should be transitioned directly to the CLOSED state.

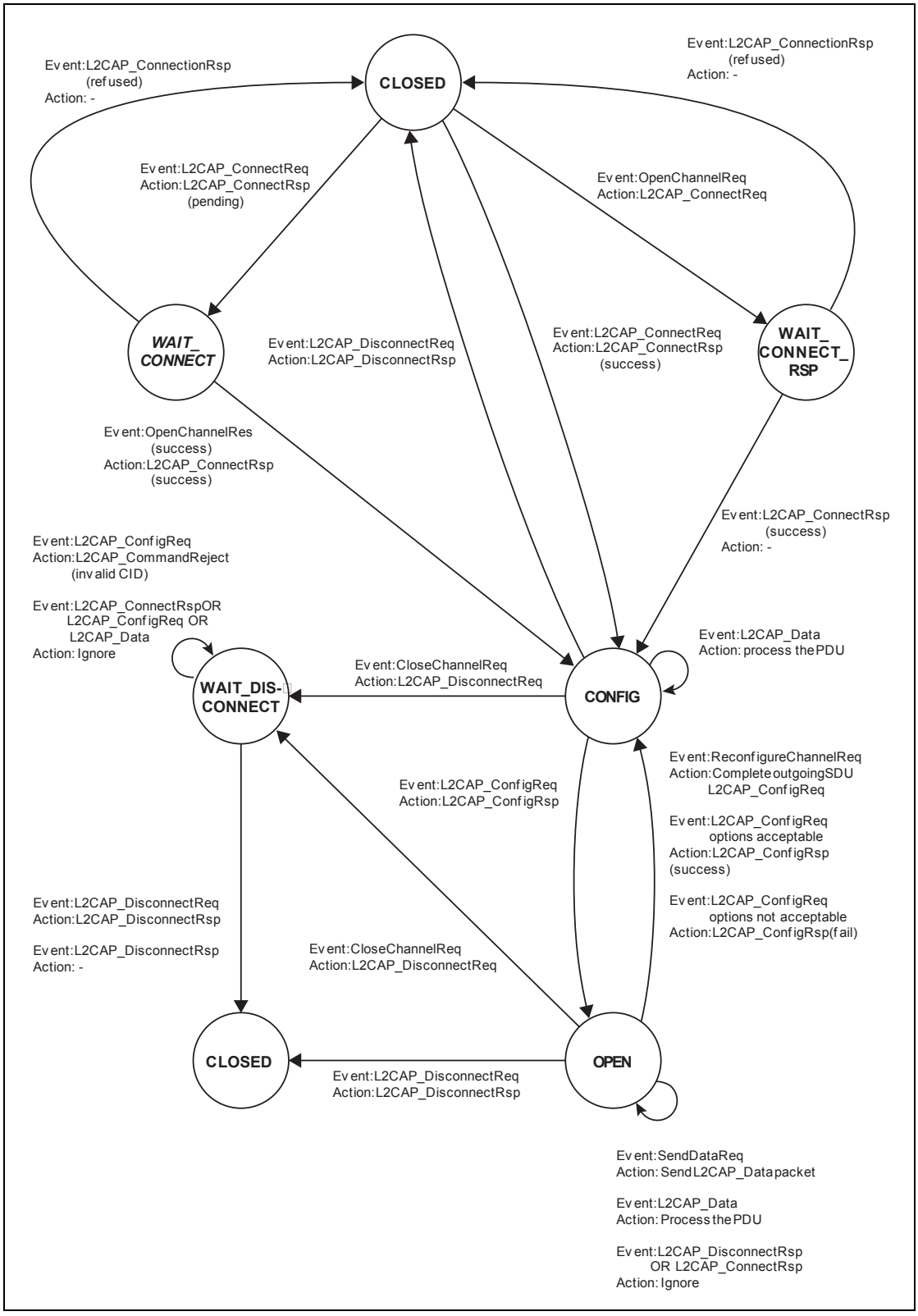


Figure 6.1: States and transitions.

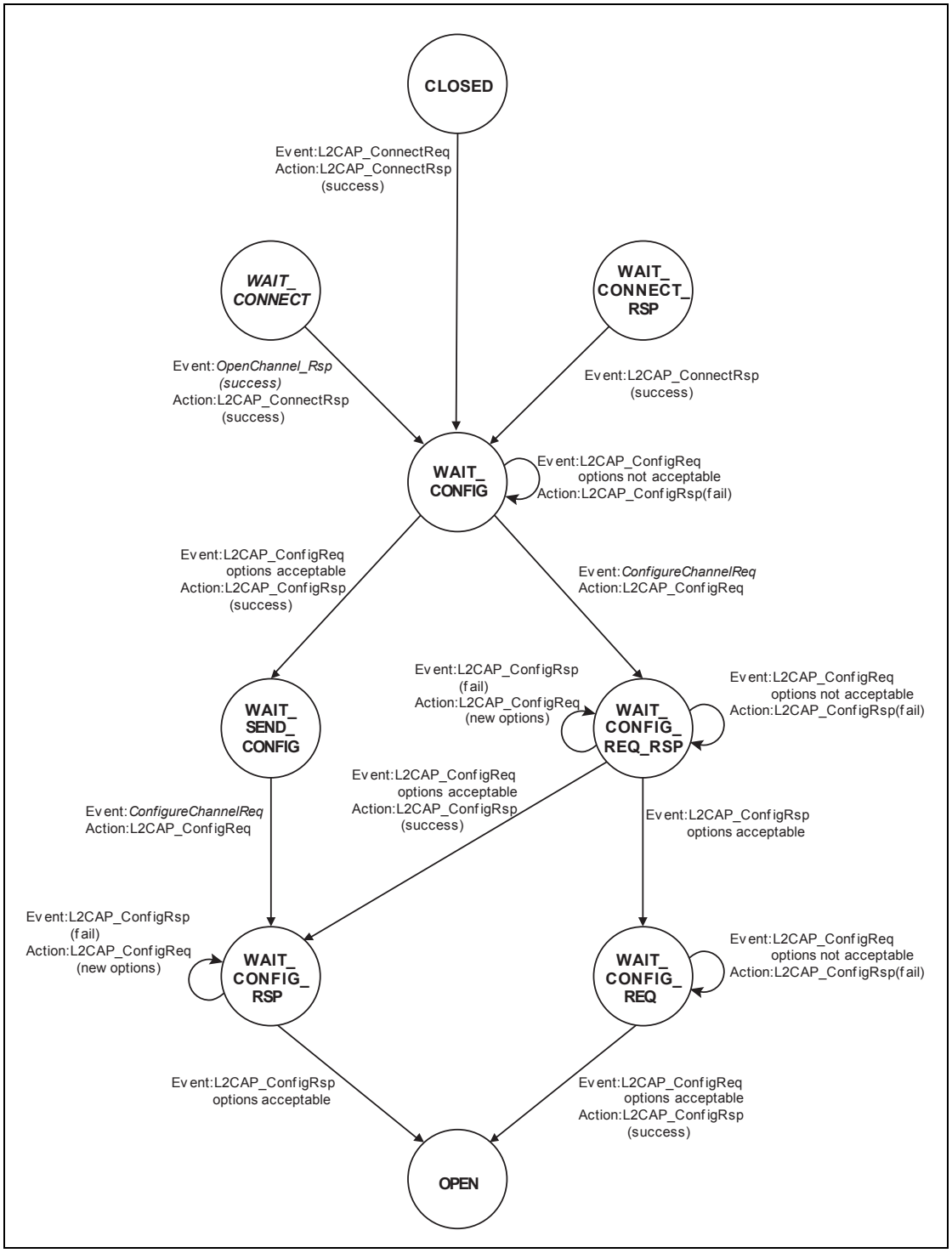


Figure 6.2: Configuration states and transitions.

7 GENERAL PROCEDURES

This section describes the general operation of L2CAP, including the configuration process, the handling and the processing of user data for transportation over the air interface. This section also describes the operation of L2CAP features including the delivery of erroneous packets, the flushing of expired data and operation in connectionless mode.

Procedures for the flow control and retransmission modes are described in Section 8 on page 87.

7.1 CONFIGURATION PROCESS

Configuring the channel parameters shall be done independently for both directions. Both configurations may be done in parallel. For each direction the following procedure shall be used:

1. Informing the remote side of the non-default parameters that the local side will accept using a Configuration Request
2. Remote side responds, agreeing or disagreeing with these values, including the default ones, using a Configuration Response.
3. The local and remote devices repeat steps (1) and (2) until agreement on all parameters is reached.

This process can be abstracted into the initial Request configuration path and a Response configuration path, followed by the reverse direction phase. Reconfiguration follows a similar two-phase process by requiring configuration in both directions.

The decision on the amount of time (or messages) spent configuring the channel parameters before terminating the configuration is left to the implementation, but it shall not last more than 120 seconds.

7.1.1 Request path

The Request Path can configure the following:

- requester's incoming MTU.
- requester's outgoing flush timeout.
- requester's outgoing QoS.
- requester's incoming flow and error control information.

Table 7.1 on page 80 defines the configuration options that may be placed in a Configuration Request.

Parameter	Description
MTU	Incoming MTU information
FlushTO	Outgoing flush timeout
QoS	Outgoing QoS information
RFCMode	Incoming Retransmission and Flow Control Mode

Table 7.1: Parameters allowed in Request

The state machine for the configuration process is described in Section 6 on page 67.

7.1.2 Response path

The Response Path can configure the following:

- responder's outgoing MTU, that is the remote side's incoming MTU.
- remote side's flush timeout.
- responder's incoming QoS Flow Specification (remote side's outgoing QoS Flow Specification).
- responder's Outgoing Flow and Error Control information

If a request-oriented parameter is not present in the Request message (reverts to last agreed value), the remote side may negotiate for a non-default value by including the proposed value in a negative Response message.

Parameter	Description
MTU	Outgoing MTU information
FlushTO	Incoming flush timeout
QoS	Incoming QoS information
RFCMode	Outgoing Retransmission and Flow Control Mode

Table 7.2: Parameters allowed in Response

7.2 FRAGMENTATION AND RECOMBINATION

Fragmentation is the breaking down of PDUs into smaller pieces for delivery from L2CAP to the lower layer. Recombination is the process of reassembling a PDU from fragments delivered up from the lower layer. Fragmentation and Recombination may be applied to any L2CAP PDUs.

7.2.1 Fragmentation of L2CAP PDUs

An L2CAP implementation may fragment any L2CAP PDU for delivery to the lower layers. If L2CAP runs directly over the link controller protocol, then an implementation may fragment the PDU into multiple baseband packets for transmission over the air. If L2CAP runs above the host controller interface, then an implementation may send HCI transport sized fragments to the Controller which passes them to the baseband. All L2CAP fragments associated with an L2CAP PDU shall be passed to the baseband before any other L2CAP PDU for the same logical transport shall be sent.

The two LLID bits defined in the first octet of baseband payload (also called the frame header) are used to signal the start and continuation of L2CAP PDUs. LLID shall be ’10’ for the first segment in an L2CAP PDU and ’01’ for a continuation segment. An illustration of fragmentation is shown in Figure 7.1 on page 81. An example of how fragmentation might be used in a device with HCI is shown in Figure 7.2 on page 82.

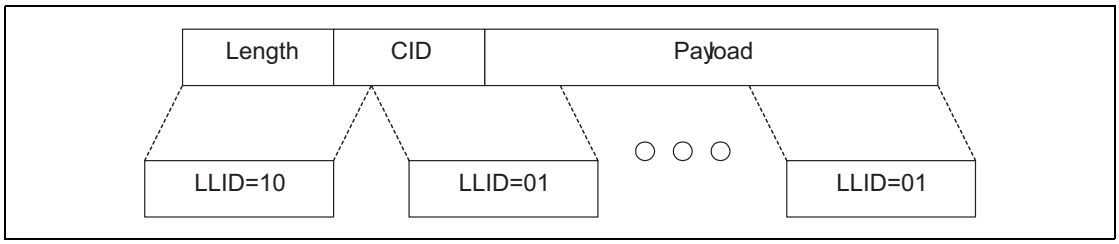


Figure 7.1: L2CAP fragmentation.

Note: The link controller is able to impose a different fragmentation on the PDU by using "start" and "continuation" indications as fragments are translated into baseband packets. Thus, both L2CAP and the link controller use the same mechanism to control the size of fragments.

7.2.2 Recombination of L2CAP PDUs

The link controller protocol attempts to deliver ACL packets in sequence and protects the integrity of the data using a 16-bit CRC. When errors are detected by the baseband it uses an automatic repeat request (ARQ) mechanism.

Recombination of fragments may occur in the Controller but ultimately it is the responsibility of L2CAP to reassemble PDUs and SDUs and to check the length field of the SDUs. As the baseband controller receives ACL packets, it either signals the L2CAP layer on the arrival of each baseband packet, or accumulates a number of packets (before the receive buffer fills up or a timer expires) before passing fragments to the L2CAP layer.

An L2CAP implementation shall use the length field in the header of L2CAP PDUs, see Section 3 on page 33, as a consistency check and shall discard any L2CAP PDUs that fail to match the length field. If channel reliability is not needed, packets with invalid lengths may be silently discarded. For reliable channels, an L2CAP implementation shall indicate to the upper layer that the channel has become unreliable. Reliable channels are defined by having an infinite flush timeout value as specified in Section 5.2 on page 59. For higher data integrity L2CAP should be operated in the Retransmission Mode.

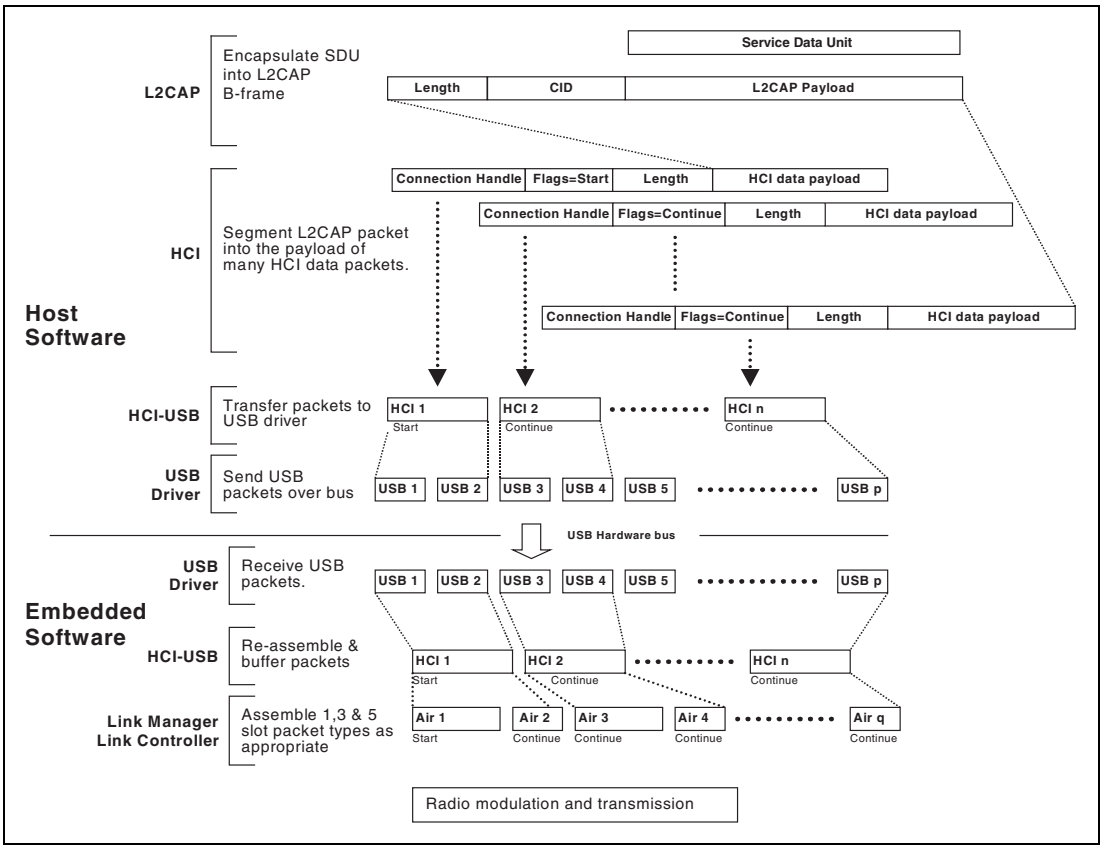


Figure 7.2: Example of fragmentation processes in a device with HCI.

7.3 ENCAPSULATION OF SDUs

All SDUs are encapsulated into one or more L2CAP PDUs.

In Basic L2CAP mode, an SDU shall be encapsulated with a minimum of L2CAP protocol elements, resulting in a type of L2CAP PDU called a Basic Information Frame (B-frame).

Segmentation and Reassembly operations are only used in Retransmission mode and Flow Control mode. SDUs may be segmented into a number of smaller packets called SDU segments. Each segment shall be encapsulated with L2CAP protocol elements resulting in an L2CAP PDU called an Information Frame (I-frame).

The maximum size of an SDU segment shall be given by the Maximum PDU Payload Size (MPS). The MPS parameter may be exported using an implementation specific interface to the upper layer.

Note that this specification does not have a normative service interface with the upper layer, nor does it assume any specific buffer management scheme of a host implementation. Consequently, a reassembly buffer may be part of the upper layer entity. It is assumed that SDU boundaries shall be preserved between peer upper layer entities.

7.3.1 Segmentation of L2CAP SDUs

In Flow Control or Retransmission modes, incoming SDUs may be broken down into segments, which shall then be individually encapsulated with L2CAP protocol elements (header and checksum fields) to form I-frames. I-frames are subject to flow control and may be subject to retransmission procedures. The header carries a 2 bit SAR field that is used to identify whether the I-frame is a 'start', 'end' or 'continuation' packet or whether it carries a complete, unsegmented SDU. Figure 7.3 on page 83 illustrates segmentation and fragmentation.

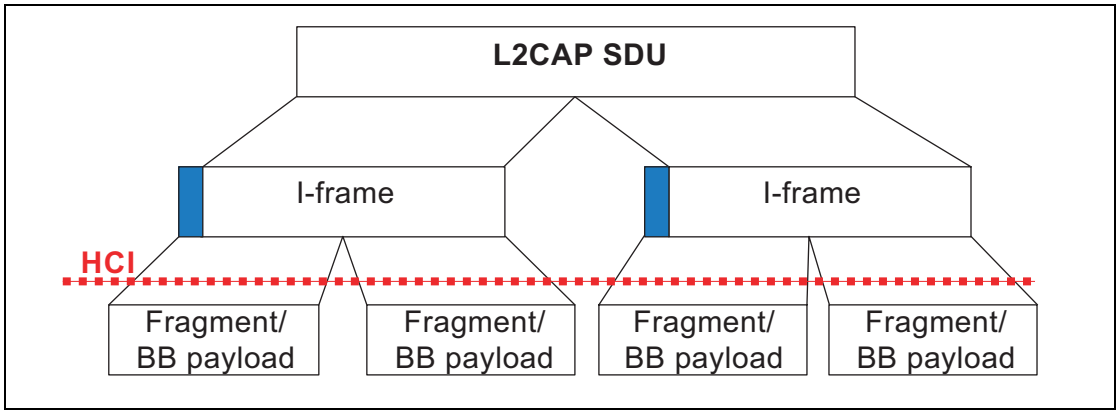


Figure 7.3: Segmentation and fragmentation of an SDU.

7.3.2 Reassembly of L2CAP SDUs

The receiving side uses the SAR field of incoming 'I-frames' for the reassembly process. The L2CAP SDU length field, present in the "start of SDU" I-frame, is an extra integrity check, and together with the sequence numbers may be used to indicate lost L2CAP SDUs to the application. Figure 7.3 on page 83 illustrates segmentation and fragmentation.

7.3.3 Segmentation and fragmentation

Figure 7.4 on page 84 illustrates the use of segmentation and fragmentation operations to transmit a single SDU. Note that while SDUs and L2CAP PDUs are transported in peer-to-peer fashion, the fragment size used by the Fragmentation and Recombination routines is implementation specific and may not be the same in the sender and the receiver. The over-the-air sequence of baseband packets as created by the sender is common to both devices.

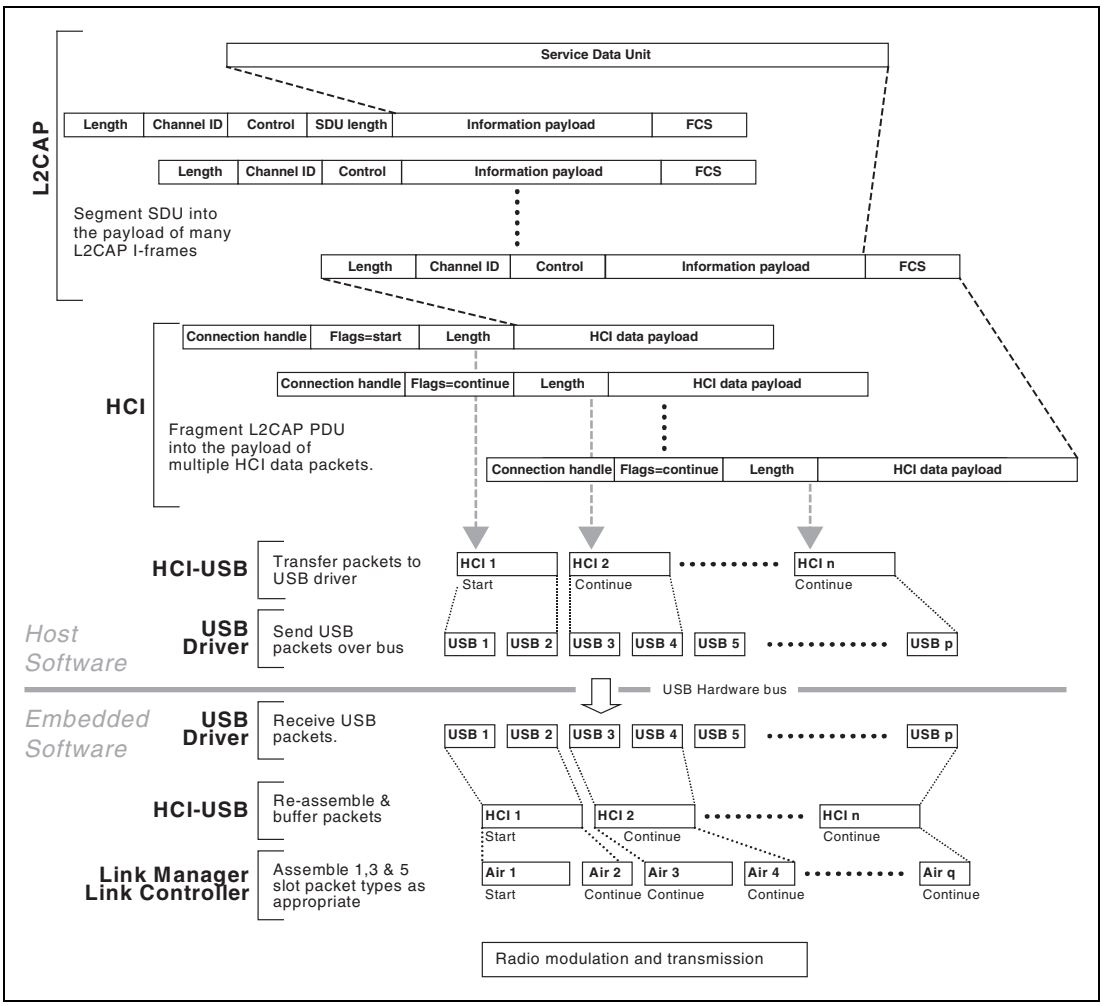


Figure 7.4: Example of segmentation and fragment processes in a device with HCI[1]

1. For simplicity, the stripping of any additional HCI and USB specific information fields prior to the creation of the baseband packets (Air_1, Air_2, etc.) is not shown in the figure.

7.4 DELIVERY OF ERRONEOUS L2CAP SDUS

Some applications may require corrupted or incomplete L2CAP SDUs to be delivered to the upper layer. If delivery of erroneous L2CAP SDUs is enabled, the receiving side will pass information to the upper layer on which parts of the L2CAP SDU (i.e., which L2CAP frames) have been lost, failed the error check, or passed the error check. If delivery of erroneous L2CAP SDUs is disabled, the receiver shall discard any L2CAP SDU segment with any missing frames or any frames failing the error checks. L2CAP SDUs whose length field does not match the actual frame length shall also be discarded.

7.5 OPERATION WITH FLUSHING

In the L2CAP configuration, the Flush Time-Out may be set separately per L2CAP channel, but there is only one flush mechanism per ACL logical transport in the baseband.

When there is more than one L2CAP channel mapped to the same ACL logical transport, the automatic flush time-out does not discriminate between L2CAP channels. The automatic flush time-out flushes a specific L2CAP PDU. The HCI Flush command flushes all outstanding L2CAP PDUs for the ACL logical transport. Therefore, care has to be taken when using the Automatic Flush Time-out and the HCI Flush command:

1. For any connection to be reliable at the L2CAP level, it should use L2CAP retransmission mode if it is mapped to an ACL logical transport with a finite automatic flush time-out. In retransmission mode, loss of flushed L2CAP PDUs on the channel is detected by the L2CAP ARQ mechanism and they are retransmitted.
2. There is only one automatic flush time-out setting per ACL logical transport. Therefore, all time bounded L2CAP channels on an ACL logical transport with a flush time-out setting should configure the same flush time-out value at the L2CAP level.
3. If Automatic Flush Time-out is used, then it should be taken into account that it only flushes one L2CAP PDU. If one PDU has timed out and needs flushing, then others on the same logical transport are also likely to need flushing. Therefore, when retransmission mode is used, flushing should be handled by the HCI Flush command so that all outstanding L2CAP PDUs are flushed.

7.6 CONNECTIONLESS DATA CHANNEL

In addition to connection-oriented channels, L2CAP also has a connectionless channel. The connectionless channel allows transmission to all members of the piconet. Data sent through the connectionless channel is sent in a best-effort manner. The connectionless channel has no quality of service and is unreliable. L2CAP makes no guarantee that data sent through the connectionless channel successfully reaches all members of the piconet. If reliable group transmission is required, it must be implemented at a higher layer.

Transmissions to the connectionless channel will be sent to all members of the piconet. If this data is not for transmission to all members of the piconet, then higher level encryption is required to support private communication.

The local device will not receive transmissions on the conectionless channel, therefore, higher layer protocols must loopback any data traffic being sent to the local device.

An L2CAP service interface could provide basic group management mechanisms including creating a group, adding members to a group, and removing members from a group.

Connectionless data channels shall not be used with Retransmission Mode or Flow Control Mode.

8 PROCEDURES FOR FLOW CONTROL AND RETRANSMISSION

When Flow Control mode or Retransmission mode is used, the procedures defined in this chapter shall be used. Including the numbering of information frames, the handling of SDU segmentation and reassembly, and the detection and notification of errored frames. Retransmission mode also allows the sender to resend errored frames on request from the receiver.

8.1 INFORMATION RETRIEVAL

Before attempting to configure flow control- or retransmission mode on a channel, it is mandatory to verify that the suggested mode is supported by performing an information retrieval for the "Extended features supported" information type (0x0002). If the information retrieval is not successful or the "Extended features mask" bit is not set for the wanted mode, the mode shall not be suggested in a configuration request.

8.2 FUNCTION OF PDU TYPES FOR FLOW CONTROL AND RETRANSMISSION

Two frame formats are defined for flow control and retransmission modes (see Section 3.3 on page 35). The I-frame is used to transport user information instead of the B-frame. The S-frame is used for signalling.

8.2.1 Information frame (I-frame)

I-frames are sequentially numbered frames containing information fields. I-frames also include the functionality of RR frames (see below).

8.2.2 Supervisory Frame (S-frame)

The S-frame is used to control the transmission of I-frames. The S-frame has two formats: Receiver Ready (RR) and Reject (REJ).

8.2.2.1 Receiver Ready (RR)

The receiver ready (RR) S-frame is used to:

1. Acknowledge I-frames numbered up to and including ReqSeq - 1.
2. Enable or disable retransmission of I-frames by updating the receiver with the current status of the Retransmission Disable Bit.

The RR frame has no information field.

8.2.2.2 Reject (REJ)

The reject (REJ) S-frame is used to request retransmission of all I-frames starting with the I-frame with TxSeq equal to ReqSeq specified in the REJ. The value of ReqSeq in the REJ frame acknowledges I-frames numbered up to and including ReqSeq - 1. I-frames that have not been transmitted, shall be transmitted following the retransmitted I-frames.

When a REJ is transmitted, it triggers a REJ Exception condition. A second REJ frame shall not be transmitted until the REJ Exception condition is cleared. The receipt of an I-frame with a TxSeq equal to the ReqSeq of the REJ frame clears the REJ Exception. The REJ Exception condition only applies to traffic in one direction. Note: this means that only valid I-frames can be rejected.

8.3 VARIABLES AND SEQUENCE NUMBERS

The sending peer uses the following variables and Sequence numbers:

- TxSeq – the send Sequence number used to sequentially number each new I-frame transmitted.
- NextTxSeq – the Sequence number to be used in the next new I-frame transmitted.
- ExpectedAckSeq – the Sequence number of the next I-frame expected to be acknowledged by the receiving peer.

The receiving peer uses the following variables and Sequence numbers:

- ReqSeq – The Sequence number sent in an acknowledgement frame to request transmission of I-frame with TxSeq = ReqSeq and acknowledge receipt of I-frames up to and including (ReqSeq-1)
- ExpectedTxSeq – the value of TxSeq expected in the next I-frame.
- BufferSeq – When segmented I-frames are buffered this is used to delay acknowledgement of received I-frame so that new I-frame transmissions do not cause buffer overflow.

All variables have the range 0 to 63. Arithmetic operations on state variables (NextTXSeq, ExpectedTxSeq, ExpectedAckSeq, BufferSeq) and sequence numbers (TxSeq, ReqSeq) contained in this document shall be taken modulo 64.

To perform Modulo 64 operation on negative numbers multiples of 64 shall be added to the negative number until the result becomes non-negative.

8.3.1 Sending peer

8.3.1.1 Send sequence number TxSeq

I-frames contain TxSeq, the send sequence number of the I-frame. When an I-frame is first transmitted, TxSeq is set to the value of the send state variable NextTXSeq. TxSeq is not changed if the I-frame is retransmitted.

8.3.1.2 Send state variable NextTXSeq

The CID sent in the information frame is the destination CID and identifies the remote endpoint of the channel. A send state variable NextTxSeq shall be maintained for each remote endpoint. NextTxSeq is the sequence number of the next in-sequence I-frame to be transmitted to that remote endpoint. When the link is created NextTXSeq shall be initialized to 0.

The value of NextTxSeq shall be incremented by 1 after each in-sequence I-frame transmission, and shall not exceed ExpectedAckSeq by more than the maximum number of outstanding I-frames (TxWindow). The value of TxWindow shall be in the range 1 to 32.

8.3.1.3 Acknowledge state variable ExpectedAckSeq

The CID sent in the information frame is the destination CID and identifies the remote endpoint of the channel. An acknowledge state variable ExpectedAck-Seq shall be maintained for each remote endpoint. ExpectedAckSeq is the sequence number of the next in-sequence I-frame that the remote receiving peer is expected to acknowledge. (ExpectedAckSeq – 1 equals the TxSeq of the last acknowledged I-frame). When the link is created ExpectedAckSeq shall be initialized to 0.
Note that if the next acknowledgement acknowledges a single I-frame then it's ReqSeq will be expectedAckSeq + 1.

If a valid ReqSeq is received from the peer then ExpectedAckSeq is set to ReqSeq. A valid ReqSeq value is one that is in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTxSeq.

Note: The comparison with NextTXSeq must be ≤ in order to handle the situations where there are no outstanding I-frames.

These inequalities shall be interpreted in the following way: ReqSeq is valid, if and only if (ReqSeq-ExpectedAckSeq) mod 64 ≤ (NextTXSeq-ExpectedAck-Seq) mod 64. Furthermore, from the description of NextTXSeq, it can be seen that (NextTXSeq-ExpectedAckSeq) mod 64 ≤ TxWindow.

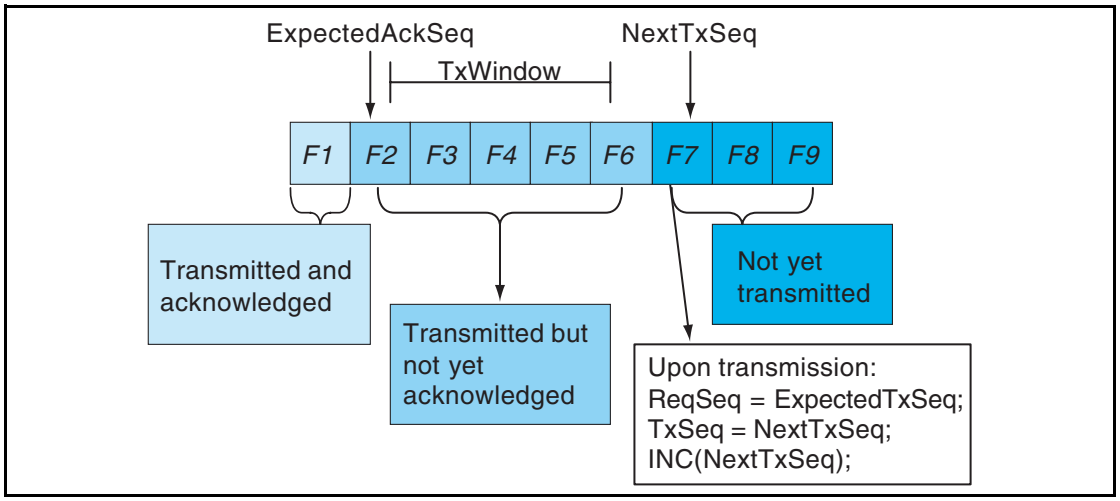


Figure 8.1: Example of the transmitter side

Figure 8.1 on page 90 shows TxWindow=5, and three frames awaiting transmission. The frame with number *F7* may be transmitted when the frame with *F2* is acknowledged. When the frame with *F7* is transmitted, TxSeq is set to the value of NextTXSeq. After TxSeq has been set, NextTxSeq is incremented.

The sending peer expects to receive legal ReqSeq values, these are in the range ExpectedAckSeq up to and including NextTxSeq. Upon receipt of a ReqSeq value equal to the current NextTxSeq all outstanding I-frames have been acknowledged by the receiver.

8.3.2 Receiving peer

8.3.2.1 Receive sequence number ReqSeq

All I-frames and S-frames contain ReqSeq, the send Sequence number (TxSeq) that the receiving peer requests in the next I-frame.

When an I-frame or an S-frame is transmitted, the value of ReqSeq shall be set to the current value of the receive state variable ExpectedTxSeq or the buffer state variable BufferSeq. The value of ReqSeq shall indicate that the data link layer entity transmitting the ReqSeq has correctly received all I-frames numbered up to and including ReqSeq – 1.

Note: The option to set ReqSeq to BufferSeq instead of ExpectedTxSeq allows the receiver to impose flow control for buffer management or other purposes. In this situation, if BufferSeq<>ExpectedTxSeq, the receiver should also set the retransmission disable bit to 1 to prevent unnecessary retransmissions.

8.3.2.2 Receive state variable, ExpectedTxSeq

Each channel shall have a receive state variable (ExpectedTxSeq). The receive state variable is the sequence number (TxSeq) of the next in-sequence I-frame expected.

The value of the receive state variable shall be the last in-sequence, valid I-frame received.

8.3.2.3 Buffer state variable BufferSeq

Each channel may have an associated BufferSeq. BufferSeq is used to delay acknowledgement of frames until they have been pulled by the upper layers, thus preventing buffer overflow. BufferSeq and ExpectedTxSeq are equal when there is no extra segmentation performed and frames are pushed to the upper layer immediately on reception. When buffer space is scarce, for example when frames reside in the buffer for a period, the receiver may choose to set ReqSeq to BufferSeq instead of ExpectedTxSeq, incrementing BufferSeq as buffer space is released. The windowing mechanism will ensure that transmission is halted when ExpectedTxSeq - BufferSeq is equal to TxWindow.

Note: Owing to the variable size of I-frames, updates of BufferSeq may be based on changes in available buffer space instead of delivery of I-frame contents.

I-Frames shall have sequence numbers in the range ExpectedTxSeq $\leq$ TxSeq $<$ (BufferSeq + TxWindow).

On receipt of an I-frame with TxSeq equal to ExpectedTxSeq, ExpectedTxSeq shall be incremented by 1 regardless of how many I-frames with TxSeq greater than ExpectedTxSeq were previously received.

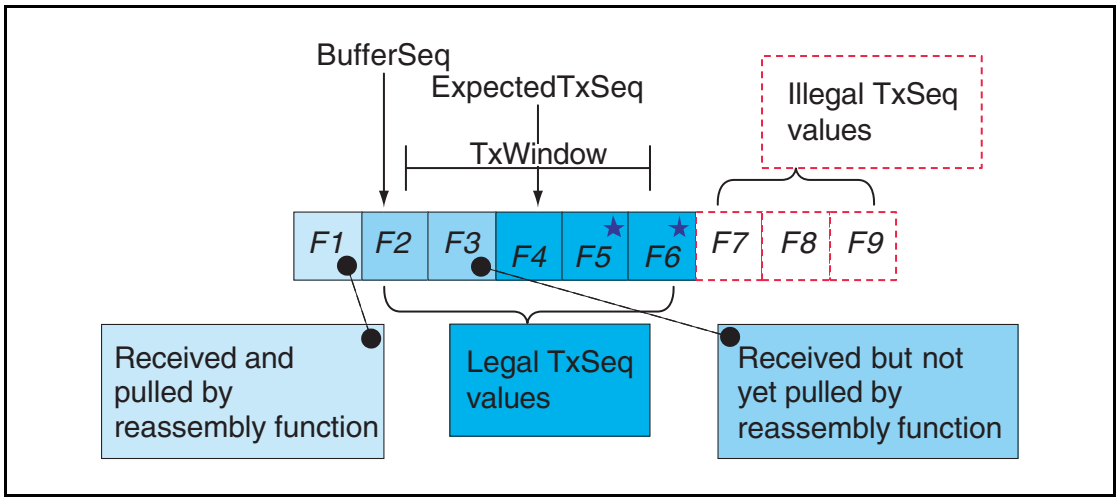


Figure 8.2: Example of the receiver side

Figure 8.2 on page 92 shows TxWindow=5. *F1* is successfully received and pulled by the upper layer. BufferSeq shows that *F2* is the next I-frame to be pulled, and ExpectedTxSeq points to the next I-frame expected from the peer. An I-frame with TxSeq equal to 5 has been received thus triggering an REJ exception. The star indicates I-frames received but discarded owing to the REJ exception. They will be resent as part of the error recovery procedure.

In Figure 8.2 on page 92 there are several I-frames in a buffer awaiting the SDU reassembly function to pull them and the TxWindow is full. The receiver would usually disable retransmission by setting the Retransmission Disable Bit to 1 and send an RR back to the sending side. This tells the transmitting peer that there is no point in performing retransmissions. Both sides will send S-frames to make sure the peer entity knows the current value of the Retransmission Disable Bit.

8.4 RETRANSMISSION MODE

8.4.1 Transmitting frames

A new I-frame shall only be transmitted when the TxWindow is not full. No I-frames shall be transmitted if the last RetransmissionDisableBit (R) received is set to one.

A previously transmitted I-frame may be retransmitted as a result of an error recovery procedure, even if the TxWindow is full. When an I-frame is retransmitted it shall always be sent with the same TxSeq value used in its initial transmission.

The state of the RetransmissionDisableBit (R) is stored and used along with the state of the RetransmissionTimer to decide the actions when transmitting I-frames. The RetransmissionTimer is running whenever I-frames have been sent but not acknowledged.

8.4.1.1 Last received R was set to zero

If the last R received was set to zero, then I-frames may be transmitted. If there are any I-frames which have been sent and not acknowledged then they shall be retransmitted when the RetransmissionTimer elapses. If the retransmission timer has not elapsed then a retransmission shall not be sent and only new I-frames may be sent.

a) If unacknowledged I-frames have been sent and the RetransmissionTimer has elapsed then an unacknowledged I-frame shall be retransmitted. The RetransmissionTimer shall be restarted.

b) If unacknowledged I-frames have been sent, the Retransmission timer has not elapsed then a new I-frame shall be sent if one is waiting and no timer action shall be taken.

c) If no unacknowledged I-frames have been sent, and a new I-frame is waiting, then the new I-frame shall be sent, the RetransmissionTimer shall be started and if the MonitorTimer is running, it shall be stopped.

d) If no unacknowledged I-frames have been sent and no new I-frames are waiting to be transmitted, and the RetransmissionTimer is running, then the retransmission timer shall be stopped and the monitor timer shall be started.

The table below summarizes actions when the RetransmissionTimer is in use and R=0.

Unacknowledged I-frames sent = Retransmission Timer is running	**Retrans-mission Timer has elapsed**	**New I-frames are waiting**	**Transmit Action**	**Timer Action**
True	True	True or False	Retransmit un-acknowledged I-frame	Restart Retransmission Timer
True	False	True	Transmit new I-frame	No timer action
True	False	False	No transmit action	No Timer action
False	False	True	Transmit new I-frame	Restart Retransmis-sion Timer
False	False	False	No Transmit action	If MonitorTimer is not running then restart MonitorTimer

Table 8.1: Summary of actions when the RetransmissionTimer is in use and R=0.

If the RetransmissionTimer is not in use, no unacknowledged I-frames have been sent and no new I-frames are waiting to be transmitted

a) If the MonitorTimer is running and has not elapsed then no transmit action shall be taken and no timer action shall be taken.

b) If the MonitorTimer has elapsed then an S-frame shall be sent and the MonitorTimer shall be restarted.

If any I-frames become available for transmission then the MonitorTimer shall be stopped, the RetransmissionTimer shall be started and the rules for when the RetransmissionTimer is in use shall be applied.

When an I-frame is sent ReqSeq shall be set to ExpectedTxSeq, TxSeq shall be set to NextTxSeq and NextTxSeq shall be incremented by one.

8.4.1.2 Last received R was set to one

If the last R received was set to one, then I-frames shall not be transmitted. The only frames which may be sent are S-frames. An S-frame shall be sent according to the rules below:

a) If the MonitorTimer is running and has not elapsed then no transmit action shall be taken and no timer action shall be taken.

b) If the MonitorTimer has elapsed then an S-frame shall be sent and the MonitorTimer shall be restarted.

8.4.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be accepted for the SDU reassembly function. ExpectedTxSeq is used by the reassembly function.

The first valid I-frame received after an REJ was sent, with a TxSeq of the received I-frame equal to ReqSeq of the REJ, shall clear the REJ Exception condition.

The ReqSeq shall be processed according to Section 8.4.6 on page 97.

If a valid I-frame with TxSeq ≠ ExpectedTxSeq is received then an exception condition shall be triggered which is handled according to Section 8.4.7 on page 97.

8.4.3 I-frames pulled by the SDU reassembly function

When the L2CAP layer has removed one or more I-frames from the buffer, BufferSeq may be incremented in accordance with the amount of buffer space released. If BufferSeq is incremented, an acknowledgement shall be sent to the peer entity.

Note: Since the primary purpose of BufferSeq is to prevent buffer overflow, an implementation may choose to set BufferSeq in accordance with how many new incoming I-frames could be stored rather than how many have been removed.

The acknowledgement may either be an RR or an I-frame. The acknowledgement shall be sent to the peer L2CAP entity with ReqSeq equal to BufferSeq. When there are no I-frames buffered for pulling ExpectedTxSeq is equal to BufferSeq.

If the MonitorTimer is active then it shall be restarted to indicate that a signal has been sent to the peer L2CAP entity.

8.4.4 Sending and receiving acknowledgements

Either the MonitorTimer or the RetransmissionTimer shall be active while in Retransmission Mode. Both timers shall not be active concurrently.

8.4.4.1 Sending acknowledgements

Whenever an L2CAP entity transmits an I-frame or an S-frame, ReqSeq shall be set to ExpectedTxSeq or BufferSeq.

8.4.4.2 Receiving acknowledgements

On receipt of a valid S-frame or I-frame, the ReqSeq contained in the frame shall acknowledge previously transmitted I-frames. ReqSeq acknowledges I-frames with a TxSeq up to and including ReqSeq – 1.

The following rules shall be applied:

1. If the RetransmissionDisableBit changed value from 0 to 1 (stop retransmissions) then the receiving entity shall
    a) If the RetransmissionTimer is running then stop it and start the MonitorTimer.
    b) Store the state of the RetransmissionDisableBit received.
2. If the RetransmissionDisableBit changed value from 1 to 0 (start retransmissions) then the receiving entity shall
    a) Store the state of the RetransmissionDisableBit received.
    b) If there are any I-frames that have been sent but not acknowledged, then stop the MonitorTimer and start the RetransmissionTimer.
    c) Any buffered I-frames shall be transmitted according to Section 8.4.1 on page 93.
3. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame, and the RetransmissionDisableBit equals 1 (retransmissions stopped), then the receiving entity shall
    a) Follow the rules in Section 8.4.1 on page 93.
4. If any unacknowledged I-frames were acknowledged by the ReqSeq contained in the frame and the RetransmissionDisableBit equals 0 (retransmissions started) then the receiving entity shall
    a) If the RetransmissionTimer is running, then stop it.
    b) If any unacknowledged I-frames have been sent then the RetransmissionTimer shall be restarted.

c) Follow the rules in Section 8.4.1 on page 93.

d) If the RetransmissionTimer is not running and the MonitorTimer is not running, then start the MonitorTimer.

On receipt of a valid S-frame or I-frame the ReqSeq contained in the frame shall acknowledge previously transmitted I-frames. ExpectedAckSeq shall be set to ReqSeq to indicate that the I-frames with TxSeq up to and including (ReqSeq - 1) have been acknowledged.

8.4.5 Receiving REJ frames

Upon receipt of a valid REJ frame, where ReqSeq identifies an I-frame not yet acknowledged, the ReqSeq acknowledges I-frames with TxSeq up to and including ReqSeq - 1. Therefore the REJ acknowledges all I-frames before the I-frame it is rejecting.

ExpectedAckSeq shall be set equal to ReqSeq to mark I-frames up to and including ReqSeq - 1 as received.

NextTXSeq shall be set to ReqSeq to cause transmissions of I-frames to resume from the point where TxSeq equals ReqSeq.

If ReqSeq equals ExpectedAckSeq then the REJ frame shall be ignored.

8.4.6 Waiting acknowledgements

A counter, TransmitCounter, counts the number of times an L2CAP PDU has been transmitted. This shall be set to 1 after the first transmission. If the RetransmissionTimer expires the following actions shall be taken:

1. If the TransmitCounter is less than MaxTransmit then:

   a) Increment the TransmitCounter

   b) Retransmit the last unacknowledged I-frame, according to Section 8.4.1 on page 93.

2. If the TransmitCounter is equal to MaxTransmit this channel to the peer entity shall be assumed lost. The channel shall move to the CLOSED state and appropriate action shall be taken to report this to the upper layers.

8.4.7 Exception conditions

Exception conditions may occur as the result of physical layer errors or L2CAP procedural errors. The error recovery procedures which are available following the detection of an exception condition at the L2CAP layer in Retransmission Mode are defined in this section.

8.4.7.1 TxSeq Sequence error

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received which contains a TxSeq value which is not equal to the expected value, thus TxSeq is not equal to ExpectedTxSeq.

The TxSeq sequence error may be due to three different causes:

- *Duplicated I-frame*

  The duplicated I-frame is identified by a TxSeq in the range BufferSeq to ExpectedTxSeq – 1 (BufferSeq ≤TxSeq<ExpectedTxSeq). The ReqSeq and RetransmissionDisableBit shall be processed according to Section 8.4.4 on page 96. The Information field shall be discarded since it has already been received.

- *Out-of-sequence I-frame*

  The out-of-sequence I-frame is identified by a TxSeq within the legal range. The ReqSeq and RetransmissionDisableBit shall be processed according to Section 8.4.4 on page 96.

  A REJ exception is triggered, and an REJ frame with ReqSeq equal to ExpectedTxSeq shall be sent to initiate recovery. The received I-frame shall be discarded.

- *Invalid TxSeq*

  An invalid TxSeq value is a value that does not meet either of the above conditions. An I-frame with an invalid TxSeq is likely to have errors in the control field and shall be silently discarded.

8.4.7.2 ReqSeq Sequence error

An ReqSeq sequence error exception condition occurs in the transmitter when a valid S-frame or I-frame is received which contains an invalid ReqSeq value. An invalid ReqSeq is one that is not in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTxSeq.

The L2CAP entity shall close the channel as a consequence of an ReqSeq Sequence error.

8.4.7.3 Timer recovery error

If an L2CAP entity fails to receive an acknowledgement for the last I-frame sent, then it will not detect an out-of-sequence exception condition and therefore will not transmit an REJ frame.

The L2CAP entity that transmitted an unacknowledged I-frame shall, on the expiry of the RetransmissionTimer, take appropriate recovery action as defined in Section 8.4.6 on page 97.

8.4.7.4 Invalid frame

Any frame received which is invalid (as defined in Section 3.3.6 on page 40) shall be discarded, and no action shall be taken as a result of that frame.

8.5 FLOW CONTROL MODE

When a link is configured to work in flow control mode, the flow control operation is similar to the procedures in retransmission mode, but all operations dealing with CRC errors in received packets are not used. Therefore

- REJ frames shall not be used in Flow Control Mode.
- The RetransmissionDisableBit shall always be set to zero in the transmitter, and shall be ignored in the receiver.

The behavior of flow control mode is specified in this section.

Assuming that the TxWindow size is equal to the buffer space available in the receiver (counted in number of I-frames), in flow control mode the number of unacknowledged frames in the transmitter window is always less than or equal to the number of frames for which space is available in the receiver. Note that a missing frame still occupies a place in the window.

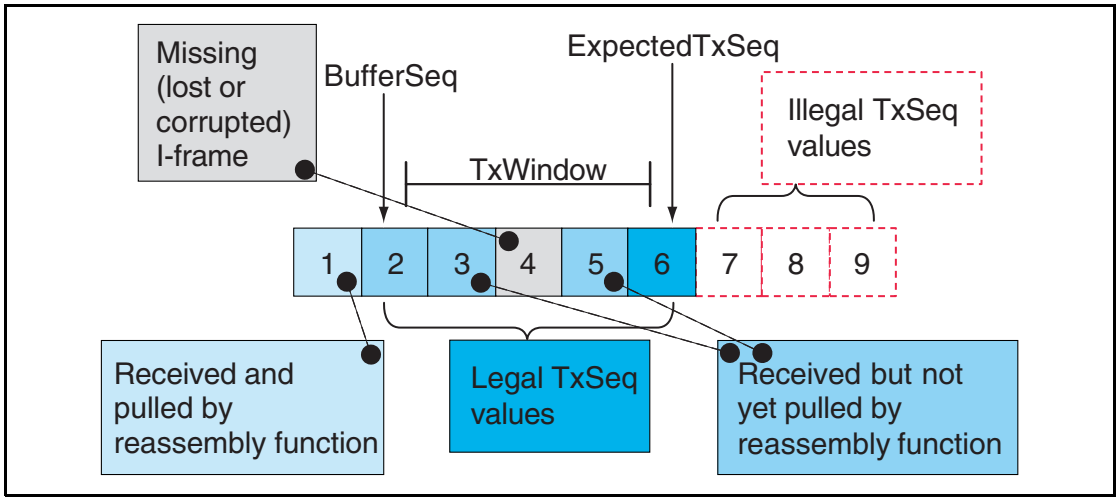


Figure 8.3: Overview of the receiver side when operating in flow control mode

8.5.1 Transmitting I-frames

A new I-frame shall only be transmitted when the TxWindow is not full.

Upon transmission of the I-frame the following actions shall be performed:

- If no unacknowledged I-frames have been sent then the MonitorTimer shall be stopped and the RetransmissionTimer shall be started.
- If any I-frames have been sent and not acknowledged then the RetransmissionTimer remains active and no timer operation is performed.

The control field parameter ReqSeq shall be set to ExpectedTxSeq, TxSeq shall be set to NextTXSeq and NextTXSeq shall be incremented by one.

8.5.2 Receiving I-frames

Upon receipt of a valid I-frame with TxSeq equal to ExpectedTxSeq, the frame shall be made available to the reassembly function. ExpectedTxSeq shall be incremented by one. An acknowledgement shall not be sent until the SDU reassembly function has pulled the I-frame.

Upon receipt of a valid I-frame with an out-of-sequence TxSeq (see Section 8.5.6 on page 101) all frames with a sequence number less than TxSeq shall be assumed lost and marked as missing. The missing I-frames are in the range from ExpectedTxSeq (the frame that the device was expecting to receive) up to TxSeq-1, (the frame that the device actually received). ExpectedTxSeq shall be set to TxSeq +1. The received I-frame shall be made available for pulling by the reassembly function. The acknowledgement shall not occur until the SDU reassembly function has pulled the I-frame. The ReqSeq shall be processed according to Section 8.5.4 on page 100.

8.5.3 I-frames pulled by the SDU reassembly function

When the L2CAP layer has removed one or more I-frames from the buffer, BufferSeq may be incremented in accordance with the amount of buffer space released. If BufferSeq is incremented, an acknowledgement shall be sent to the peer entity. If the MonitorTimer is active then it shall be restarted to indicate that a signal has been sent to the peer L2CAP entity.

Note: Since the primary purpose of BufferSeq is to prevent buffer overflow, an implementation may choose to set BufferSeq in accordance with how many new incoming I-frames could be stored rather than how many have been removed.

The acknowledgement may be an RR or an I-frame. The acknowledgement shall be sent to the peer L2CAP entity with ReqSeq equal to BufferSeq. When there is no I-frame buffered for pulling, ExpectedTxSeq is equal to BufferSeq.

8.5.4 Sending and receiving acknowledgements

One of the timers MonitorTimer or RetransmissionTimer shall always be active while in Flow Control mode. Both timers shall never be active concurrently.

8.5.4.1 Sending acknowledgements

Whenever a data link layer entity transmits an I-frame or a S-frame, ReqSeq shall be set to ExpectedTxSeq or BufferSeq.

8.5.4.2 Receiving acknowledgements

On receipt of a valid S-frame or I-frame the ReqSeq contained in the frame shall be used to acknowledge previously transmitted I-frames. ReqSeq acknowledges I-frames with a TxSeq up to and including ReqSeq – 1.

1. If any outstanding I-frames were acknowledged then
    a) Stop the RetransmissionTimer
    b) If there are still unacknowledged I-frames then restart the RetransmissionTimer, otherwise start the MonitorTimer.
    c) Transmit any I-frames awaiting transmission according to Section 8.5.1 on page 99.

ExpectedAckSeq shall be set to ReqSeq to indicate that the I-frames with TxSeq up to and including ExpectedAckSeq have been acknowledged.

8.5.5 Waiting acknowledgements

If the RetransmissionTimer expires the following actions shall be taken:
The I-frame supervised by the RetransmissionTimer shall be considered lost, and ExpectedAckSeq shall be incremented by one.

1. If I-frames are waiting to be sent
    a) The RetransmissionTimer is restarted
    b) I-frames awaiting transmission are transmitted according to Section 8.5.1 on page 99.
2. If there are no I-frames waiting to be sent
    a) If there are still unacknowledged I-frames the RetransmissionTimer is restarted, otherwise the MonitorTimer is started.

8.5.6 Exception conditions

Exception conditions may occur as the result of physical layer errors or L2CAP procedural errors. The error recovery procedures which are available following the detection of an exception condition at the L2CAP layer in flowcontrol only mode are defined in this section.

8.5.6.1 TxSeq Sequence error

A TxSeq sequence error exception condition occurs in the receiver when a valid I-frame is received which contains a TxSeq value which is not equal to the expected value, thus TxSeq is not equal to ExpectedTxSeq.

The TxSeq sequence error may be due to three different causes:

- *Duplicated I-frame*

  The duplicated I-frame is identified by a TxSeq in the range BufferSeq to ExpectedTxSeq – 1. The ReqSeq shall be processed according to Section 8.5.4 on page 100. The Information field shall be discarded since it has already been received.

- *Out-of-sequence I-frame*

  The out-of-sequence I-frame is identified by a TxSeq within the legal range ExpectedTxSeq < TxSeq < (BufferSeq + TxWindow). The ReqSeq shall be processed according to Section 8.5.4 on page 100.

  The missing I-frame(s) are considered lost and ExpectedTXSeq is set equal to TxSeq+1 as specified in Section 8.5.2 on page 100. The missing I-frame(s) are reported as lost to the SDU reassembly function.

- *Invalid TxSeq*

  An invalid TxSeq value is a value that does not meet either of the above conditions and TxSeq is not equal to ExpectedTxSeq. An I-frame with an invalid TxSeq is likely to have errors in the control field and shall be silently discarded.

8.5.6.2 ReqSeq Sequence error

An ReqSeq sequence error exception condition occurs in the transmitter when a valid S-frame or I-frame is received which contains an invalid ReqSeq value. An invalid ReqSeq is one that is not in the range ExpectedAckSeq ≤ ReqSeq ≤ NextTXSeq.

The L2CAP entity shall close the channel as a consequence of an ReqSeq Sequence error.

8.5.6.3 Timer recovery error

An L2CAP entity that fails to receive an acknowledgement for an I-frame shall, on the expiry of the RetransmissionTimer, take appropriate recovery action as defined in Section 8.5.5 on page 101.

8.5.6.4 Invalid frame

Any frame received that is invalid (as defined in Section 3.3.6 on page 40) shall be discarded, and no action shall be taken as a result of that frame, unless the receiving L2CAP entity is configured to deliver erroneous frames to the layer above L2CAP. In that case, the data contained in invalid frames may also be added to the receive buffer and made available for pulling from the SDU reassembly function.

9 LIST OF FIGURES

10 LIST OF TABLES

APPENDIX A: CONFIGURATION MSCs

The examples in this appendix describe a sample of the multiple possible configuration scenarios that might occur.

Figure I illustrates the basic configuration process. In this example, the devices exchange MTU information. All other values are assumed to be default.

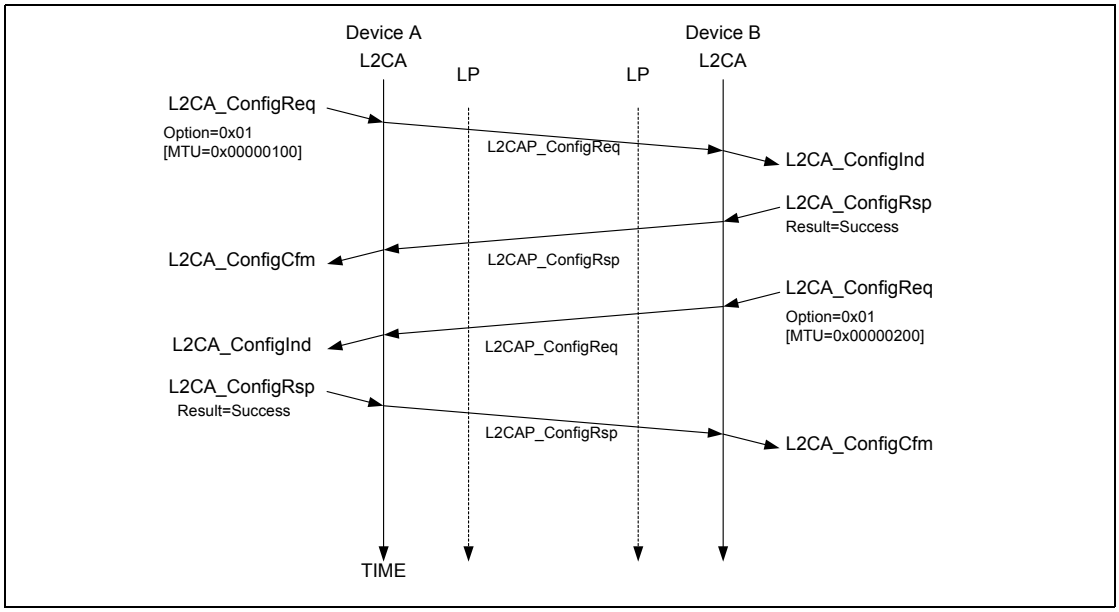


Figure I: Basic MTU exchange

Figure II on page 108 illustrates how two devices interoperate even though one device supports more options than the other does. Device A is an upgraded version. It uses a hypothetically defined option type 0x20 for link-level security. Device B rejects the command using the Configuration Response packet with result 'unknown parameter' informing Device A that option 0x20 is not understood. Device A then resends the request omitting option 0x20. Device B notices that it does not need to such a large MTU and accepts the request but includes in the response the MTU option informing Device A that Device B will not send an L2CAP packet with a payload larger than 0x80 octets over this channel. On receipt of the response, Device A could reduce the buffer allocated to hold incoming traffic.

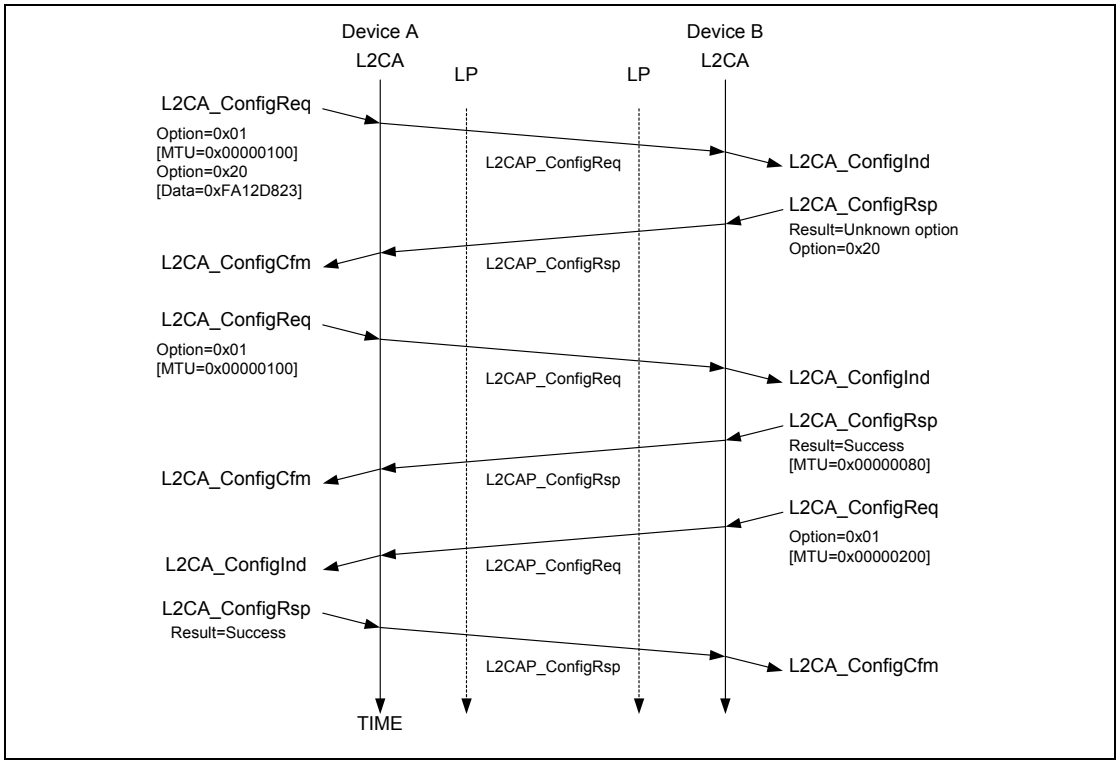


Figure II: Dealing with Unknown Options

Figure III on page 109 illustrates an unsuccessful configuration request. There are two problems described by this example. The first problem is that the configuration request is placed in an L2CAP packet that cannot be accepted by the remote device, due to its size. The remote device informs the sender of this problem using the Command Reject message. Device A then resends the configuration options using two smaller L2CAP_ConfigReq messages.

The second problem is an attempt to configure a channel with an invalid CID. For example device B may not have an open connection on that CID (0x01234567 in this example case).

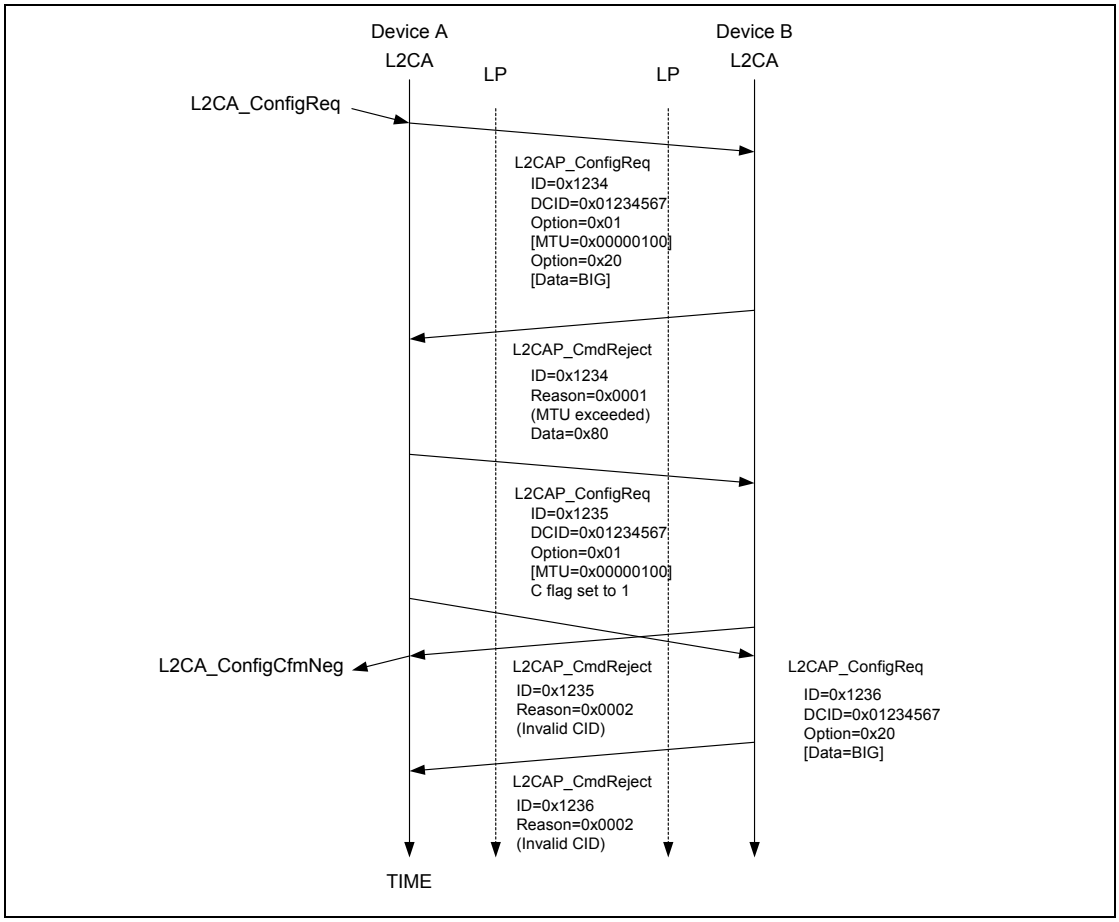


Figure III: Unsuccessful Configuration Request

Core System Package [Host volume]
Part B

SERVICE DISCOVERY PROTOCOL (SDP)

This specification defines a protocol for locating services provided by or available through a Bluetooth device.

CONTENTS

1 INTRODUCTION

1.1 GENERAL DESCRIPTION

The service discovery protocol (SDP) provides a means for applications to discover which services are available and to determine the characteristics of those available services.

1.2 MOTIVATION

Service Discovery in the Bluetooth environment, where the set of services that are available changes dynamically based on the RF proximity of devices in motion, is qualitatively different from service discovery in traditional network-based environments. The service discovery protocol defined in this specification is intended to address the unique characteristics of the Bluetooth environment. See Section , “Appendix A – Background Information,” on page 157, for further information on this topic.

1.3 REQUIREMENTS

The following capabilities have been identified as requirements for version 1.0 of the Service Discovery Protocol.

1. SDP shall provide the ability for clients to search for needed services based on specific attributes of those services.
2. SDP shall permit services to be discovered based on the class of service.
3. SDP shall enable browsing of services without a priori knowledge of the specific characteristics of those services.
4. SDP shall provide the means for the discovery of new services that become available when devices enter RF proximity with a client device as well as when a new service is made available on a device that is in RF proximity with the client device.
5. SDP shall provide a mechanism for determining when a service becomes unavailable when devices leave RF proximity with a client device as well as when a service is made unavailable on a device that is in RF proximity with the client device.
6. SDP shall provide for services, classes of services, and attributes of services to be uniquely identified.
7. SDP shall allow a client on one device to discover a service on another device without consulting a third device.
8. SDP should be suitable for use on devices of limited complexity.
9. SDP shall provide a mechanism to incrementally discover information about the services provided by a device. This is intended to minimize the quantity

of data that must be exchanged in order to determine that a particular service is not needed by a client.

10. SDP should support the caching of service discovery information by intermediary agents to improve the speed or efficiency of the discovery process.
11. SDP should be transport independent.
12. SDP shall function while using L2CAP as its transport protocol.
13. SDP shall permit the discovery and use of services that provide access to other service discovery protocols.
14. SDP shall support the creation and definition of new services without requiring registration with a central authority.

1.4 NON-REQUIREMENTS AND DEFERRED REQUIREMENTS

The Bluetooth SIG recognizes that the following capabilities are related to service discovery. These items are not addressed in SDP version 1.0. However, some may be addressed in future revisions of the specification.

1. SDP 1.0 does not provide access to services. It only provides access to information about services.
2. SDP 1.0 does not provide brokering of services.
3. SDP 1.0 does not provide for negotiation of service parameters.
4. SDP 1.0 does not provide for billing of service use.
5. SDP 1.0 does not provide the means for a client to control or change the operation of a service.
6. SDP 1.0 does not provide an event notification when services, or information about services, become unavailable.
7. SDP 1.0 does not provide an event notification when attributes of services are modified.
8. This specification does not define an application programming interface for SDP.
9. SDP 1.0 does not provide support for service agent functions such as service aggregation or service registration.

1.5 CONVENTIONS

1.5.1 Bit And Byte Ordering Conventions

When multiple bit fields are contained in a single byte and represented in a drawing in this specification, the more significant (high-order) bits are shown toward the left and less significant (low-order) bits toward the right.

Multiple-byte fields are drawn with the more significant bytes toward the left and the less significant bytes toward the right. Multiple-byte fields are transferred in network byte order. See section 4.1 on page 131.

2 OVERVIEW

2.1 SDP CLIENT-SERVER INTERACTION

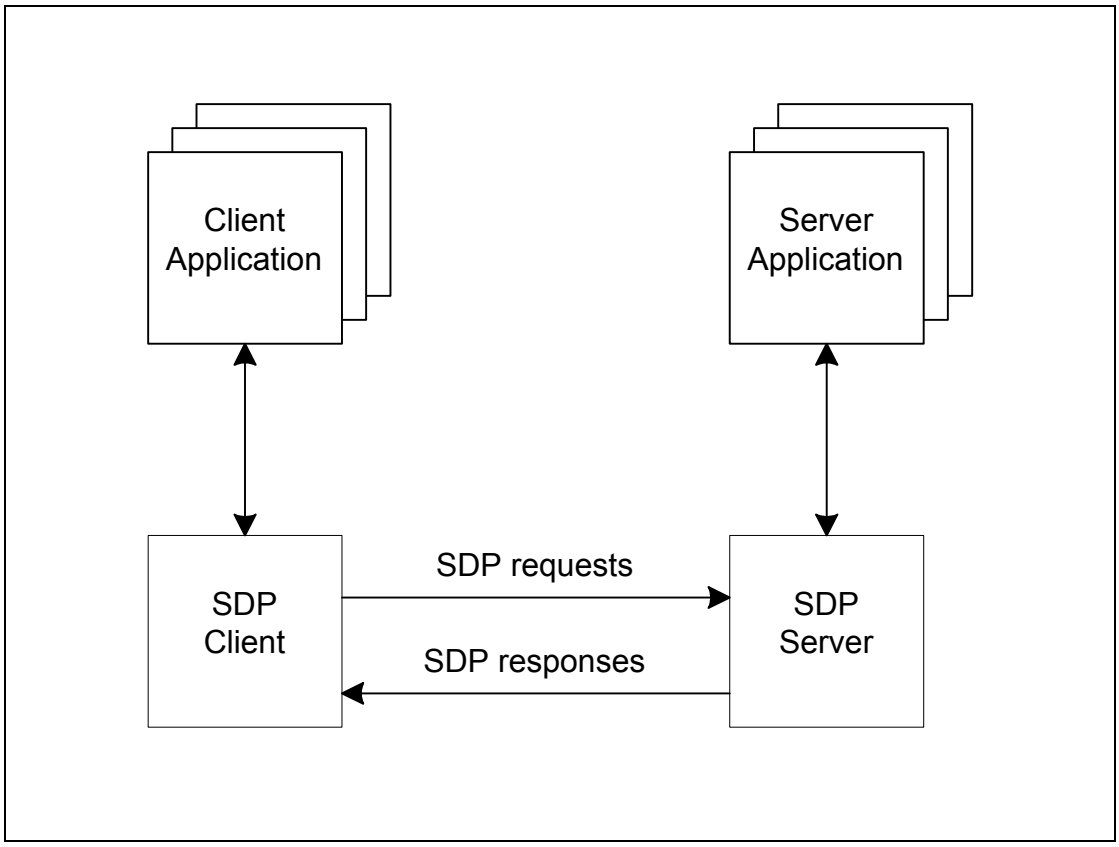


Figure 2.1:

The service discovery mechanism provides the means for client applications to discover the existence of services provided by server applications as well as the attributes of those services. The attributes of a service include the type or class of service offered and the mechanism or protocol information needed to utilize the service.

As far as the Service Discovery Protocol (SDP) is concerned, the configuration shown in may be simplified to that shown in Figure 2.1 may be simplified to that shown in Figure 2.2.

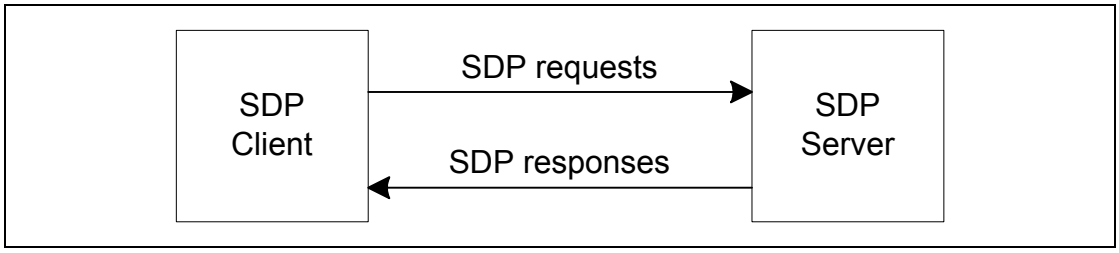


Figure 2.2:

SDP involves communication between an SDP server and an SDP client. The server maintains a list of service records that describe the characteristics of services associated with the server. Each service record contains information about a single service. A client may retrieve information from a service record maintained by the SDP server by issuing an SDP request.

If the client, or an application associated with the client, decides to use a service, it must open a separate connection to the service provider in order to utilize the service. SDP provides a mechanism for discovering services and their attributes (including associated service access protocols), but it does not provide a mechanism for utilizing those services (such as delivering the service access protocols).

There is a maximum of one SDP server per Bluetooth device. (If a Bluetooth device acts only as a client, it needs no SDP server.) A single Bluetooth device may function both as an SDP server and as an SDP client. If multiple applications on a device provide services, an SDP server may act on behalf of those service providers to handle requests for information about the services that they provide.

Similarly, multiple client applications may utilize an SDP client to query servers on behalf of the client applications.

The set of SDP servers that are available to an SDP client can change dynamically based on the RF proximity of the servers to the client. When a server becomes available, a potential client must be notified by a means other than SDP so that the client can use SDP to query the server about its services. Similarly, when a server leaves proximity or becomes unavailable for any reason, there is no explicit notification via the service discovery protocol. However, the client may use SDP to poll the server and may infer that the server is not available if it no longer responds to requests.

Additional information regarding application interaction with SDP is contained in the Bluetooth Service Discovery Profile document.

2.2 SERVICE RECORD

A service is any entity that can provide information, perform an action, or control a resource on behalf of another entity. A service may be implemented as software, hardware, or a combination of hardware and software.

All of the information about a service that is maintained by an SDP server is contained within a single service record. The service record consists entirely of a list of service attributes.

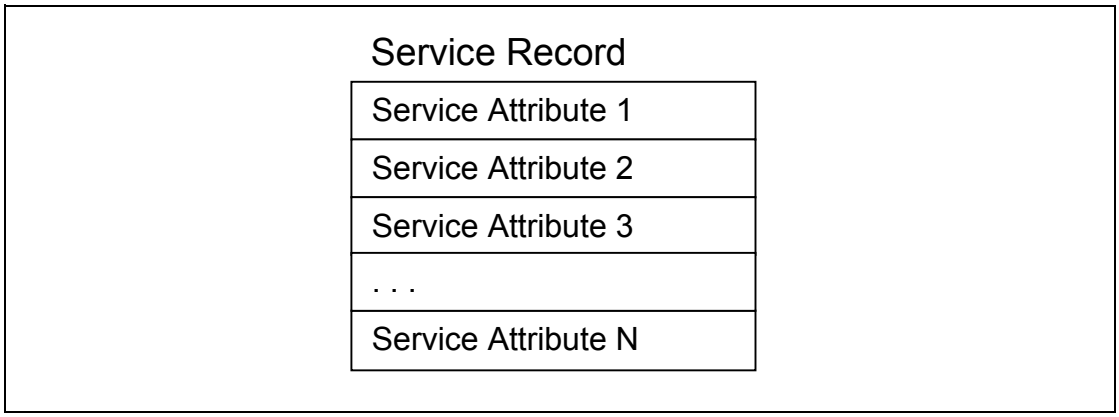


Figure 2.3: Service Record

A service record handle is a 32-bit number that uniquely identifies each service record within an SDP server. It is important to note that, in general, each handle is unique only within each SDP server. If SDP server S1 and SDP server S2 both contain identical service records (representing the same service), the service record handles used to reference these identical service records are completely independent. The handle used to reference the service on S1 will be meaningless if presented to S2.

The service discovery protocol does not provide a mechanism for notifying clients when service records are added to or removed from an SDP server. While an L2CAP (Logical Link Control and Adaptation Protocol) connection is established to a server, a service record handle acquired from the server will remain valid unless the service record it represents is removed. If a service is removed from the server, further requests to the server (during the L2CAP connection in which the service record handle was acquired) using the service's (now stale) record handle will result in an error response indicating an invalid service record handle. An SDP server must ensure that no service record handle values are re-used while an L2CAP connection remains established. Note that service record handles are known to remain valid across successive L2CAP connections while the ServiceDatabaseState attribute value remains unchanged. See the ServiceRecordState and ServiceDatabaseState attributes in section 5 on page 145.

There is one service record handle whose meaning is consistent across all SDP servers. This service record handle has the value 0x00000000 and is a handle to the service record that represents the SDP server itself. This service record contains attributes for the SDP server and the protocol it supports. For example, one of its attributes is the list of SDP protocol versions supported by the server. Service record handle values 0x00000001-0x0000FFFF are reserved.

2.3 SERVICE ATTRIBUTE

Each service attribute describes a single characteristic of a service. Some examples of service attributes are:

ServiceClassIDList	Identifies the type of service represented by a service record. In other words, the list of classes of which the service is an instance
ServiceID	Uniquely identifies a specific instance of a service
ProtocolDescriptorList	Specifies the protocol stack(s) that may be used to utilize a service
ProviderName	The textual name of the individual or organization that provides a service
IconURL	Specifies a URL that refers to an icon image that may be used to represent a service
ServiceName	A text string containing a human-readable name for the service
ServiceDescription	A text string describing the service

See section 5.1 on page 145, for attribute definitions that are common to all service records. Service providers can also define their own service attributes.

A service attribute consists of two components: an attribute ID and an attribute value.

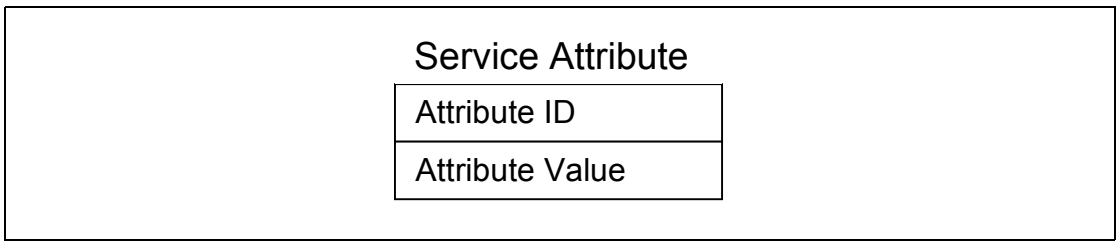


Figure 2.4: Service Attribute

2.4 ATTRIBUTE ID

An attribute ID is a 16-bit unsigned integer that distinguishes each service attribute from other service attributes within a service record. The attribute ID also identifies the semantics of the associated attribute value.

A service class definition specifies each of the attribute IDs for a service class and assigns a meaning to the attribute value associated with each attribute ID.

For example, assume that service class C specifies that the attribute value associated with attribute ID 12345 is a text string containing the date the service was created. Assume further that service A is an instance of service class C. If service A's service record contains a service attribute with an attribute ID

of 12345, the attribute value must be a text string containing the date that service A was created. However, services that are not instances of service class C may assign a different meaning to attribute ID 12345.

All services belonging to a given service class assign the same meaning to each particular attribute ID. See section 2.6 on page 121.

In the Service Discovery Protocol, an attribute ID is often represented as a data element. See section 3 on page 127.

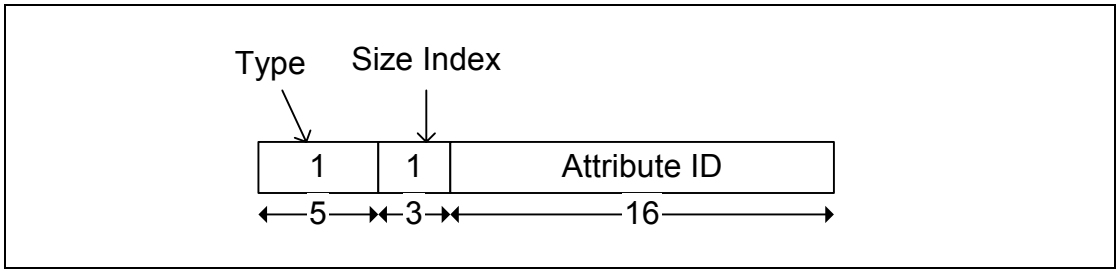


Figure 2.5:

2.5 ATTRIBUTE VALUE

The attribute value is a variable length field whose meaning is determined by the attribute ID associated with it and by the service class of the service record in which the attribute is contained. In the Service Discovery Protocol, an attribute value is represented as a data element. (See section 3 on page 127.) Generally, any type of data element is permitted as an attribute value, subject to the constraints specified in the service class definition that assigns an attribute ID to the attribute and assigns a meaning to the attribute value. See section 5 on page 145, for attribute value examples.

2.6 SERVICE CLASS

Each service is an instance of a service class. The service class definition provides the definitions of all attributes contained in service records that represent instances of that class. Each attribute definition specifies the numeric value of the attribute ID, the intended use of the attribute value, and the format of the attribute value. A service record contains attributes that are specific to a service class as well as universal attributes that are common to all services.

Each service class is also assigned a unique identifier. This service class identifier is contained in the attribute value for the ServiceClassIDList attribute, and is represented as a UUID (see section 2.7.1 on page 123). Since the format and meanings of many attributes in a service record are dependent on the service class of the service record, the ServiceClassIDList attribute is very important. Its value should be examined or verified before any class-specific attributes are used. Since all of the attributes in a service record must conform to all of the service's classes, the service class identifiers contained in the ServiceClassIDList attribute are related. Typically, each service class is a subclass

of another class whose identifier is contained in the list. A service subclass definition differs from its superclass in that the subclass contains additional attribute definitions that are specific to the subclass. The service class identifiers in the ServiceClassIDList attribute are listed in order from the most specific class to the most general class.

When a new service class is defined that is a subclass of an existing service class, the new service class retains all of the attributes defined in its superclass. Additional attributes will be defined that are specific to the new service class. In other words, the mechanism for adding new attributes to some of the instances of an existing service class is to create a new service class that is a subclass of the existing service class.

2.6.1 A Printer Service Class Example

A color postscript printer with duplex capability might conform to 4 ServiceClass definitions and have a ServiceClassIDList with UUIDs (See section 2.7.1 on page 123.) representing the following ServiceClasses:

DuplexColorPostscriptPrinterServiceClassID,
ColorPostscriptPrinterServiceClassID,
PostscriptPrinterServiceClassID,
PrinterServiceClassID

Note that this example is only illustrative. This may not be a practical printer class hierarchy.

2.7 SEARCHING FOR SERVICES

Once an SDP client has a service record handle, it may easily request the values of specific attributes, but how does a client initially acquire a service record handle for the desired service records? The Service Search transaction allows a client to retrieve the service record handles for particular service records based on the values of attributes contained within those service records.

The capability search for service records based on the values of arbitrary attributes is not provided. Rather, the capability is provided to search only for attributes whose values are Universally Unique Identifiers[1] (UUIDs). Important attributes of services that can be used to search for a service are represented as UUIDs.

2.7.1 UUID

A UUID is a universally unique identifier that is guaranteed to be unique across all space and all time. UUIDs can be independently created in a distributed fashion. No central registry of assigned UUIDs is required. A UUID is a 128-bit value.

To reduce the burden of storing and transferring 128-bit UUID values, a range of UUID values has been pre-allocated for assignment to often-used, registered purposes. The first UUID in this pre-allocated range is known as the Bluetooth Base UUID and has the value 00000000-0000-1000-8000-00805F9B34FB, from the Bluetooth Assigned Numbers document. UUID values in the pre-allocated range have aliases that are represented as 16-bit or 32-bit values. These aliases are often called 16-bit and 32-bit UUIDs, but it is important to note that each actually represents a 128-bit UUID value.

The full 128-bit value of a 16-bit or 32-bit UUID may be computed by a simple arithmetic operation.

128_bit_value = 16_bit_value * 2^{96} + Bluetooth_Base_UUID

128_bit_value = 32_bit_value * 2^{96} + Bluetooth_Base_UUID

A 16-bit UUID may be converted to 32-bit UUID format by zero-extending the 16-bit value to 32-bits. An equivalent method is to add the 16-bit UUID value to a zero-valued 32-bit UUID.

Note that two 16-bit UUIDs may be compared directly, as may two 32-bit UUIDs or two 128-bit UUIDs. If two UUIDs of differing sizes are to be com-

1. The format of UUIDs is defined by the International Organization for Standardization in ISO/IEC 11578:1996. “Information technology – Open Systems Interconnection – Remote Procedure Call (RPC)”

pared, the shorter UUID must be converted to the longer UUID format before comparison.

2.7.2 Service Search Patterns

A service search pattern is a list of UUIDs used to locate matching service records. A service search pattern is said to match a service record if each and every UUID in the service search pattern is contained within any of the service record's attribute values. The UUIDs need not be contained within any specific attributes or in any particular order within the service record. The service search pattern matches if the UUIDs it contains constitute a subset of the UUIDs in the service record's attribute values. The only time a service search pattern does not match a service record is if the service search pattern contains at least one UUID that is not contained within the service record's attribute values. Note also that a valid service search pattern must contain at least one UUID.

2.8 BROWSING FOR SERVICES

Normally, a client searches for services based on some desired characteristic(s) (represented by a UUID) of the services. However, there are times when it is desirable to discover which types of services are described by an SDP server's service records without any a priori information about the services. This process of looking for any offered services is termed browsing. In SDP, the mechanism for browsing for services is based on an attribute shared by all service classes. This attribute is called the BrowseGroupList attribute. The value of this attribute contains a list of UUIDs. Each UUID represents a browse group with which a service may be associated for the purpose of browsing.

When a client desires to browse an SDP server's services, it creates a service search pattern containing the UUID that represents the root browse group. All services that may be browsed at the top level are made members of the root browse group by having the root browse group's UUID as a value within the BrowseGroupList attribute.

Normally, if an SDP server has relatively few services, all of its services will be placed in the root browse group. However, the services offered by an SDP server may be organized in a browse group hierarchy, by defining additional browse groups below the root browse group. Each of these additional browse groups is described by a service record with a service class of BrowseGroupDescriptor.

A browse group descriptor service record defines a new browse group by means of its Group ID attribute. In order for a service contained in one of these newly defined browse groups to be browseable, the browse group descriptor service record that defines the new browse group must in turn be browseable. The hierarchy of browseable services that is provided by the use of browse group descriptor service records allows the services contained in an SDP

server to be incrementally browsed and is particularly useful when the SDP server contains many service records.

2.8.1 Example Service Browsing Hierarchy

Here is a fictitious service browsing hierarchy that may illuminate the manner in which browse group descriptors are used. Browse group descriptor service records are identified with (G); other service records with (S).

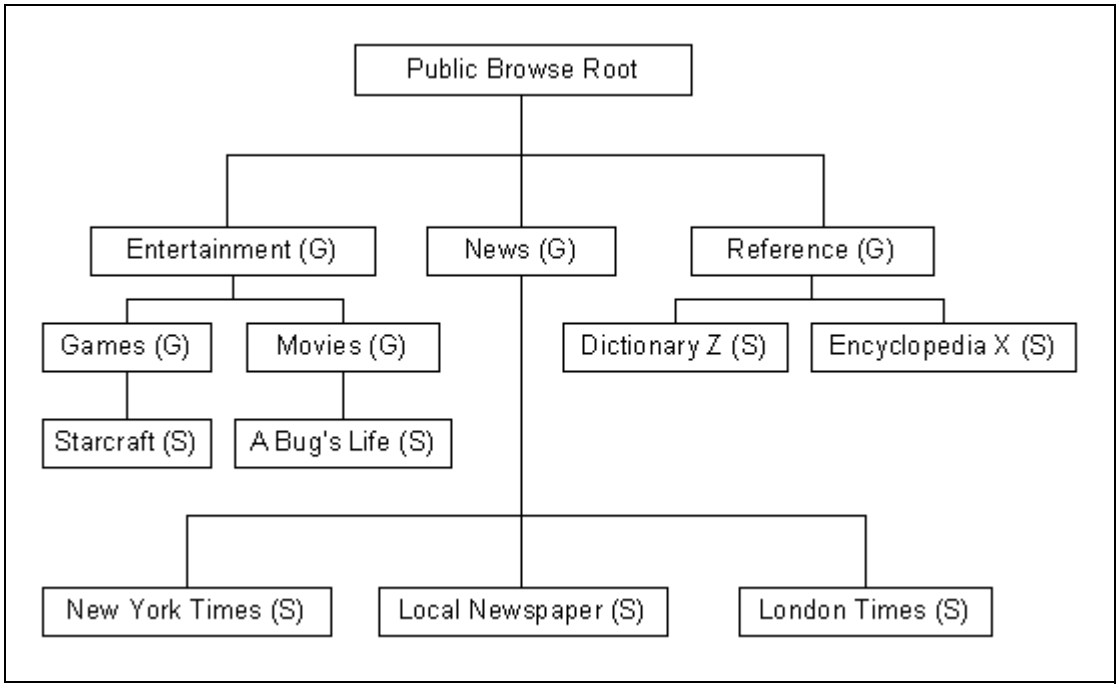


Figure 2.6:

This table shows the services records and service attributes necessary to implement the browse hierarchy.

Service Name	Service Class	Attribute Name	Attribute Value
Entertainment	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	EntertainmentID
News	BrowsegroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	NewsID
Reference	BrowseGroupDescriptor	BrowseGroupList	PublicBrowseRoot
		GroupID	ReferenceID
Games	BrowseGroupDescriptor	BrowseGroupList	EntertainmentID
		GroupID	GamesID
Movies	BrowseGroupDescriptor	BrowseGroupList	EntertainmentID
		GroupID	MoviesID
Starcraft	Video Game Class ID	BrowseGroupList	GamesID
A Bug's Life	Movie Class ID	BrowseGroupList	MovieID
Dictionary Z	Dictionary Class ID	BrowseGroupList	ReferenceID
Encyclopedia X	Encyclopedia Class ID	BrowseGroupList	ReferenceID
New York Times	Newspaper ID	BrowseGroupList	NewspaperID
London Times	Newspaper ID	BrowseGroupList	NewspaperID
Local Newspaper	Newspaper ID	BrowseGroupList	NewspaperID

Table 2.1:

3 DATA REPRESENTATION

Attribute values can contain information of various types with arbitrary complexity; thus enabling an attribute list to be generally useful across a wide variety of service classes and environments.

SDP defines a simple mechanism to describe the data contained within an attribute value. The primitive construct used is the data element.

3.1 DATA ELEMENT

A data element is a typed data representation. It consists of two fields: a header field and a data field. The header field, in turn, is composed of two parts: a type descriptor and a size descriptor. The data is a sequence of bytes whose length is specified in the size descriptor (described in section 3.3 on page 128) and whose meaning is (partially) specified by the type descriptor.

3.2 DATA ELEMENT TYPE DESCRIPTOR

A data element type is represented as a 5-bit type descriptor. The type descriptor is contained in the most significant (high-order) 5 bits of the first byte of the data element header. The following types have been defined.

Type Descriptor Value	Valid Size Descriptor Values	Type Description
0	0	Nil, the null type
1	0, 1, 2, 3, 4	Unsigned Integer
2	0, 1, 2, 3, 4	Signed twos-complement integer
3	1, 2, 4	UUID, a universally unique identifier
4	5, 6, 7	Text string
5	0	Boolean[1]
6	5, 6, 7	Data element sequence, a data element whose data field is a sequence of data elements
7	5, 6, 7	Data element alternative, data element whose data field is a sequence of data elements from which one data element is to be selected.
8	5, 6, 7	URL, a uniform resource locator
9-31		Reserved

Table 3.1: Data Element Type.

1. False is represented by the value 0, and true is represented by the value 1. However, to maximize interoperability, any non-zero value received must be accepted as representing true.

3.3 DATA ELEMENT SIZE DESCRIPTOR

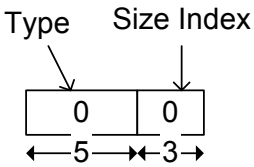
The data element size descriptor is represented as a 3-bit size index followed by 0, 8, 16, or 32 bits. The size index is contained in the least significant (low-order) 3 bits of the first byte of the data element header. The size index is encoded as follows.

Size Index	Additional bits	Data Size
0	0	1 byte. Exception: if the data element type is nil, the data size is 0 bytes.
1	0	2 bytes
2	0	4 bytes
3	0	8 bytes
4	0	16 bytes
5	8	The data size is contained in the additional 8 bits, which are interpreted as an unsigned integer.
6	16	The data size is contained in the additional 16 bits, which are interpreted as an unsigned integer.
7	32	The data size is contained in the additional 32 bits, which are interpreted as an unsigned integer.

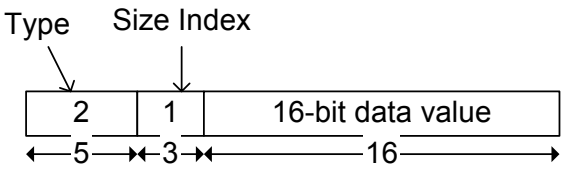
Table 3.2:

3.4 DATA ELEMENT EXAMPLES

Nil is represented as:



A 16-bit signed integer is represented as:



The 3 character ASCII string "Hat" is represented as:

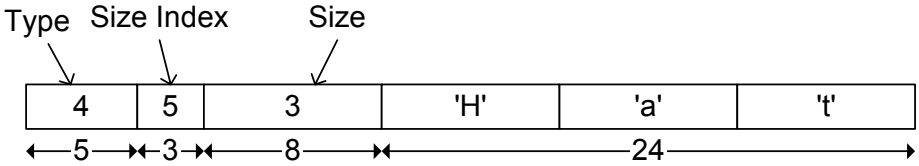


Figure 3.1:

4 PROTOCOL DESCRIPTION

SDP is a simple protocol with minimal requirements on the underlying transport. It can function over a reliable packet transport (or even unreliable, if the client implements timeouts and repeats requests as necessary).

SDP uses a request/response model where each transaction consists of one request protocol data unit (PDU) and one response PDU. In the case where SDP is used with the Bluetooth L2CAP transport protocol, only one SDP request PDU per connection to a given SDP server may be outstanding at a given instant. In other words, a client must receive a response to each request before issuing another request on the same L2CAP connection. Limiting SDP to sending one unacknowledged request PDU provides a simple form of flow control.

The protocol examples found in Appendix B – Example SDP Transactions, may be helpful in understanding the protocol transactions.

4.1 TRANSFER BYTE ORDER

The service discovery protocol transfers multiple-byte fields in standard network byte order (Big Endian), with more significant (high-order) bytes being transferred before less-significant (low-order) bytes.

4.2 PROTOCOL DATA UNIT FORMAT

Every SDP PDU consists of a PDU header followed by PDU-specific parameters. The header contains three fields: a PDU ID, a Transaction ID, and a ParameterLength. Each of these header fields is described here. Parameters may include a continuation state parameter, described below; PDU-specific parameters for each PDU type are described later in separate PDU descriptions.

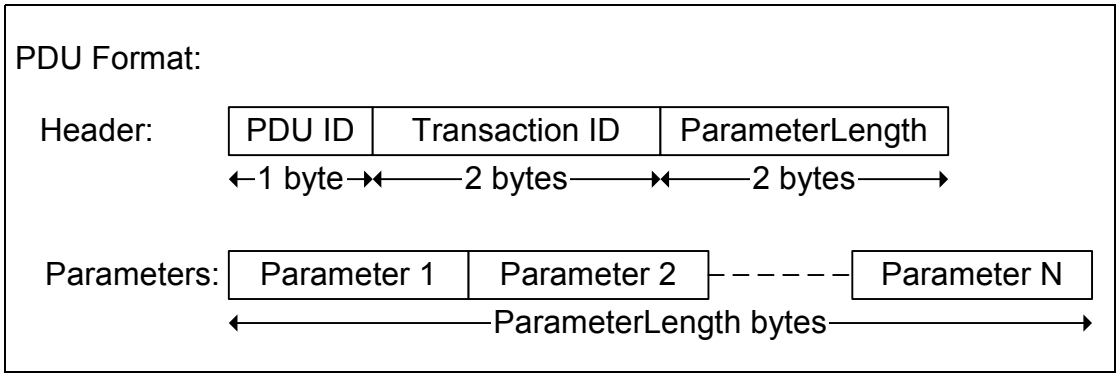


Figure 4.1:

PDU ID: *Size: 1 Byte*

Value	Parameter Description
N	The PDU ID field identifies the type of PDU. I.e. its meaning and the specific parameters.
0x00	Reserved
0x01	SDP_ErrorResponse
0x02	SDP_ServiceSearchRequest
0x03	SDP_ServiceSearchResponse
0x04	SDP_ServiceAttributeRequest
0x05	SDP_ServiceAttributeResponse
0x06	SDP_ServiceSearchAttributeRequest
0x07	SDP_ServiceSearchAttributeResponse
0x07-0xFF	Reserved

TransactionID: *Size: 2 Bytes*

Value	Parameter Description
N	The TransactionID field uniquely identifies request PDUs and is used to match response PDUs to request PDUs. The SDP client can choose any value for a request's TransactionID provided that it is different from all outstanding requests. The TransactionID value in response PDUs is required to be the same as the request that is being responded to. Range: 0x0000 – 0xFFFF

ParameterLength: *Size: 2 Bytes*

Value	Parameter Description
N	The ParameterLength field specifies the length (in bytes) of all parameters contained in the PDU. Range: 0x0000 – 0xFFFF

4.3 PARTIAL RESPONSES AND CONTINUATION STATE

Some SDP requests may require responses that are larger than can fit in a single response PDU. In this case, the SDP server will generate a partial response along with a continuation state parameter. The continuation state parameter can be supplied by the client in a subsequent request to retrieve the next portion of the complete response. The continuation state parameter is a variable length field whose first byte contains the number of additional bytes of continuation information in the field. The format of the continuation information is not standardized among SDP servers. Each continuation state parameter is meaningful only to the SDP server that generated it.

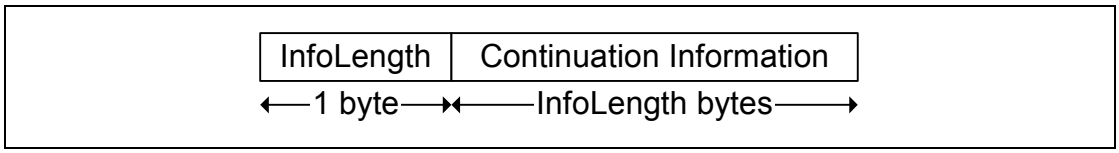


Figure 4.2: Continuation State Format

After a client receives a partial response and the accompanying continuation state parameter, it can re-issue the original request (with a new transaction ID) and include the continuation state in the new request indicating to the server that the remainder of the original response is desired. The maximum allowable value of the InfoLength field is 16 (0x10).

Note that an SDP server can split a response at any arbitrary boundary when it generates a partial response. The SDP server may select the boundary based on the contents of the reply, but is not required to do so.

4.4 ERROR HANDLING

Each transaction consists of a request and a response PDU. Generally, each type of request PDU has a corresponding type of response PDU. However, if the server determines that a request is improperly formatted or for any reason the server cannot respond with the appropriate PDU type, it will respond with an SDP_ErrorResponse PDU.

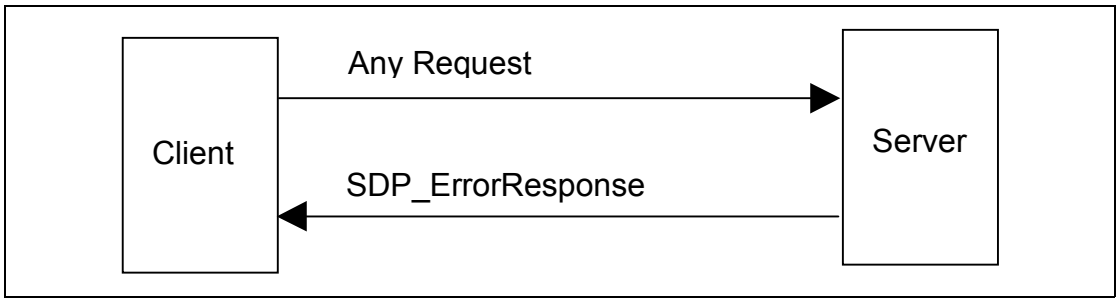


Figure 4.3:

4.4.1 SDP_ErrorResponse PDU

PDU Type	PDU ID	Parameters
SDP_ErrorResponse	0x01	ErrorCode, ErrorInfo

Description:

The SDP server generates this PDU type in response to an improperly formatted request PDU or when the SDP server, for whatever reason, cannot generate an appropriate response PDU.

PDU Parameters:

ErrorCode: *Size: 2 Bytes*

Value	Parameter Description
N	The ErrorCode identifies the reason that an SDP_ErrorResponse PDU was generated.
0x0000	Reserved
0x0001	Invalid/unsupported SDP version
0x0002	Invalid Service Record Handle
0x0003	Invalid request syntax
0x0004	Invalid PDU Size
0x0005	Invalid Continuation State
0x0006	Insufficient Resources to satisfy Request
0x0007-0xFFFF	Reserved

ErrorInfo: *Size: N Bytes*

Value	Parameter Description
Error-specific	ErrorInfo is an ErrorCode-specific parameter. Its interpretation depends on the ErrorCode parameter. The currently defined ErrorCode values do not specify the format of an ErrorInfo field.

4.5 SERVICESEARCH TRANSACTION

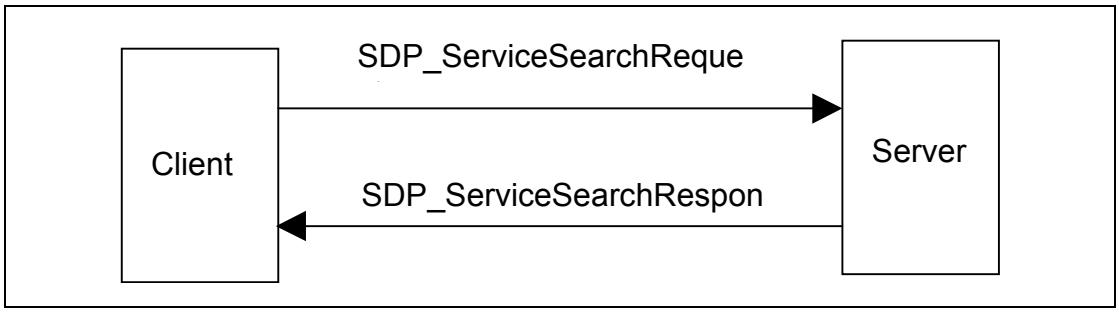


Figure 4.4:

4.5.1 SDP_ServiceSearchRequest PDU

PDU Type	PDU ID	Parameters
SDP_ServiceSearchRequest	0x02	ServiceSearchPattern, MaximumServiceRecordCount, ContinuationState

Description:

The SDP client generates an SDP_ServiceSearchRequest to locate service records that match the service search pattern given as the first parameter of the PDU. Upon receipt of this request, the SDP server will examine its service record data base and return an SDP_ServiceSearchResponse containing the service record handles of service records that match the given service search pattern.

Note that no mechanism is provided to request information for all service records. However, see section 2.8 on page 124 for a description of a mechanism that permits browsing for non-specific services without a priori knowledge of the services.

PDU Parameters:

ServiceSearchPattern: *Size: Varies*

Value	Parameter Description
Data Element Sequence	The ServiceSearchPattern is a data element sequence where each element in the sequence is a UUID. The sequence must contain at least one UUID. The maximum number of UUIDs in the sequence is 12[1]. The list of UUIDs constitutes a service search pattern.

1. The value of 12 has been selected as a compromise between the scope of a service search and the size of a search request PDU. It is not expected that more than 12 UUIDs will be useful in a service search pattern.

MaximumServiceRecordCount: *Size: 2 Bytes*

Value	Parameter Description
N	MaximumServiceRecordCount is a 16-bit count specifying the maximum number of service record handles to be returned in the response(s) to this request. The SDP server should not return more handles than this value specifies. If more than N service records match the request, the SDP server determines which matching service record handles to return in the response(s). Range: 0x0001-0xFFFF

ContinuationState: *Size: 1 to 17 Bytes*

Value	**Parameter Description**
Continuation State	ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N is required to be less than or equal to 16. If no continuation state is to be provided in the request, N is set to 0.

4.5.2 SDP_ServiceSearchResponse PDU

PDU Type	**PDU ID**	**Parameters**
SDP_ServiceSearchResponse	0x03	TotalServiceRecordCount, CurrentServiceRecordCount, ServiceRecordHandleList, ContinuationState

Description:

The SDP server generates an SDP_ServiceSearchResponse upon receipt of a valid SDP_ServiceSearchRequest. The response contains a list of service record handles for service records that match the service search pattern given in the request. Note that if a partial response is generated, it must contain an integral number of complete service record handles; a service record handle value may not be split across multiple PDUs.

PDU Parameters:

TotalServiceRecordCount: *Size: 2 Bytes*

Value	Parameter Description
N	The TotalServiceRecordCount is an integer containing the number of service records that match the requested service search pattern. If no service records match the requested service search pattern, this parameter is set to 0. N should never be larger than the MaximumServiceRecordCount value specified in the SDP_ServiceSearchRequest. When multiple partial responses are used, each partial response contains the same value for TotalServiceRecordCount. Range: 0x0000-0xFFFF

CurrentServiceRecordCount: *Size: 2 Bytes*

Value	Parameter Description
N	The CurrentServiceRecordCount is an integer indicating the number of service record handles that are contained in the next parameter. If no service records match the requested service search pattern, this parameter is set to 0. N should never be larger than the TotalServiceRecordCount value specified in the current response. Range: 0x0000-0xFFFF

ServiceRecordHandleList: *Size: (CurrentServiceRecordCount*4) Bytes*

Value	Parameter Description
List of 32-bit handles	The ServiceRecordHandleList contains a list of service record handles. The number of handles in the list is given in the CurrentServiceRecordCount parameter. Each of the handles in the list refers to a service record that matches the requested service search pattern. Note that this list of service record handles does not have the format of a data element. It contains no header fields, only the 32-bit service record handles.

ContinuationState: *Size: 1 to 17 Bytes*

Value	Parameter Description
Continuation State	ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is contained in the PDU, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response.

4.6 SERVICEATTRIBUTE TRANSACTION

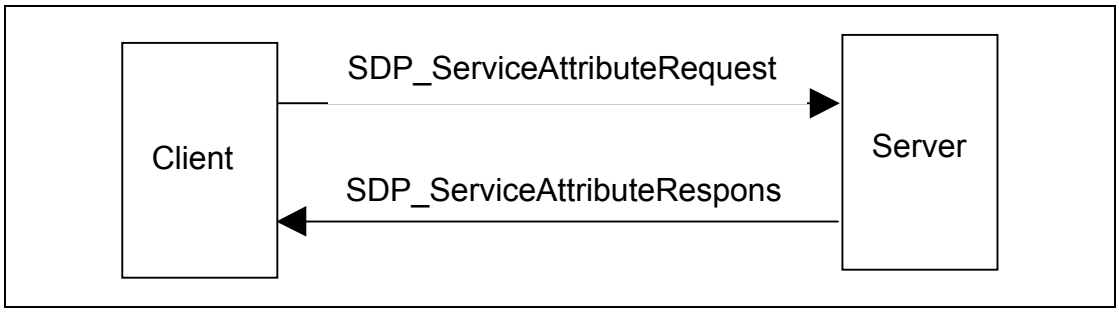


Figure 4.5:

4.6.1 SDP_ServiceAttributeRequest PDU

PDU Type	PDU ID	Parameters
SDP_ServiceAttributeRequest	0x04	ServiceRecordHandle, MaximumAttributeByteCount, AttributeIDList, ContinuationState

Description:

The SDP client generates an SDP_ServiceAttributeRequest to retrieve specified attribute values from a specific service record. The service record handle of the desired service record and a list of desired attribute IDs to be retrieved from that service record are supplied as parameters.

Command Parameters:

ServiceRecordHandle: *Size: 4 Bytes*

Value	Parameter Description
32-bit handle	The ServiceRecordHandle parameter specifies the service record from which attribute values are to be retrieved. The handle is obtained via a previous SDP_ServiceSearch transaction.

MaximumAttributeByteCount: *Size: 2 Bytes*

Value	Parameter Description
N	MaximumAttributeByteCount specifies the maximum number of bytes of attribute data to be returned in the response to this request. The SDP server should not return more than N bytes of attribute data in the response PDU. If the requested attributes require more than N bytes, the SDP server determines how to segment the list. In this case the client may request each successive segment by issuing a request containing the continuation state that was returned in the previous response PDU. Note that in the case where multiple response PDUs are needed to return the attribute data, MaximumAttributeByteCount specifies the maximum size of the portion of the attribute data contained in each response PDU. Range: 0x0007-0xFFFF

AttributeIDList: *Size: Varies*

Value	Parameter Description
Data Element Sequence	The AttributeIDList is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. Each attribute ID is encoded as a 16-bit unsigned integer data element. Each attribute ID range is encoded as a 32-bit unsigned integer data element, where the high order 16 bits are interpreted as the beginning attribute ID of the range and the low order 16 bits are interpreted as the ending attribute ID of the range. The attribute IDs contained in the AttributeIDList must be listed in ascending order without duplication of any attribute ID values. Note that all attributes may be requested by specifying a range of 0x0000-0xFFFF.

ContinuationState: *Size: 1 to 17 Bytes*

Value	Parameter Description
Continuation State	ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N is required to be less than or equal to 16. If no continuation state is to be provided in the request, N is set to 0.

4.6.2 SDP_ServiceAttributeResponse PDU

PDU Type	PDU ID	Parameters
SDP_ServiceAttributeResponse	0x05	AttributeListByteCount, AttributeList, ContinuationState

Description:

The SDP server generates an SDP_ServiceAttributeResponse upon receipt of a valid SDP_ServiceAttributeRequest. The response contains a list of attributes (both attribute ID and attribute value) from the requested service record.

PDU Parameters:

AttributeListByteCount: *Size: 2 Bytes*

Value	Parameter Description
N	The AttributeListByteCount contains a count of the number of bytes in the AttributeList parameter. N must never be larger than the MaximumAttributeByteCount value specified in the SDP_ServiceAttributeRequest. Range: 0x0002-0xFFFF

AttributeList: *Size: AttributeListByteCount*

Value	Parameter Description
Data Element Sequence	The AttributeList is a data element sequence containing attribute IDs and attribute values. The first element in the sequence contains the attribute ID of the first attribute to be returned. The second element in the sequence contains the corresponding attribute value. Successive pairs of elements in the list contain additional attribute ID and value pairs. Only attributes that have non-null values within the service record and whose attribute IDs were specified in the SDP_ServiceAttributeRequest are contained in the AttributeList. Neither an attribute ID nor an attribute value is placed in the AttributeList for attributes in the service record that have no value. The attributes are listed in ascending order of attribute ID value.

ContinuationState: *Size: 1 to 17 Bytes*

Value	Parameter Description
Continuation State	ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is given, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response.

4.7 SERVICESEARCHATTRIBUTE TRANSACTION

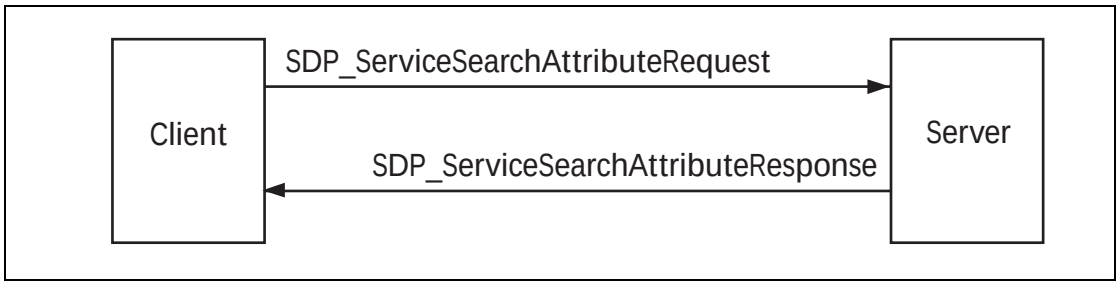


Figure 4.6:

4.7.1 SDP_ServiceSearchAttributeRequest PDU

PDU Type	PDU ID	Parameters
SDP_ServiceSearchAttributeRequest	0x06	ServiceSearchPattern, MaximumAttributeByteCount, AttributeIDList, ContinuationState

Description:

The SDP_ServiceSearchAttributeRequest transaction combines the capabilities of the SDP_ServiceSearchRequest and the SDP_ServiceAttributeRequest into a single request. As parameters, it contains both a service search pattern and a list of attributes to be retrieved from service records that match the service search pattern. The SDP_ServiceSearchAttributeRequest and its response are more complex and may require more bytes than separate SDP_ServiceSearch and SDP_ServiceAttribute transactions. However, using SDP_ServiceSearchAttributeRequest may reduce the total number of SDP transactions, particularly when retrieving multiple service records.

Note that the service record handle for each service record is contained in the ServiceRecordHandle attribute of that service and may be requested along with other attributes.

PDU Parameters:

ServiceSearchPattern: *Size: Varies*

Value	Parameter Description
Data Element Sequence	The ServiceSearchPattern is a data element sequence where each element in the sequence is a UUID. The sequence must contain at least one UUID. The maximum number of UUIDs in the sequence is 12[1]. The list of UUIDs constitutes a service search pattern.

1. The value of 12 has been selected as a compromise between the scope of a service search and the size of a search request PDU. It is not expected that more than 12 UUIDs will be useful in a service search pattern.

MaximumAttributeByteCount: *Size: 2 Bytes*

Value	Parameter Description
N	MaximumAttributeByteCount specifies the maximum number of bytes of attribute data to be returned in the response to this request. The SDP server should not return more than N bytes of attribute data in the response PDU. If the requested attributes require more than N bytes, the SDP server determines how to segment the list. In this case the client may request each successive segment by issuing a request containing the continuation state that was returned in the previous response PDU. Note that in the case where multiple response PDUs are needed to return the attribute data, MaximumAttributeByteCount specifies the maximum size of the portion of the attribute data contained in each response PDU. Range: 0x0007-0xFFFF

AttributeIDList: *Size: Varies*

Value	Parameter Description
Data Element Sequence	The AttributeIDList is a data element sequence where each element in the list is either an attribute ID or a range of attribute IDs. Each attribute ID is encoded as a 16-bit unsigned integer data element. Each attribute ID range is encoded as a 32-bit unsigned integer data element, where the high order 16 bits are interpreted as the beginning attribute ID of the range and the low order 16 bits are interpreted as the ending attribute ID of the range. The attribute IDs contained in the AttributeIDList must be listed in ascending order without duplication of any attribute ID values. Note that all attributes may be requested by specifying a range of 0x0000-0xFFFF.

ContinuationState: *Size: 1 to 17 Bytes*

Value	Parameter Description
Continuation State	ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation state information that were returned in a previous response from the server. N is required to be less than or equal to 16. If no continuation state is to be provided in the request, N is set to 0.

4.7.2 SDP_ServiceSearchAttributeResponse PDU

PDU Type	PDU ID	Parameters
SDP_ServiceSearchAttributeResponse	0x07	AttributeListsByteCount, AttributeLists, ContinuationState

Description:

The SDP server generates an SDP_ServiceSearchAttributeResponse upon receipt of a valid SDP_ServiceSearchAttributeRequest. The response contains a list of attributes (both attribute ID and attribute value) from the service records that match the requested service search pattern.

PDU Parameters:

AttributeListsByteCount: *Size: 2 Bytes*

Value	Parameter Description
N	The AttributeListsByteCount contains a count of the number of bytes in the AttributeLists parameter. N must never be larger than the MaximumAttributeByteCount value specified in the SDP_ServiceSearchAttributeRequest. Range: 0x0002-0xFFFF

AttributeLists: *Size: Varies*

Value	Parameter Description
Data Element Sequence	The AttributeLists is a data element sequence where each element in turn is a data element sequence representing an attribute list. Each attribute list contains attribute IDs and attribute values from one service record. The first element in each attribute list contains the attribute ID of the first attribute to be returned for that service record. The second element in each attribute list contains the corresponding attribute value. Successive pairs of elements in each attribute list contain additional attribute ID and value pairs. Only attributes that have non-null values within the service record and whose attribute IDs were specified in the SDP_ServiceSearchAttributeRequest are contained in the AttributeLists. Neither an attribute ID nor attribute value is placed in AttributeLists for attributes in the service record that have no value. Within each attribute list, the attributes are listed in ascending order of attribute ID value.

ContinuationState: *Size: 1 to 17 Bytes*

Value	Parameter Description
Continuation State	ContinuationState consists of an 8-bit count, N, of the number of bytes of continuation state information, followed by the N bytes of continuation information. If the current response is complete, this parameter consists of a single byte with the value 0. If a partial response is given, the ContinuationState parameter may be supplied in a subsequent request to retrieve the remainder of the response.

5 SERVICE ATTRIBUTE DEFINITIONS

The service classes and attributes contained in this document are necessarily a partial list of the service classes and attributes supported by SDP. Only service classes that directly support the SDP server are included in this document. Additional service classes will be defined in other documents and possibly in future revisions of this document. Also, it is expected that additional attributes will be discovered that are applicable to a broad set of services; these may be added to the list of Universal attributes in future revisions of this document.

5.1 UNIVERSAL ATTRIBUTE DEFINITIONS

Universal attributes are those service attributes whose definitions are common to all service records. Note that this does not mean that every service record must contain values for all of these service attributes. However, if a service record has a service attribute with an attribute ID allocated to a universal attribute, the attribute value must conform to the universal attribute's definition.

Only two attributes are required to exist in every service record instance. They are the ServiceRecordHandle (attribute ID 0x0000) and the ServiceClassIDList (attribute ID 0x0001). All other service attributes are optional within a service record.

5.1.1 ServiceRecordHandle Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceRecordHandle	0x0000	32-bit unsigned integer

Description:

A service record handle is a 32-bit number that uniquely identifies each service record within an SDP server. It is important to note that, in general, each handle is unique only within each SDP server. If SDP server S1 and SDP server S2 both contain identical service records (representing the same service), the service record handles used to reference these identical service records are completely independent. The handle used to reference the service on S1 will, in general, be meaningless if presented to S2.

5.1.2 ServiceClassIDList Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceClassIDList	0x0001	Data Element Sequence

Description:

The ServiceClassIDList attribute consists of a data element sequence in which each data element is a UUID representing the service classes that a given service record conforms to. The UUIDs are listed in order from the most specific class to the most general class. The ServiceClassIDList must contain at least one service class UUID.

5.1.3 ServiceRecordState Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceRecordState	0x0002	32-bit unsigned integer

Description:

The ServiceRecordState is a 32-bit integer that is used to facilitate caching of ServiceAttributes. If this attribute is contained in a service record, its value is guaranteed to change when any other attribute value is added to, deleted from or changed within the service record. This permits a client to check the value of this single attribute. If its value has not changed since it was last checked, the client knows that no other attribute values within the service record have changed.

5.1.4 ServiceID Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceID	0x0003	UUID

Description:

The ServiceID is a UUID that universally and uniquely identifies the service instance described by the service record. This service attribute is particularly useful if the same service is described by service records in more than one SDP server.

5.1.5 ProtocolDescriptorList Attribute

Attribute Name	Attribute ID	Attribute Value Type
ProtocolDescriptorList	0x0004	Data Element Sequence or Data Element Alternative

Description:

The ProtocolDescriptorList attribute describes one or more protocol stacks that may be used to gain access to the service described by the service record.

If the ProtocolDescriptorList describes a single stack, it takes the form of a data element sequence in which each element of the sequence is a protocol descriptor. Each protocol descriptor is, in turn, a data element sequence whose first element is a UUID identifying the protocol and whose successive elements are protocol-specific parameters. Potential protocol-specific parameters are a protocol version number and a connection-port number. The protocol descriptors are listed in order from the lowest layer protocol to the highest layer protocol used to gain access to the service.

If it is possible for more than one kind of protocol stack to be used to gain access to the service, the ProtocolDescriptorList takes the form of a data element alternative where each member is a data element sequence as described in the previous paragraph.

Protocol Descriptors

A protocol descriptor identifies a communications protocol and provides protocol-specific parameters. A protocol descriptor is represented as a data element sequence. The first data element in the sequence must be the UUID that identifies the protocol. Additional data elements optionally provide protocol-specific information, such as the L2CAP protocol/service multiplexer (PSM) and the RFCOMM server channel number (CN) shown below.

ProtocolDescriptorList Examples

These examples are intended to be illustrative. The parameter formats for each protocol are not defined within this specification.

In the first two examples, it is assumed that a single RFCOMM instance exists on top of the L2CAP layer. In this case, the L2CAP protocol specific information (PSM) points to the single instance of RFCOMM. In the last example, two different and independent RFCOMM instances are available on top of the L2CAP layer. In this case, the L2CAP protocol specific information (PSM) points to a distinct identifier that distinguishes each of the RFCOMM instances. According to the L2CAP specification, this identifier takes values in the range 0x1000-0xFFFF.

IrDA-like printer

((L2CAP, PSM=RFCOMM), (RFCOMM, CN=1), (PostscriptStream))

IP Network Printing

((L2CAP, PSM=RFCOMM), (RFCOMM, CN=2), (PPP), (IP), (TCP), (IPP))

Synchronization Protocol Descriptor Example

((L2CAP, PSM=0x1001), (RFCOMM, CN=1), (Obex), (vCal))

((L2CAP, PSM=0x1002), (RFCOMM, CN=1), (Obex), (otherSynchronisationApplication))

5.1.6 BrowseGroupList Attribute

Attribute Name	Attribute ID	Attribute Value Type
BrowseGroupList	0x0005	Data Element Sequence

Description:

The BrowseGroupList attribute consists of a data element sequence in which each element is a UUID that represents a browse group to which the service record belongs. The top-level browse group ID, called PublicBrowseRoot and representing the root of the browsing hierarchy, has the value 00001002-0000-1000-8000-00805F9B34FB (UUID16: 0x1002) from the Bluetooth Assigned Numbers document.

5.1.7 LanguageBaseAttributeIDList Attribute

Attribute Name	Attribute ID	Attribute Value Type
LanguageBaseAttributeIDList	0x0006	Data Element Sequence

Description:

In order to support human-readable attributes for multiple natural languages in a single service record, a base attribute ID is assigned for each of the natural languages used in a service record. The human-readable universal attributes are then defined with an attribute ID offset from each of these base values, rather than with an absolute attribute ID.

The LanguageBaseAttributeIDList attribute is a list in which each member contains a language identifier, a character encoding identifier, and a base attribute

ID for each of the natural languages used in the service record. The LanguageBaseAttributeIDList attribute consists of a data element sequence in which each element is a 16-bit unsigned integer. The elements are grouped as triplets (threes).

The first element of each triplet contains an identifier representing the natural language. The language is encoded according to ISO 639:1988 (E/F): “Code for the representation of names of languages”.

The second element of each triplet contains an identifier that specifies a character encoding used for the language. Values for character encoding can be found in IANA's database[1], and have the values that are referred to as MIBEnum values. The recommended character encoding is UTF-8.

The third element of each triplet contains an attribute ID that serves as the base attribute ID for the natural language in the service record. Different service records within a server may use different base attribute ID values for the same language.

To facilitate the retrieval of human-readable universal attributes in a principal language, the base attribute ID value for the primary language supported by a service record must be 0x0100. Also, if a LanguageBaseAttributeIDList attribute is contained in a service record, the base attribute ID value contained in its first element must be 0x0100.

5.1.8 ServiceInfoTimeToLive Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceInfoTimeToLive	0x0007	32-bit unsigned integer

Description:

The ServiceTimeToLive attribute is a 32-bit integer that contains the number of seconds for which the information in a service record is expected to remain valid and unchanged. This time interval is measured from the time that the attribute value is retrieved from the SDP server. This value does not imply a guarantee that the service record will remain available or unchanged. It is simply a hint that a client may use to determine a suitable polling interval to revalidate the service record contents.

1. See http://www.isi.edu/in-notes/iana/assignments/character-sets

5.1.9 ServiceAvailability Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceAvailability	0x0008	8-bit unsigned integer

Description:

The ServiceAvailability attribute is an 8-bit unsigned integer that represents the relative ability of the service to accept additional clients. A value of 0xFF indicates that the service is not currently in use and is thus fully available, while a value of 0x00 means that the service is not accepting new clients. For services that support multiple simultaneous clients, intermediate values indicate the relative availability of the service on a linear scale.

For example, a service that can accept up to 3 clients should provide ServiceAvailability values of 0xFF, 0xAA, 0x55, and 0x00 when 0, 1, 2, and 3 clients, respectively, are utilizing the service. The value 0xAA is approximately (2/3) * 0xFF and represents 2/3 availability, while the value 0x55 is approximately (1/3)*0xFF and represents 1/3 availability. Note that the availability value may be approximated as

(1 - (current_number_of_clients / maximum_number_of_clients)) * 0xFF

When the maximum number of clients is large, this formula must be modified to ensure that ServiceAvailability values of 0x00 and 0xFF are reserved for their defined meanings of unavailability and full availability, respectively.

Note that the maximum number of clients a service can support may vary according to the resources utilized by the service's current clients.

A non-zero value for ServiceAvailability does not guarantee that the service will be available for use. It should be treated as a hint or an approximation of availability status.

5.1.10 BluetoothProfileDescriptorList Attribute

Attribute Name	Attribute ID	Attribute Value Type
BluetoothProfileDescriptorList	0x0009	Data Element Sequence

Description:

The BluetoothProfileDescriptorList attribute consists of a data element sequence in which each element is a profile descriptor that contains information about a Bluetooth profile to which the service represented by this service record conforms. Each profile descriptor is a data element sequence whose first element is the UUID assigned to the profile and whose second element is a 16-bit profile version number.

Each version of a profile is assigned a 16-bit unsigned integer profile version number, which consists of two 8-bit fields. The higher-order 8 bits contain the major version number field and the lower-order 8 bits contain the minor version number field. The initial version of each profile has a major version of 1 and a minor version of 0. When upward compatible changes are made to the profile, the minor version number will be incremented. If incompatible changes are made to the profile, the major version number will be incremented.

5.1.11 DocumentationURL Attribute

Attribute Name	Attribute ID	Attribute Value Type
DocumentationURL	0x000A	URL

Description:

This attribute is a URL which points to documentation on the service described by a service record.

5.1.12 ClientExecutableURL Attribute

Attribute Name	Attribute ID	Attribute Value Type
ClientExecutableURL	0x000B	URL

Description:

This attribute contains a URL that refers to the location of an application that may be used to utilize the service described by the service record. Since different operating environments require different executable formats, a mechanism has been defined to allow this single attribute to be used to locate an executable that is appropriate for the client device's operating environment. In the attribute value URL, the first byte with the value 0x2A (ASCII character '*') is to be replaced by the client application with a string representing the desired operating environment before the URL is to be used.

The list of standardized strings representing operating environments is contained in the Bluetooth Assigned Numbers document.

For example, assume that the value of the ClientExecutableURL attribute is http://my.fake/public/*/client.exe. On a device capable of executing SH3 WindowsCE files, this URL would be changed to http://my.fake/public/sh3-microsoft-wince/client.exe. On a device capable of executing Windows 98 binaries, this URL would be changed to http://my.fake/public/i86-microsoft-win98/client.exe.

5.1.13 IconURL Attribute

Attribute Name	Attribute ID	Attribute Value Type
IconURL	0x000C	URL

Description:

This attribute contains a URL that refers to the location of an icon that may be used to represent the service described by the service record. Since different hardware devices require different icon formats, a mechanism has been defined to allow this single attribute to be used to locate an icon that is appropriate for the client device. In the attribute value URL, the first byte with the value 0x2A (ASCII character '*') is to be replaced by the client application with a string representing the desired icon format before the URL is to be used.

The list of standardized strings representing icon formats is contained in the Bluetooth Assigned Numbers document.

For example, assume that the value of the IconURL attribute is http://my.fake/public/icons/*. On a device that prefers 24 x 24 icons with 256 colors, this URL would be changed to http://my.fake/public/icons/24x24x8.png. On a device that prefers 10 x 10 monochrome icons, this URL would be changed to http://my.fake/public/icons/10x10x1.png.

5.1.14 ServiceName Attribute

Attribute Name	Attribute ID Offset	Attribute Value Type
ServiceName	0x0000	String

Description:

The ServiceName attribute is a string containing the name of the service represented by a service record. It should be brief and suitable for display with an Icon representing the service. The offset 0x0000 must be added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

5.1.15 ServiceDescription Attribute

Attribute Name	Attribute ID Offset	Attribute Value Type
ServiceDescription	0x0001	String

Description:

This attribute is a string containing a brief description of the service. It should be less than 200 characters in length. The offset 0x0001 must be added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

5.1.16 ProviderName Attribute

Attribute Name	Attribute ID Offset	Attribute Value Type
ProviderName	0x0002	String

Description:

This attribute is a string containing the name of the person or organization providing the service. The offset 0x0002 must be added to the attribute ID base (contained in the LanguageBaseAttributeIDList attribute) in order to compute the attribute ID for this attribute.

5.1.17 Reserved Universal Attribute IDs

Attribute IDs in the range of 0x000D-0x01FF are reserved.

5.2 SERVICEDISCOVERYSERVER SERVICE CLASS ATTRIBUTE DEFINITIONS

This service class describes service records that contain attributes of service discovery server itself. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the ServiceDiscoveryServerServiceClassID. Note that all of the universal attributes may be included in service records of the ServiceDiscoveryServer class.

5.2.1 ServiceRecordHandle Attribute

Described in the universal attribute definition for ServiceRecordHandle.

Value

A 32-bit integer with the value 0x000000000.

5.2.2 ServiceClassIDList Attribute

Described in the universal attribute definition for ServiceClassIDList.

Value

A UUID representing the ServiceDiscoveryServerServiceClassID.

5.2.3 VersionNumberList Attribute

Attribute Name	Attribute ID	Attribute Value Type
VersionNumberList	0x0200	Data Element Sequence

Description:

The VersionNumberList is a data element sequence in which each element of the sequence is a version number supported by the SDP server.

A version number is a 16-bit unsigned integer consisting of two fields. The higher-order 8 bits contain the major version number field and the low-order 8 bits contain the minor version number field. The initial version of SDP has a major version of 1 and a minor version of 0. When upward compatible changes are made to the protocol, the minor version number will be incremented. If incompatible changes are made to SDP, the major version number will be incremented. This guarantees that if a client and a server support a common major version number, they can communicate if each uses only features of the specification with a minor version number that is supported by both client and server.

5.2.4 ServiceDatabaseState Attribute

Attribute Name	Attribute ID	Attribute Value Type
ServiceDatabaseState	0x0201	32-bit unsigned integer

Description:

The ServiceDatabaseState is a 32-bit integer that is used to facilitate caching of service records. If this attribute exists, its value is guaranteed to change when any of the other service records are added to or deleted from the server's database. If this value has not changed since the last time a client queried its value, the client knows that a) none of the other service records maintained by the SDP server have been added or deleted; and b) any service record handles acquired from the server are still valid. A client should query this attribute's value when a connection to the server is established, prior to using any service record handles acquired during a previous connection.

Note that the ServiceDatabaseState attribute does not change when existing service records are modified, including the addition, removal, or modification of service attributes. A service record's ServiceRecordState attribute indicates when that service record is modified.

5.2.5 Reserved Attribute IDs

Attribute IDs in the range of 0x0202-0x02FF are reserved.

5.3 BROWSEGROUPDESCRIPTOR SERVICE CLASS ATTRIBUTE DEFINITIONS

This service class describes the ServiceRecord provided for each BrowseGroupDescriptor service offered on a Bluetooth device. The attributes listed in this section are only valid if the ServiceClassIDList attribute contains the BrowseGroupDescriptorServiceClassID. Note that all of the universal attributes may be included in service records of the BrowseGroupDescriptor class.

5.3.1 ServiceClassIDList Attribute

Described in the universal attribute definition for ServiceClassIDList.

Value

A UUID representing the BrowseGroupDescriptorServiceClassID.

5.3.2 GroupID Attribute

Attribute Name	Attribute ID	Attribute Value Type
GroupID	0x0200	UUID

Description:

This attribute contains a UUID that can be used to locate services that are members of the browse group that this service record describes.

5.3.3 Reserved Attribute IDs

Attribute IDs in the range of 0x0201-0x02FF are reserved.

APPENDIX A – BACKGROUND INFORMATION

A.1. Service Discovery

As computing continues to move to a network-centric model, finding and making use of services that may be available in the network becomes increasingly important. Services can include common ones such as printing, paging, FAXing, and so on, as well as various kinds of information access such as teleconferencing, network bridges and access points, eCommerce facilities, and so on — most any kind of service that a server or service provider might offer. In addition to the need for a standard way of discovering available services, there are other considerations: getting access to the services (finding and obtaining the protocols, access methods, “drivers” and other code necessary to utilize the service), controlling access to the services, advertising the services, choosing among competing services, billing for services, and so on. This problem is widely recognized; many companies, standards bodies and consortia are addressing it at various levels in various ways. Service Location Protocol (SLP), Jini™, and Salutation™, to name just a few, all address some aspect of service discovery.

A.2. Bluetooth Service Discovery

Bluetooth Service Discovery Protocol (SDP) addresses service discovery specifically for the Bluetooth environment. It is optimized for the highly dynamic nature of Bluetooth communications. SDP focuses primarily on discovering services available from or through Bluetooth devices. SDP does not define methods for accessing services; once services are discovered with SDP, they can be accessed in various ways, depending upon the service. This might include the use of other service discovery and access mechanisms such as those mentioned above; SDP provides a means for other protocols to be used along with SDP in those environments where this can be beneficial. While SDP can coexist with other service discovery protocols, it does not require them. In Bluetooth environments, services can be discovered using SDP and can be accessed using other protocols defined by Bluetooth.

APPENDIX B – EXAMPLE SDP TRANSACTIONS

The following are simple examples of typical SDP transactions. These are meant to be illustrative of SDP flows. The examples do not consider:

- Caching (in a caching system, the SDP client would make use of the ServiceRecordState and ServiceDatabaseState attributes);
- Service availability (if this is of interest, the SDP client should use the ServiceAvailability and/or ServiceTimeToLive attributes);
- SDP versions (the VersionNumberList attribute could be used to determine compatible SDP versions);
- SDP Error Responses (an SDP error response is possible for any SDP request that is in error); and
- Communication connection (the examples assume that an L2CAP connection is established).

The examples are meant to be illustrative of the protocol. The format used is `ObjectName[ObjectSizeInBytes] {SubObjectDefinitions}`, but this is not meant to illustrate an interface. The `ObjectSizeInBytes` is the size of the object in decimal. The `SubObjectDefinitions` (inside of `{}` characters) are components of the immediately enclosing object. Hexadecimal values shown as lower-case letters, such as for transaction IDs and service handles, are variables (the particular value is not important for the illustration, but each such symbol always represents the same value). Comments are included in this manner: `/* comment text */`.

Numeric values preceded by “0x” are hexadecimal, while those preceded by “0b” are binary. All other numeric values are decimal.

B.1. SDP Example 1 – ServiceSearchRequest

The first example is that of an SDP client searching for a generic printing service. The client does not specify a particular type of printing service. In the example, the SDP server has two available printing services. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceSearchRequest, specifying the PrinterServiceClassID (represented as a DataElement with a 32-bit UUID value of `ppp...ppp`) as the only element of the ServiceSearchPattern. The PrinterServiceClassID is assumed to be a 32-bit UUID and the data element type for it is illustrated. The TransactionID is illustrated as `tttt`.
2. SDP server to SDP client: SDP_ServiceSearchResponse, returning handles to two printing services, represented as `qqqqqqqq` for the first printing service and `rrrrrrrr` for the second printing service. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (`tttt`).

```
/* Sent from SDP Client to SDP server */
SDP_ServiceSearchRequest[15] {
  PDUID[1] {
    0x02
  }
  TransactionID[2] {
    0xtttt
  }
  ParameterLength[2] {
    0x000A
  }
  ServiceSearchPattern[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* PrinterServiceClassID */
        0b00011 0b010 0xpppppppp
      }
    }
  }
  MaximumServiceRecordCount[2] {
    0x0003
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}


/* Sent from SDP server to SDP client */
SDP_ServiceSearchResponse[18] {
  PDUID[1] {
    0x03
  }
  TransactionID[2] {
    0xtttt
  }
  ParameterLength[2] {
    0x000D
  }
  TotalServiceRecordCount[2] {
    0x0002
  }
  CurrentServiceRecordCount[2] {
    0x0002
  }
  ServiceRecordHandleList[8] {
    /* print service 1 handle */
    0xqqqqqqqq
    /* print service 2 handle */
    0xrrrrrrrr
  }
  ContinuationState[1] {
    /* no continuation state */
```

```
    0x00
  }
}
```

B.2. SDP Example 2 – ServiceAttributeTransaction

The second example continues the first example. In Example 1, the SDP client obtained handles to two printing services. In Example 2, the client uses one of those service handles to obtain the ProtocolDescriptorList attribute for that printing service. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceAttributeRequest, presenting the previously obtained service handle (the one denoted as qqqqqqqq) and specifying the ProtocolDescriptorList attribute ID (AttributeID 0x0004) as the only attribute requested (other attributes could be retrieved in the same transaction if desired). The TransactionID is illustrated as uuuu to distinguish it from the TransactionID of Example 1.
2. SDP server to SDP client: SDP_ServiceAttributeResponse, returning the ProtocolDescriptorList for the specified printing service. This protocol stack is assumed to be ((L2CAP), (RFCOMM, 2), (PostscriptStream)). The ProtocolDescriptorList is a data element sequence in which each element is, in turn, a data element sequence whose first element is a UUID representing the protocol, and whose subsequent elements are protocol-specific parameters. In this example, one such parameter is included for the RFCOMM protocol, an 8-bit value indicating RFCOMM server channel 2. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (uuuu). The Attributes returned are illustrated as a data element sequence where the protocol descriptors are 32-bit UUIDs and the RFCOMM server channel is a data element with an 8-bit value of 2.

```
 /* Sent from SDP Client to SDP server */
SDP_ServiceAttributeRequest[17] {
  PDUID[1] {
    0x04
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x000C
  }
  ServiceRecordHandle[4] {
    0xqqqqqqqq
  }
  MaximumAttributeByteCount[2] {
    0x0080
  }
  AttributeIDList[5] {
    DataElementSequence[5] {
```

```
      0b00110 0b101 0x03
      AttributeID[3] {
        0b00001 0b001 0x0004
      }
    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}

/* Sent from SDP server to SDP client */
SDP_ServiceAttributeResponse[38] {
  PDUID[1] {
    0x05
  }
  TransactionID[2] {
    0xuuuu
  }
  ParameterLength[2] {
    0x0021
  }
  AttributeListByteCount[2] {
    0x001E
  }
  AttributeList[30] {
    DataElementSequence[30] {
      0b00110 0b101 0x1C
      Attribute[28] {
        AttributeID[3] {
          0b00001 0b001 0x0004
        }
        AttributeValue[25] {
          /* ProtocolDescriptorList */
          DataElementSequence[25] {
            0b00110 0b101 0x17
            /* L2CAP protocol descriptor */
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* L2CAP Protocol UUID */
                0b00011 0b010 <32-bit L2CAP UUID>
              }
            }
            /* RFCOMM protocol descriptor */
            DataElementSequence[9] {
              0b00110 0b101 0x07
              UUID[5] {
                /* RFCOMM Protocol UUID */
                0b00011 0b010 <32-bit RFCOMM UUID>
              }
              /* parameter for server 2 */
              Uint8[2] {
                0b00001 0b000 0x02
              }
            }
```

```
            /* PostscriptStream protocol descriptor */
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* PostscriptStream Protocol UUID */
                0b00011 0b010 <32-bit PostscriptStream UUID>
              }
            }
          }
        }
      }
    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}
```

B.3. SDP Example 3 – ServiceSearchAttributeTransaction

The third example is a form of service browsing, although it is not generic browsing in that it does not make use of SDP browse groups. Instead, an SDP client is searching for available Synchronization services that can be presented to the user for selection. The SDP client does not specify a particular type of synchronization service. In the example, the SDP server has three available synchronization services: an address book synchronization service and a calendar synchronization service (both from the same provider), and a second calendar synchronization service from a different provider. The SDP client is retrieving the same attributes for each of these services; namely, the data formats supported for the synchronization service (vCard, vCal, ICal, etc.) and those attributes that are relevant for presenting information to the user about the services. Also assume that the maximum size of a response is 400 bytes. Since the result is larger than this, the SDP client will repeat the request supplying a continuation state parameter to retrieve the remainder of the response. The transaction illustrates:

1. SDP client to SDP server: SDP_ServiceSearchAttributeRequest, specifying the generic SynchronisationServiceClassID (represented as a data element whose 32-bit UUID value is `sss...sss`) as the only element of the ServiceSearchPattern. The SynchronisationServiceClassID is assumed to be a 32-bit UUID. The requested attributes are the ServiceRecordHandle (attribute ID 0x0000), ServiceClassIDList (attribute ID 0x0001), IconURL (attribute ID 0x000C), ServiceName (attribute ID 0x0100), ServiceDescription (attribute ID 0x0101), and ProviderName (attributeID 0x0102) attributes; as well as the service-specific SupportedDataStores (AttributeID 0x0301). Since the first two attribute IDs (0x0000 and 0x0001) and three other attribute IDs(0x0100, 0x0101, and 0x0102 are consecutive, they are specified as attribute ranges. The TransactionID is illustrated as `vvvv` to distinguish it from the TransactionIDs of the other Examples.

Note that values in the service record's primary language are requested for the text attributes (ServiceName, ServiceDescription and ProviderName) so that absolute attribute IDs may be used, rather than adding offsets to a base obtained from the LanguageBaseAttributeIDList attribute.

2. SDP server to SDP client: SDP_ServiceSearchAttributeResponse, returning the specified attributes for each of the three synchronization services. In the example, each ServiceClassIDList is assumed to contain a single element, the generic SynchronisationServiceClassID (a 32-bit UUID represented as sss...sss). Each of the other attributes contain illustrative data in the example (the strings have illustrative text; the icon URLs are illustrative, for each of the respective three synchronization services; and the SupportedDataStore attribute is represented as an unsigned 8-bit integer where 0x01 = vCard2.1, 0x02 = vCard3.0, 0x03 = vCal1.0 and 0x04 = iCal). Note that one of the service records (the third for which data is returned) has no ServiceDescription attribute. The attributes are returned as a data element sequence, where each element is in turn a data element sequence representing a list of attributes. Within each attribute list, the ServiceClassIDList is a data element sequence while the remaining attributes are single data elements. The Transaction ID is the same value as supplied by the SDP client in the corresponding request (0xvvvv). Since the entire result cannot be returned in a single response, a non-null continuation state is returned in this first response.

   Note that the total length of the initial data element sequence (487 in the example) is indicated in the first response, even though only a portion of this data element sequence (368 bytes in the example, as indicated in the AttributeLists byte count) is returned in the first response. The remainder of this data element sequence is returned in the second response (without an additional data element header).

3. SDP client to SDP server: SDP_ServiceSearchAttributeRequest, with the same parameters as in step 1, except that the continuation state received from the server in step 2 is included as a request parameter. The TransactionID is changed to 0xwww to distinguish it from previous request.
4. SDP server to SDP client: SDP_ServiceSearchAttributeResponse, with the remainder of the result computed in step 2 above. Since all of the remaining result fits in this second response, a null continuation state is included.

```
/* Part 1 -- Sent from SDP Client to SDP server */
SdpSDP_ServiceSearchAttributeRequest[33] {
  PDUID[1] {
    0x06
  }
  TransactionID[2] {
    0xvvvv
  }
  ParameterLength[2] {
    0x001B
  }
  ServiceSearchPattern[7] {
```

```
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* SynchronisationServiceClassID */
        0b00011 0b010 0xssssssss
      }
    }
  }
  MaximumAttributeByteCount[2] {
    0x0190
  }
AttributeIDList[18] {
    DataElementSequence[18] {
      0b00110 0b101 0x10
      AttributeIDRange[5] {
        0b00001 0b010 0x00000001
      }
      AttributeID[3] {
        0b00001 0b001 0x000C
      }
      AttributeIDRange[5] {
        0b00001 0b010 0x01000102
      }
      AttributeID[3] {
        0b00001 0b001 0x0301
      }    }
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}

/* Part 2 -- Sent from SDP server to SDP client */
SdpSDP_ServiceSearchAttributeResponse[384] {
  PDUID[1] {
    0x07
  }
  TransactionID[2] {
    0xvvvv
  }
  ParameterLength[2] {
    0x017B
  }
  AttributeListByteCount[2] {
    0x0170
  }
  AttributeLists[368] {
    DataElementSequence[487] {
      0b00110 0b110 0x01E4
      DataElementSequence[178] {
        0b00110 0b101 0xB0
        Attribute[8] {
          AttributeID[3] {
            0b00001 0b001 0x0000
          }
          AttributeValue[5] {
```

```
            /* service record handle */
            0b00001 0b010 0xhhhhhhhh
          }
        }
        Attribute[10] {
          AttributeID[3] {
            0b00001 0b001 0x0001
          }
AttributeValue[7] {
            DataElementSequence[7] {
              0b00110 0b101 0x05
              UUID[5] {
                /* SynchronisationServiceClassID */
                0b00011 0b010 0xssssssss
              }
            }
          }
        }
        Attribute[35] {
          AttributeID[3] {
            0b00001 0b001 0x000C
          }
          AttributeValue[32] {
            /* IconURL; '*' replaced by client application */
            0b01000 0b101 0x1E
            "http://Synchronisation/icons/*"
          }
        }
        Attribute[22] {
          AttributeID[3] {
            0b00001 0b001 0x0100
          }
          AttributeValue[19] {
            /* service name */
            0b00100 0b101 0x11
            "Address Book Sync"
          }
        }
        Attribute[59] {
          AttributeID[3] {
            0b00001 0b001 0x0101
          }
          AttributeValue[56] {
            /* service description */
            0b00100 0b101 0x36
            "Synchronisation Service for"
            " vCard Address Book Entries"
          }
        }
        Attribute[37] {
          AttributeID[3] {
            0b00001 0b001 0x0102
          }
          AttributeValue[34] {
            /* service provider */
            0b00100 0b101 0x20
            "Synchronisation Specialists Inc."
```

```
        }
      }
      Attribute[5] {
        AttributeID[3] {
          0b00001 0b001 0x0301
        }
        AttributeValue[2] {
          /* Supported Data Store 'phonebook' */
          0b00001 0b000 0x01
        }
      }
    }
    DataElementSequence[175] {
      0b00110 0b101 0xAD
      Attribute[8] {
        AttributeID[3] {
          0b00001 0b001 0x0000
        }
        AttributeValue[5] {
          /* service record handle */
          0b00001 0b010 0xmmmmmmmm
        }
      }
      Attribute[10] {
        AttributeID[3] {
          0b00001 0b001 0x0001
        }
                  AttributeValue[7] {
        DataElementSequence[7] {
          0b00110 0b101 0x05
          UUID[5] {
            /* SynchronisationServiceClassID */
            0b00011 0b010 0xssssssss
          }
        }
      }
      Attribute[35] {
        AttributeID[3] {
          0b00001 0b001 0x000C
        }
        AttributeValue[32] {
          /* IconURL; '*' replaced by client application */
          0b01000 0b101 0x1E
          "http://Synchronisation/icons/*"
        }
      }
      Attribute[21] {
        AttributeID[3] {
          0b00001 0b001 0x0100
        }
        AttributeValue[18] {
          /* service name */
          0b00100 0b101 0x10
          "Appointment Sync"
        }
      }
```

```
        Attribute[57] {
          AttributeID[3] {
            0b00001 0b001 0x0101
          }
          AttributeValue[54] {
            /* service description */
            0b00100 0b101 0x34
            "Synchronisation Service for"
            " vCal Appointment Entries"
          }
        }
        Attribute[37] {
          AttributeID[3] {
            0b00001 0b001 0x0102
          }
          AttributeValue[34] {
            /* service provider */
            0b00100 0b101 0x20
            "Synchronisation Specialists Inc."
          }
        }
        Attribute[5] {
          AttributeID[3] {
            0b00001 0b001 0x0301
          }
          AttributeValue[2] {
            /* Supported Data Store 'calendar' */
            0b00001 0b000 0x03
          }
        }
      }
    /* } Data element sequence of attribute lists */
    /* is not completed in this PDU. */
  }
  ContinuationState[9] {
    /* 8 bytes of continuation state */
    0x08 0xzzzzzzzzzzzzzzzz
  }
}

/* Part 3 -- Sent from SDP Client to SDP server */
SdpSDP_ServiceSearchAttributeRequest[41] {
  PDUID[1] {
    0x06
  }
  TransactionID[2] {
    0xwwww
  }
  ParameterLength[2] {
    0x0024
  }
  ServiceSearchPattern[7] {
    DataElementSequence[7] {
      0b00110 0b101 0x05
      UUID[5] {
        /* SynchronisationServiceClassID */
        0b00011 0b010 0xssssssss
```

```
      }
    }
  }
  MaximumAttributeByteCount[2] {
    0x0180
  }
    AttributeIDList[18] {
    DataElementSequence[18] {
      0b00110 0b101 0x10
      AttributeIDRange[5] {
        0b00001 0b010 0x00000001
      }
      AttributeID[3] {
        0b00001 0b001 0x000C
      }
      AttributeIDRange[5] {
        0b00001 0b010 0x01000102
      }
      AttributeID[3] {
        0b00001 0b001 0x0301
      }
    }
  }
  ContinuationState[9] {
    /* same 8 bytes of continuation state */
    /* received in part 2 */
    0x08 0xzzzzzzzzzzzzzzzz
  }
}

Part 4 -- Sent from SDP server to SDP client

SdpSDP_ServiceSearchAttributeResponse[115] {
  PDUID[1] {
    0x07
  }
  TransactionID[2] {
    0xwwww
  }
  ParameterLength[2] {
    0x006E
  }
  AttributeListByteCount[2] {
    0x006B
  }
  AttributeLists[107] {
    /* Continuing the data element sequence of */
    /* attribute lists begun in Part 2. */
      DataElementSequence[107] {
        0b00110 0b101 0x69
        Attribute[8] {
          AttributeID[3] {
            0b00001 0b001 0x0000
          }
          AttributeValue[5] {
            /* service record handle */
            0b00001 0b010 0xffffffff
```

```
        }
      }
      Attribute[10] {
        AttributeID[3] {
          0b00001 0b001 0x0001
        }
        AttributeValue[7] {
          DataElementSequence[7] {
            0b00110 0b101 0x05
            UUID[5] {
              /* SynchronisationServiceClassID */
              0b00011 0b010 0xssssssss
            }
          }
        }
      }
      Attribute[35] {
        AttributeID[3] {
          0b00001 0b001 0x000C
        }
        AttributeValue[32] {
          /* IconURL; '*' replaced by client application */
          0b01000 0b101 0x1E
          "http://DevManufacturer/icons/*"
        }
      }
      Attribute[18] {
        AttributeID[3] {
          0b00001 0b001 0x0100
        }
        AttributeValue[15] {
          /* service name */
          0b00100 0b101 0x0D
          "Calendar Sync"
        }
      }
      Attribute[29] {
        AttributeID[3] {
          0b00001 0b001 0x0102
        }
        AttributeValue[26] {
          /* service provider */
          0b00100 0b101 0x18
          "Device Manufacturer Inc."
        }
      }
      Attribute[5] {
        AttributeID[3] {
          0b00001 0b001 0x0301
        }
        AttributeValue[2] {
          /* Supported Data Store 'calendar' */
          0b00001 0b000 0x03
        }
      }
    }
  /* This completes the data element sequence */
```

```
    /* of attribute lists begun in Part 2.
  }
  ContinuationState[1] {
    /* no continuation state */
    0x00
  }
}
```

GENERIC ACCESS PROFILE

This profile defines the generic procedures related to discovery of Bluetooth devices (idle mode procedures) and link management aspects of connecting to Bluetooth devices (connecting mode procedures). It also defines procedures related to use of different security levels. In addition, this profile includes common format requirements for parameters accessible on the user interface level.

CONTENTS

FOREWORD

Interoperability between devices from different manufacturers is provided for a specific service and use case, if the devices conform to a Bluetooth SIG-defined profile specification. A profile defines a selection of messages and procedures (generally termed *capabilities*) from the Bluetooth SIG specifications and gives a description of the air interface for specified service(s) and use case(s).

All defined features are process-mandatory. This means that, if a feature is used, it is used in a specified manner. Whether the provision of a feature is mandatory or optional is stated separately for both sides of the Bluetooth air interface.

1 INTRODUCTION

1.1 SCOPE

The purpose of the Generic Access Profile is:

To introduce definitions, recommendations and common requirements related to modes and access procedures that are to be used by transport and application profiles.

To describe how devices are to behave in standby and connecting states in order to guarantee that links and channels always can be established between Bluetooth devices, and that multi-profile operation is possible. Special focus is put on discovery, link establishment and security procedures.

To state requirements on user interface aspects, mainly coding schemes and names of procedures and parameters, that are needed to guarantee a satisfactory user experience.

1.2 SYMBOLS AND CONVENTIONS

1.2.1 Requirement status symbols

In this document (especially in the profile requirements tables), the following symbols are used:

‘M’ for mandatory to support (used for capabilities that shall be used in the profile);

’O’ for optional to support (used for capabilities that can be used in the profile);

‘C’ for conditional support (used for capabilities that shall be used in case a certain other capability is supported);

‘X’ for excluded (used for capabilities that may be supported by the unit but shall never be used in the profile);

’N/A’ for not applicable (in the given context it is impossible to use this capability).

Some excluded capabilities are capabilities that, according to the relevant Bluetooth specification, are mandatory. These are features that may degrade operation of devices following this profile. Therefore, these features shall never be activated while a unit is operating as a unit within this profile.

In this specification, the word *shall* is used for mandatory requirements, the word *should* is used to express recommendations and the word *may* is used for options.

1.2.2 Signaling diagram conventions

The following arrows are used in diagrams describing procedures
:

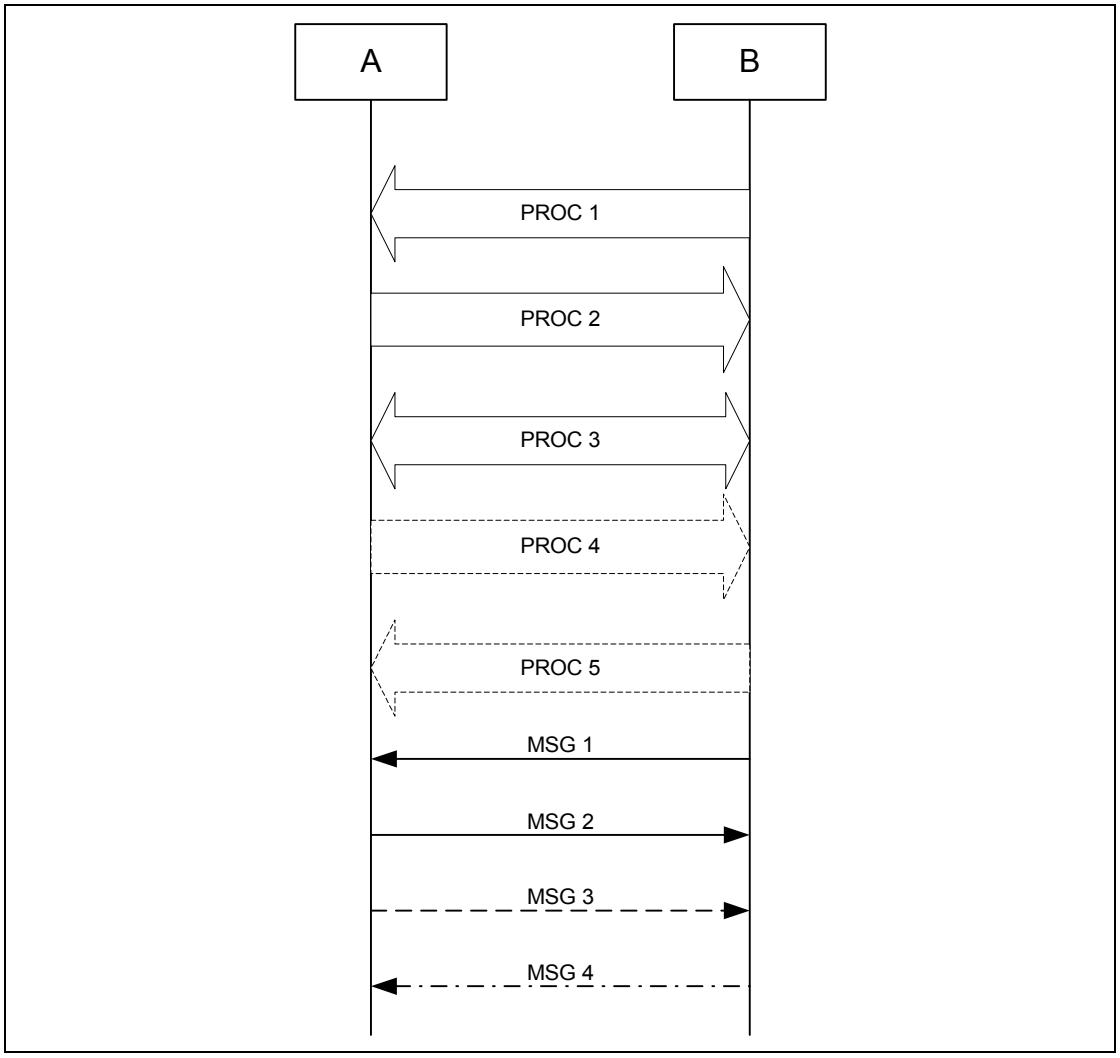


Figure 1.1: Arrows used in signaling diagrams

In the figure above, the following cases are shown: PROC1 is a sub-procedure initiated by B. PROC2 is a sub-procedure initiated by A. PROC3 is a sub-procedure where the initiating side is undefined (may be both A or B). Dashed arrows denote optional steps. PROC4 indicates an optional sub-procedure initiated by A, and PROC5 indicates an optional sub-procedure initiated by B.

MSG1 is a message sent from B to A. MSG2 is a message sent from A to B. MSG3 indicates an optional message from A to B, and MSG4 indicates a conditional message from B to A.

1.2.3 Notation for timers and counters

Timers are introduced specific to this profile. To distinguish them from timers used in the Bluetooth protocol specifications and other profiles, these timers are named in the following format: 'T_{GAP}*(nnn)*'.

2 PROFILE OVERVIEW

2.1 PROFILE STACK

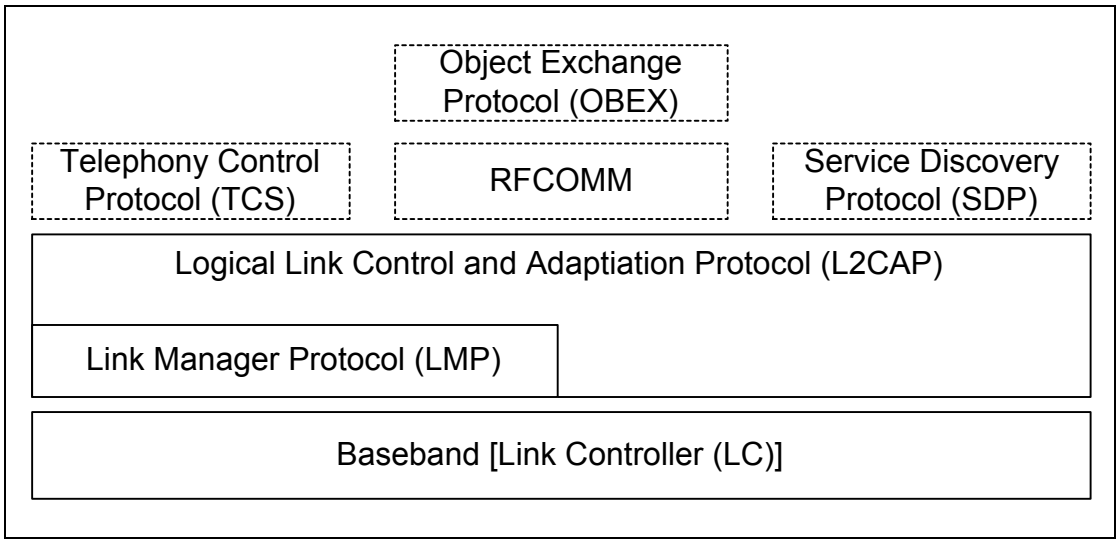


Figure 2.1: Profile stack covered by this profile.

The main purpose of this profile is to describe the use of the lower layers of the Bluetooth protocol stack (LC and LMP). To describe security related alternatives, also higher layers (L2CAP, RFCOMM and OBEX) are included.

2.2 CONFIGURATIONS AND ROLES

For the descriptions in this profile of the roles that the two devices involved in a Bluetooth communication can take, the generic notation of the A-party (the *paging device* in case of link establishment, or *initiator* in case of another procedure on an established link) and the B-party (*paged device* or *acceptor*) is used. The A-party is the one that, for a given procedure, initiates the establishment of the physical link or initiates a transaction on an existing link.

This profile handles the procedures between two devices related to discovery and connecting (link and connection establishment) for the case where none of the two devices has any link established as well as the case where (at least) one device has a link established (possibly to a third device) before starting the described procedure.

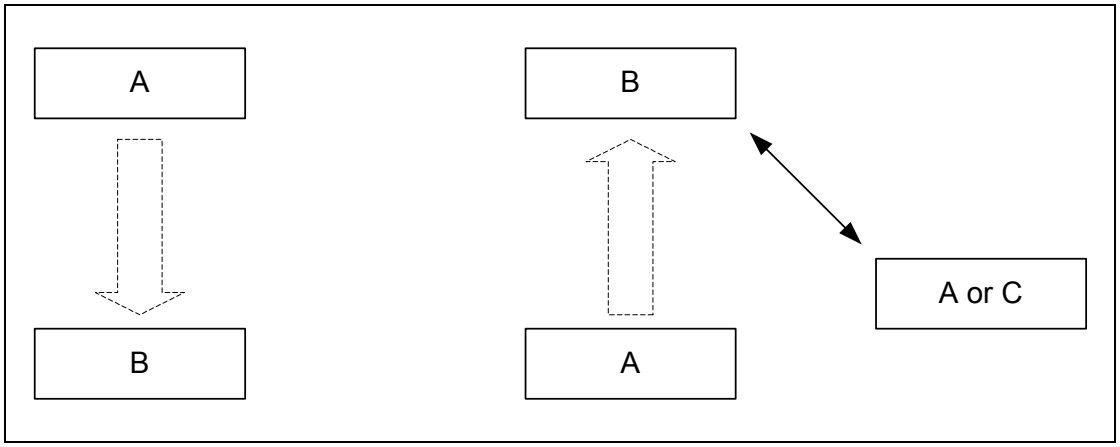


Figure 2.2: This profile covers procedures initiated by one device (A) towards another device (B) that may or may not have an existing Bluetooth link active.

The initiator and the acceptor generally operate the generic procedures according to this profile or another profile referring to this profile. If the acceptor operates according to several profiles simultaneously, this profile describes generic mechanisms for how to handle this.

2.3 USER REQUIREMENTS AND SCENARIOS

The Bluetooth user should in principle be able to connect a Bluetooth device to any other Bluetooth device. Even if the two connected devices don't share any common application, it should be possible for the user to find this out using basic Bluetooth capabilities. When the two devices do share the same application but are from different manufacturers, the ability to connect them should not be blocked just because manufacturers choose to call basic Bluetooth capabilities by different names on the user interface level or implement basic procedures to be executed in different orders.

2.4 PROFILE FUNDAMENTALS

This profile states the requirements on names, values and coding schemes used for names of parameters and procedures experienced on the user interface level.

This profile defines modes of operation that are not service- or profile-specific, but that are generic and can be used by profiles referring to this profile, and by devices implementing multiple profiles.

This profile defines the general procedures that can be used for discovering identities, names and basic capabilities of other Bluetooth devices that are in a mode where they can be discoverable. Only procedures where no channel or connection establishment is used are specified.

This profile defines the general procedure for how to create bonds (i.e. dedicated exchange of link keys) between Bluetooth devices.

This profile describes the general procedures that can be used for establishing connections to other Bluetooth devices that are in mode that allows them to accept connections and service requests.

2.5 CONFORMANCE

Bluetooth devices that do not conform to any other Bluetooth profile shall conform to this profile to ensure basic interoperability.

Bluetooth devices that conform to another Bluetooth profile may use adaptations of the generic procedures as specified by that other profile. They shall, however, be compatible with devices compliant to this profile at least on the level of the supported generic procedures.

If conformance to this profile is claimed, all capabilities indicated mandatory for this profile shall be supported in the specified manner (process-mandatory). This also applies for all optional and conditional capabilities for which support is indicated. All mandatory capabilities, and optional and conditional capabilities for which support is indicated, are subject to verification as part of the Bluetooth certification program.

3 USER INTERFACE ASPECTS

3.1 THE USER INTERFACE LEVEL

In the context of this specification, the user interface level refers to places (such as displays, dialog boxes, manuals, packaging, advertising, etc.) where users of Bluetooth devices encounters names, values and numerical representation of Bluetooth terminology and parameters.

This profile specifies the generic terms that should be used on the user interface level.

3.2 REPRESENTATION OF BLUETOOTH PARAMETERS

3.2.1 Bluetooth device address (BD_ADDR)

3.2.1.1 Definition

BD_ADDR is the address used by a Bluetooth device as defined in [1]. It is received from a remote device during the device discovery procedure.

3.2.1.2 Term on user interface level

When the Bluetooth address is referred to on UI level, the term ’Bluetooth Device Address’ should be used.

3.2.1.3 Representation

On BB level the BD_ADDR is represented as 48 bits [1].

On the UI level the Bluetooth address shall be represented as 12 hexadecimal characters, possibly divided into sub-parts separated by’:’.
(E.g., ’000C3E3A4B69’ or ’00:0C:3E:3A:4B:69’.) At UI level, any number shall have the MSB -> LSB (from left to right) ’natural’ ordering (e.g., the number ’16’ shall be shown as ’0x10’).

3.2.2 Bluetooth device name (the user-friendly name)

3.2.2.1 Definition

The Bluetooth device name is the user-friendly name that a Bluetooth device presents itself with. It is a character string returned in LMP_name_res as response to a LMP_name_req.

3.2.2.2 Term on user interface level

When the Bluetooth device name is referred to on UI level, the term ’Bluetooth Device Name’ should be used.

3.2.2.3 Representation

The Bluetooth device name can be up to 248 bytes maximum according to [2]. It shall be encoded according to UTF-8 (i.e. name entered on UI level may be down to 82 characters outside the Unicode range 0x00-0x7F are used).

A device can not expect that a general remote device is able to handle more than the first 40 characters of the Bluetooth device name. If a remote device has limited display capabilities, it may use only the first 20 characters.

3.2.3 Bluetooth passkey (Bluetooth PIN)

3.2.3.1 Definition

The Bluetooth PIN is used to authenticate two Bluetooth devices (that have not previously exchanged link keys) to each other and create a trusted relationship between them. The PIN is used in the pairing procedure (see Section 10.2 on page 224) to generate the initial link key that is used for further authentication.

The PIN may be entered on UI level but may also be stored in the device; e.g. in the case of a device without sufficient MMI for entering and displaying digits.

3.2.3.2 Terms at user interface level

When the Bluetooth PIN is referred to on UI level, the term ’Bluetooth Passkey’ should be used.

3.2.3.3 Representation

The Bluetooth PIN has different representations on different levels. PINBB is used on baseband level, and PINUI is used on user interface level. PINBB is the PIN used by [1] for calculating the initialization key during the pairing procedure. PINUI is the character representation of the PIN that is entered on UI level. The transformation from PINUI to PINBB shall be according to UTF-8. According to [1], PINBB can be 128 bits (16 bytes).

PIN codes may be up to 16 characters. In order to take advantage of the full level of security all PINs should be 16 characters long. Variable PINs should be composed of alphanumeric characters chosen from within the Unicode range 0x00-0x7F. If the PIN contains any decimal digits these shall be encoded using the Unicode Basic Latin characters (i.e. code points 0x30 to 0x39) (Note 1).

For compatibility with devices with numeric keypads fixed PINs shall be composed of only decimal digits, and variable PINS may be composed of only decimal digits.

If a device supports entry of characters outside the Unicode range 0x00-0x7F other Unicode code points may be used (Note 2), except the Halfwidth and Fullwidth Forms from within the Unicode range FF00 - FFEF shall not be used (Note 3).

Examples:

User-entered code	**Corresponding PIN_{BB}[0..length-1] (value as a sequence of octets in hexadecimal notation)**
’0196554200906493’	length = 16, value = 0x30 0x31 0x39 0x36 0x35 0x35 0x34 0x32 0x30 0x30 0x39 0x30 0x36 0x34 0x39 0x33
’Børnelitteratur’	length = 16, value = 0x42 0xC3 0xB8 0x72 0x6e 0x65 0x6c 0x69 0x74 0x74 0x65 0x72 0x61 0x74 0x75 0x72

Note 1: This is to prevent interoperability problems since there are decimal digits at other code points (e.g. the Fullwidth digits at code points 0xff10 to 0xff19).

Note 2: Unicode characters outside the Basic Latin range (0x00 - 0x7F) encode to multiple bytes, therefore when characters outside the Basic Latin range are used the maximum number of characters in the PINUI will be less than 16. The second example illustrates a case where a 16 character string encodes to 15 bytes because the character ø is outside the Basic Latin range and encodes to two bytes (0xC3 0xB8).

Note 3: This is to prevent interoperability problems since the Halfwidth and Fullwidth forms contain alternative variants of ASCII, Katakana, Hangul, punctuation and symbols. All of the characters in the Halfwidth and Fullwidth forms have other more commonly used Unicode code points.

3.2.4 Class of Device

3.2.4.1 Definition

Class of device is a parameter received during the device discovery procedure, indicating the type of device and which types of service that are supported.

3.2.4.2 Term on user interface level

The information within the Class of Device parameter should be referred to as ’Bluetooth Device Class’ (i.e. the major and minor device class fields) and ’Bluetooth Service Type’ (i.e. the service class field). The terms for the defined Bluetooth Device Types and Bluetooth Service Types are defined in [11].

When using a mix of information found in the Bluetooth Device Class and the Bluetooth Service Type, the term ’Bluetooth Device Type’ should be used.

3.2.4.3 Representation

The Class of device is a bit field and is defined in [11]. The UI-level representation of the information in the Class of device is implementation specific.

3.3 PAIRING

Two procedures are defined that make use of the pairing procedure defined on LMP level (LMP-pairing, see Section 10.2 on page 224). Either the user initiates the bonding procedure and enters the passkey with the explicit purpose of creating a bond (and maybe also a secure relationship) between two Bluetooth devices, or the user is requested to enter the passkey during the establishment procedure since the devices did not share a common link key beforehand. In the first case, the user is said to perform ’bonding (with entering of passkey)’ and in the second case the user is said to ’authenticate using the passkey’.

4 MODES

Procedure	Ref.	Support
Discoverability modes:	4.1	
Non-discoverable mode		C1
Limited discoverable mode		O
General discoverable mode		O
Connectability modes:	4.1.3.3	
Non-connectable mode		O
Connectable mode		M
Pairing modes:	4.2.2.2	
Non-pairable mode		O
Pairable mode		C2
C1: If limited discoverable mode is supported, non-discoverable mode is mandatory, otherwise optional.		
C2: If the bonding procedure is supported, support for pairable mode is mandatory, otherwise optional.		

Table 4.1: Conformance requirements related to modes defined in this section

4.1 DISCOVERABILITY MODES

With respect to inquiry, a Bluetooth device shall be either in non-discoverable mode or in a discoverable mode. (The device shall be in one, and only one, discoverability mode at a time.) The two discoverable modes defined here are called limited discoverable mode and general discoverable mode. Inquiry is defined in [1].

When a Bluetooth device is in non-discoverable mode it does not respond to inquiry.

A Bluetooth device is said to be made discoverable, or set into a discoverable mode, when it is in limited discoverable mode or in general discoverable mode. Even when a Bluetooth device is made discoverable it may be unable to respond to inquiry due to other baseband activity [1]. A Bluetooth device that does not respond to inquiry for any of these two reasons is called a silent device.

After being made discoverable, the Bluetooth device shall be discoverable for at least $T_{GAP}(103)$.

The speed of discovery is dependent on the configuration of the inquiry scan interval and inquiry scan type of the Bluetooth device. The Host is able to configure these parameters based on trade-offs between power consumption, bandwidth and the desired speed of discovery.

4.1.1 Non-discoverable mode

4.1.1.1 Definition

When a Bluetooth device is in non-discoverable mode, it shall never enter the INQUIRY_RESPONSE state.

4.1.1.2 Term on UI-level

Bluetooth device is ’non-discoverable’ or in ’non-discoverable mode’.

4.1.2 Limited discoverable mode

4.1.2.1 Definition

The limited discoverable mode should be used by devices that need to be discoverable only for a limited period of time, during temporary conditions or for a specific event. The purpose is to respond to a device that makes a limited inquiry (inquiry using the LIAC).

A Bluetooth device should not be in limited discoverable mode for more than $T_{GAP}(104)$. The scanning for the limited inquiry access code can be done either in parallel or in sequence with the scanning of the general inquiry access code. When in limited discoverable mode, one of the following options shall be used.

- *Parallel scanning*

  When a Bluetooth device is in limited discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

  If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.

  When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and non-Interlaced Inquiry scan is used.

  In all cases the Bluetooth device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC and the LIAC for at least $T_{GAP}(101)$

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

- *Sequential scanning*

  When a Bluetooth device is in limited discoverable mode, it shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$ and enter the INQUIRY_SCAN state more often than once in $T_{GAP}(102)$ and scan for the LIAC for at least $T_{GAP}(101)$.

  If an inquiry message is received when in limited discoverable mode, the entry into the INQUIRY_RESPONSE state takes precedence over the next entries into INQUIRY_SCAN state until the inquiry response is completed.

4.1.2.2 Conditions

When a device is in limited discoverable mode it shall set bit no 13 in the Major Service Class part of the Class of Device/Service field [11].

4.1.2.3 Term on UI-level

Bluetooth device is ’discoverable’ or in ’discoverable mode’.

4.1.3 General discoverable mode

4.1.3.1 Definition

The general discoverable mode shall be used by devices that need to be discoverable continuously or for no specific condition. The purpose is to respond to a device that makes a general inquiry (inquiry using the GIAC).

4.1.3.2 Conditions

When a Bluetooth device is in general discoverable mode and when discovery speed is more important than power consumption or bandwidth, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(105)$ and that Interlaced Inquiry scan is used.

If, however, power consumption or bandwidth is important, but not critical, it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Interlaced Inquiry scan is used.

When power consumption or bandwidth is critical it is recommended that the Bluetooth device enter the INQUIRY_SCAN state at least every $T_{GAP}(102)$ and Non-interlaced Inquiry scan is used.

In all cases the Bluetooth device shall enter the INQUIRY_SCAN state at least once in $T_{GAP}(102)$ and scan for the GIAC for at least $T_{GAP}(101)$.

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the discoverability time.

A device in general discoverable mode shall not respond to a LIAC inquiry.

4.1.3.3 Term on UI-level

Bluetooth device is ’discoverable’ or in ’discoverable mode’.

4.2 CONNECTABILITY MODES

With respect to paging, a Bluetooth device shall be either in non-connectable mode or connectable mode. Paging is defined in [1].

When a Bluetooth device is in non-connectable mode it does not respond to paging. When a Bluetooth device is in connectable mode it responds to paging.

The speed of connections is dependent on the configuration of the page scan interval and page scan type of the Bluetooth device. The Host is able to configure these parameters based on trade-offs between power consumption, bandwidth and the desired speed of connection.

4.2.1 Non-connectable mode

4.2.1.1 Definition

When a Bluetooth device is in non-connectable mode it shall never enter the PAGE_SCAN state.

4.2.1.2 Term on UI-level

Bluetooth device is ’non-connectable’ or in ’non-connectable mode’.

4.2.2 Connectable mode

4.2.2.1 Definition

When a Bluetooth device is in connectable mode it shall periodically enter the PAGE_SCAN state. The device makes page scan using the Bluetooth device address, BD_ADDR. Connection speed is a trade-off between power consumption / available bandwidth and speed. The Bluetooth Host is able to make these trade-offs using the Page Scan interval, Page Scan window, and Interlaced Scan parameters.

R0 page scanning should be used when connection speeds are critically important and when the paging device has a very good estimate of the Bluetooth clock. Under these conditions it is possible for paging to complete within two times the page scan window. Because the page scan interval is equal to the page scan window it is not possible for any other traffic to go over the Bluetooth link when using R0 page scanning. In R0 page scanning it is not possible to use interlaced scan. R0 page scanning is the highest power consumption mode of operation.

When connection times are critical but the other device either does not have an estimate of the Bluetooth clock or when the estimate is possibly out of date, it is better to use R1 page scanning with a very short page scan interval,

$T_{GAP}(106)$, and Interlaced scan. This configuration is also useful to achieve nearly the same connection speeds as R0 page scanning but using less power consumption and leaving bandwidth available for other connections. Under these circumstances it is possible for paging to complete within $T_{GAP}(106)$. In this case the Bluetooth device shall page scan for at least $T_{GAP}(101)$.

When connection times are important but not critical enough to sacrifice significant bandwidth and/or power consumption it is recommended to use either $T_{GAP}(107)$ or $T_{GAP}(108)$ for the scanning interval. Using Interlaced scan will reduce the connection time by half but may use twice the power consumption. Under these circumstances it is possible for paging to complete within one or two times the page scanning interval depending on whether Interlaced Scan is used. In this case the Bluetooth device shall page scan for at least $T_{GAP}(101)$.

In all cases the Bluetooth device shall enter the PAGE_SCAN state at least once in $T_{GAP}(102)$ and scan for at least $T_{GAP}(101)$.

The page scan interval, page scan window size and scan type for the six scenarios are described in the following table:

Scenario	Page Scan Interval	Page Scan Window	Scan Type
R0 (1.28s)	$T_{GAP}(107)$	$T_{GAP}(107)$	Normal scan
Fast R1 (100ms)	$T_{GAP}(106)$	$T_{GAP}(101)$	Interlaced scan
Medium R1 (1.28s)	$T_{GAP}(107)$	$T_{GAP}(101)$	Interlaced scan
Slow R1 (1.28s)	$T_{GAP}(107)$	$T_{GAP}(101)$	Normal scan
Fast R2 (2.56s)	$T_{GAP}(108)$	$T_{GAP}(101)$	Interlaced scan
Slow R2 (2.56s)	$T_{GAP}(108)$	$T_{GAP}(101)$	Normal scan

Table 4.2: Page scan parameters for connection speed scenarios

When either a SCO or eSCO link is in operation, it is recommended to use interlaced scan to significantly decrease the connection time.

4.2.2.2 Term on UI-level

Bluetooth device is ’connectable’ or in ’connectable mode’.

4.3 PAIRING MODES

With respect to pairing, a Bluetooth device shall be either in non-pairable mode or in pairable mode. In pairable mode the Bluetooth device accepts paring – i.e. creation of bonds – initiated by the remote device, and in non-pairable mode it does not. Pairing is defined in [1] and [2].

4.3.1 Non-pairable mode

4.3.1.1 Definition

When a Bluetooth device is in non-pairable mode it shall respond to a received LMP_in_rand with LMP_not_accepted with the reason *pairing not allowed*.

4.3.1.2 Term on UI-level

Bluetooth device is ’non-bondable’ or in ’non-bondable mode’ or “does not accept bonding”.

4.3.2 Pairable mode

4.3.2.1 Definition

When a Bluetooth device is in pairable mode it shall respond to a received LMP_in_rand with LMP_accepted (or with LMP_in_rand if it has a fixed PIN).

4.3.2.2 Term on UI-level

Bluetooth device is ’bondable’ or in ’bondable mode’ or “accepts bonding”.

5 SECURITY ASPECTS

	Procedure	Ref.	Support
1	Authentication	5.1	C1
2	Security modes	5.2	
	Security mode 1		O
	Security mode 2		C2
	Security mode 3		C2
C1: If security mode 1 is the only security mode that is supported, support for authentication is optional, otherwise mandatory. (Note: support for LMP-authentication and LMP-pairing is mandatory according [2] independent of which security mode that is used.)			
C2: If secure communication is supported, then support for at least one of Security mode 2 or Security mode 3 is mandatory.			

Table 5.1: Conformance requirements related to the generic authentication procedure and the security modes defined in this section

5.1 AUTHENTICATION

5.1.1 Purpose

The generic authentication procedure describes how the LMP-authentication and LMP-pairing procedures are used when authentication is initiated by one Bluetooth device towards another, depending on if a link key exists or not and if pairing is allowed or not.

5.1.2 Term on UI level

'Bluetooth authentication'.

5.1.3 Procedure

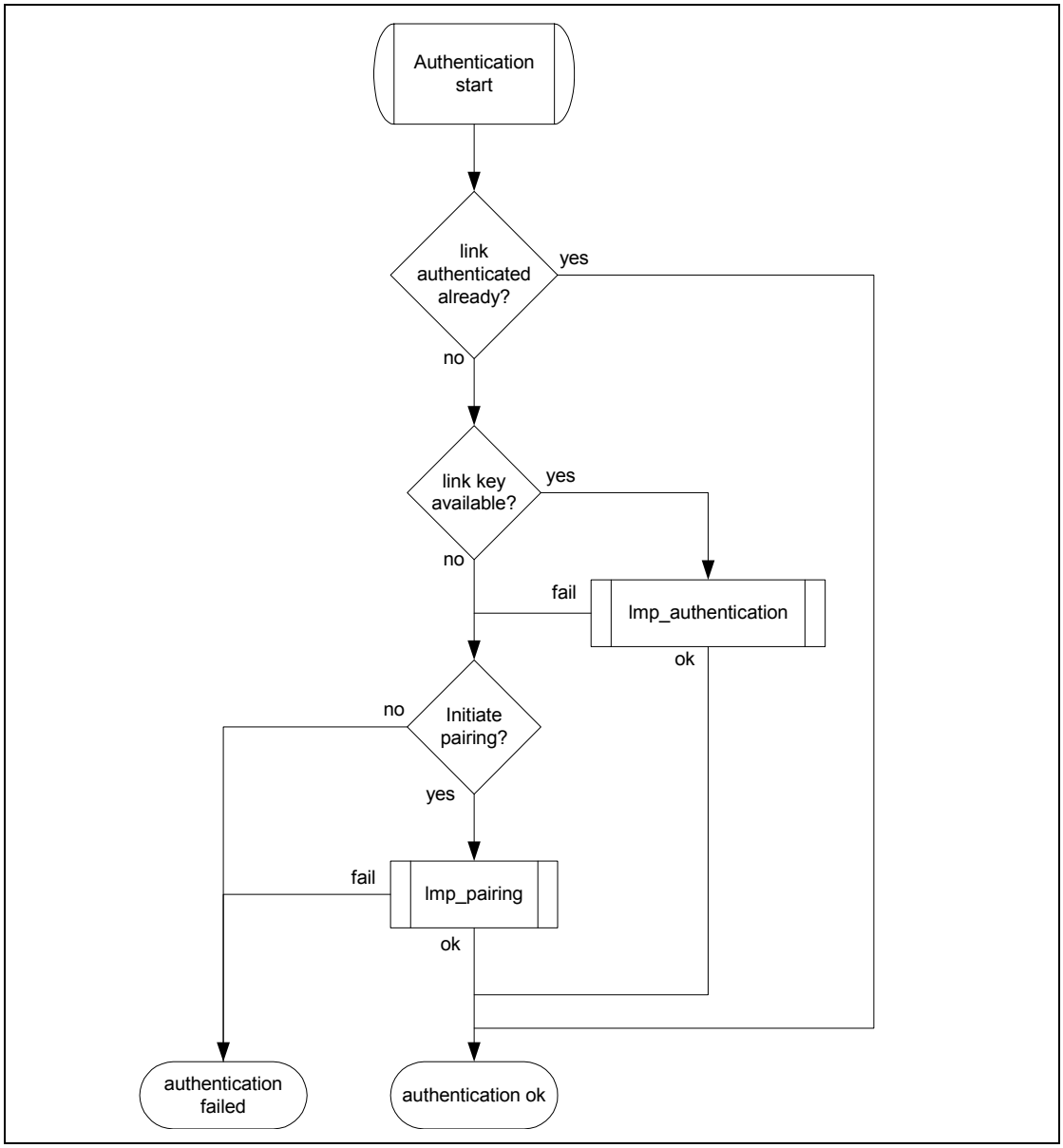


Figure 5.1: Definition of the generic authentication procedure.

5.1.4 Conditions

The device that initiates authentication has to be in security mode 2 or in security mode 3.

5.2 SECURITY MODES

The following flow chart describes where in the channel establishment procedures initiation of authentication takes place, depending on which security mode the Bluetooth device is in.

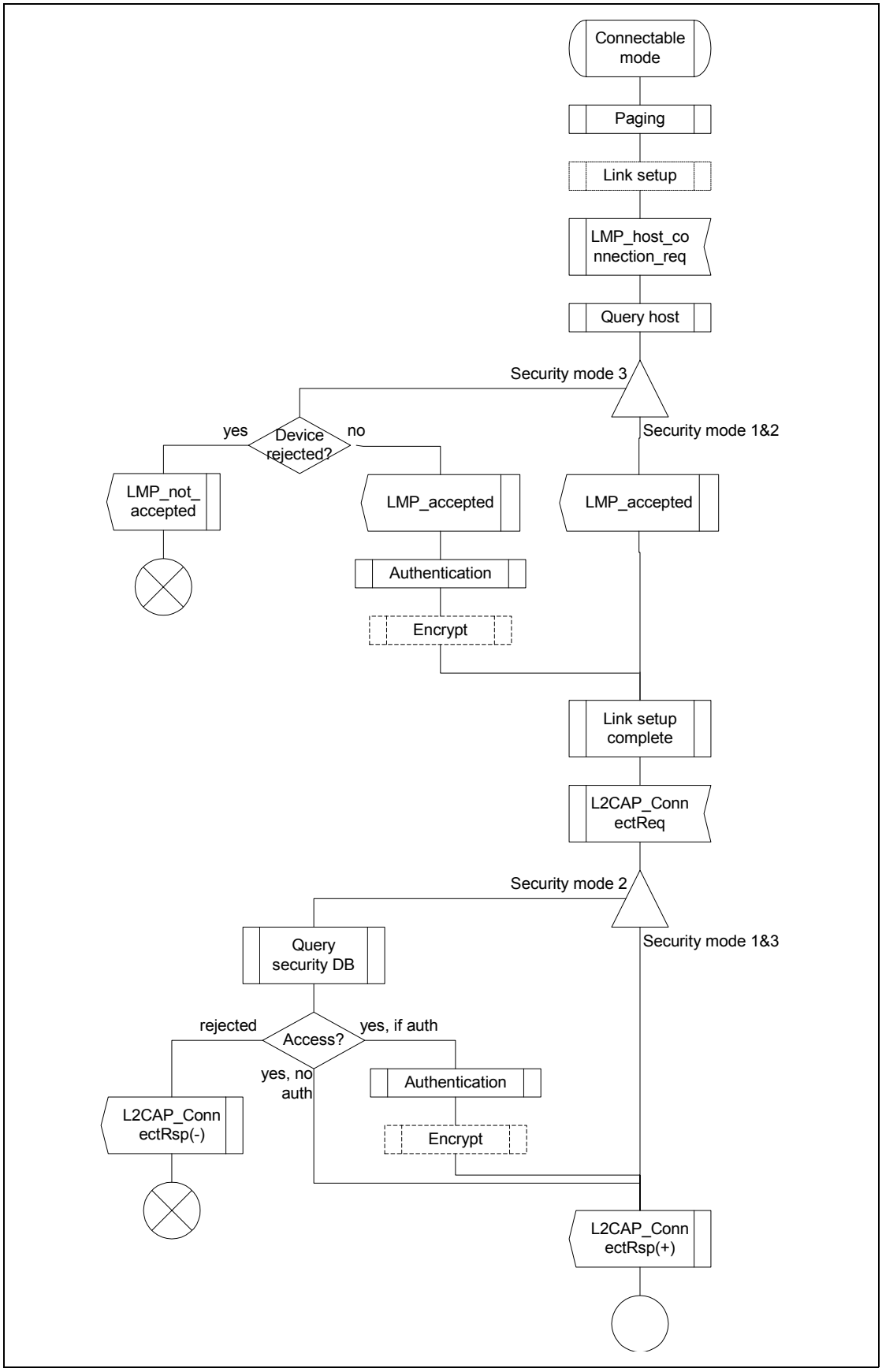


Figure 5.2: Illustration of channel establishment using different security modes.

When authentication is initiated towards a Bluetooth device, it shall act according to [2] and the current pairing mode, independent of which security mode it is in.

5.2.1 Security mode 1 (non-secure)

When a Bluetooth device is in security mode 1 it shall never initiate any security procedure (i.e., it shall never send LMP_au_rand, LMP_in_rand or LMP_encryption_mode_req).

5.2.2 Security mode 2 (service level enforced security)

When a Bluetooth device is in security mode 2 it shall not initiate any security procedure before a channel establishment request (L2CAP_ConnectReq) has been received or a channel establishment procedure has been initiated by itself. (The behavior of a device in security mode 2 is further described in [10].) Whether a security procedure is initiated or not depends on the security requirements of the requested channel or service.

A Bluetooth device in security mode 2 should classify the security requirements of its services using at least the following attributes:

- Authorization required;
- Authentication required;
- Encryption required.

Note: Security mode 1 can be considered (at least from a remote device point of view) as a special case of security mode 2 where no service has registered any security requirements.

5.2.3 Security modes 3 (link level enforced security)

When a Bluetooth device is in security mode 3 it shall initiate security procedures before it sends LMP_link_setup_complete. (The behavior of a device in security mode 3 is as described in [2].)

A Bluetooth device in security mode 3 may reject the host connection request (respond with LMP_not_accepted to the LMP_host_connection_req) based on settings in the host (e.g. only communication with pre-paired devices allowed).

6 IDLE MODE PROCEDURES

The inquiry and discovery procedures described here are applicable only to the device that initiates them (A). The requirements on the behavior of B is according to the modes specified in Section 4 on page 189 and to [2].

	Procedure	Ref.	Support
1	General inquiry	6.1	C1
2	Limited inquiry	6.2	C1
3	Name discovery	6.3	O
4	Device discovery	6.4	O
5	Bonding	6.5	O
C1: If initiation of bonding is supported, support for at least one inquiry procedure is mandatory, otherwise optional. (Note: support for LMP-pairing is mandatory [2].)			

6.1 GENERAL INQUIRY

6.1.1 Purpose

The purpose of the general inquiry procedure is to provide the initiator with the Bluetooth device address, clock, Class of Device and used page scan mode of general discoverable devices (i.e. devices that are in range with regard to the initiator and are set to scan for inquiry messages with the General Inquiry Access Code). Also devices in limited discoverable mode will be discovered using general inquiry.

The general inquiry should be used by devices that need to discover devices that are made discoverable continuously or for no specific condition.

6.1.2 Term on UI level

’Bluetooth Device Inquiry’.

6.1.3 Description

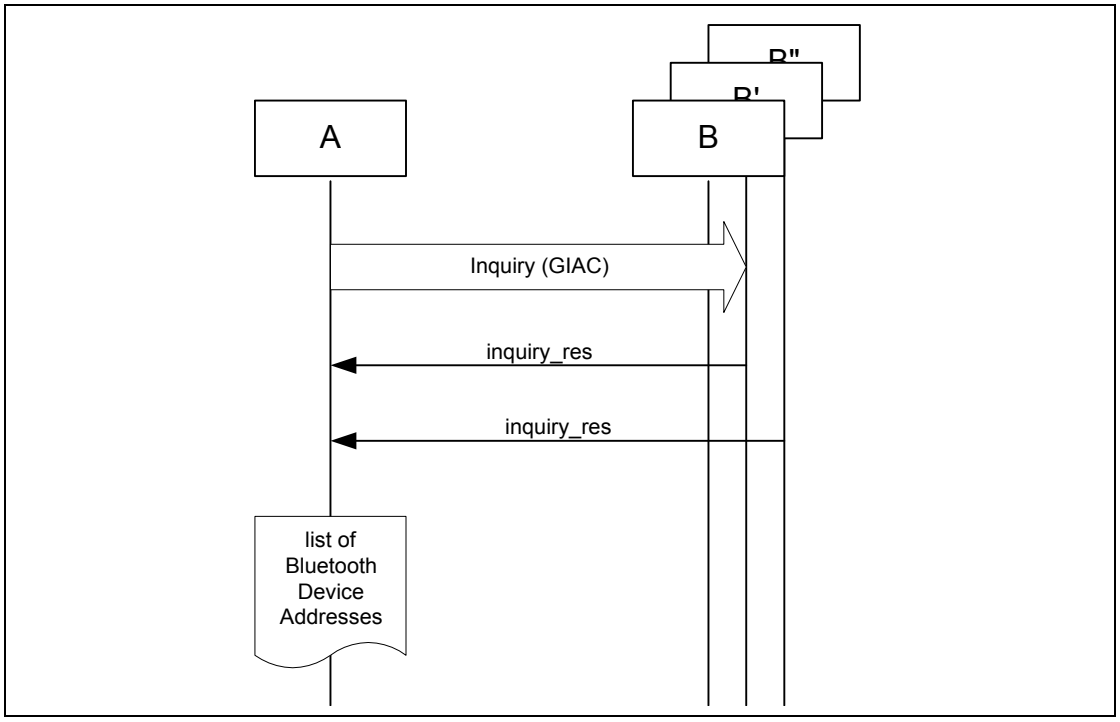


Figure 6.1: General inquiry ,where B is a device in non-discoverable mode, B´ is a device in limited discoverable mode and B" is a device in general discoverable mode. (Note that all discoverable devices are discovered using general inquiry, independent of which discoverable mode they are in.)

6.1.4 Conditions

When general inquiry is initiated by a Bluetooth device, the INQUIRY state shall last $T_{GAP}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The Bluetooth device shall perform inquiry using the GIAC.

In order to receive inquiry response, the remote devices in range have to be made discoverable (limited or general).

6.2 LIMITED INQUIRY

6.2.1 Purpose

The purpose of the limited inquiry procedure is to provide the initiator with the Bluetooth device address, clock, Class of Device and used page scan mode of limited discoverable devices. The latter devices are devices that are in range with regard to the initiator, and may be set to scan for inquiry messages with the Limited Inquiry Access Code, in addition to scanning for inquiry messages with the General Inquiry Access Code.

The limited inquiry should be used by devices that need to discover devices that are made discoverable only for a limited period of time, during temporary

conditions or for a specific event. Since it is not guaranteed that the discoverable device scans for the LIAC, the initiating device may choose any inquiry procedure (general or limited). Even if the remote device that is to be discovered is expected to be made limited discoverable (e.g. when a dedicated bonding is to be performed), the limited inquiry should be done in sequence with a general inquiry in such a way that both inquiries are completed within the time the remote device is limited discoverable, i.e. at least $T_{GAP}(103)$.

6.2.2 Term on UI level

'Bluetooth Device Inquiry'.

6.2.3 Description

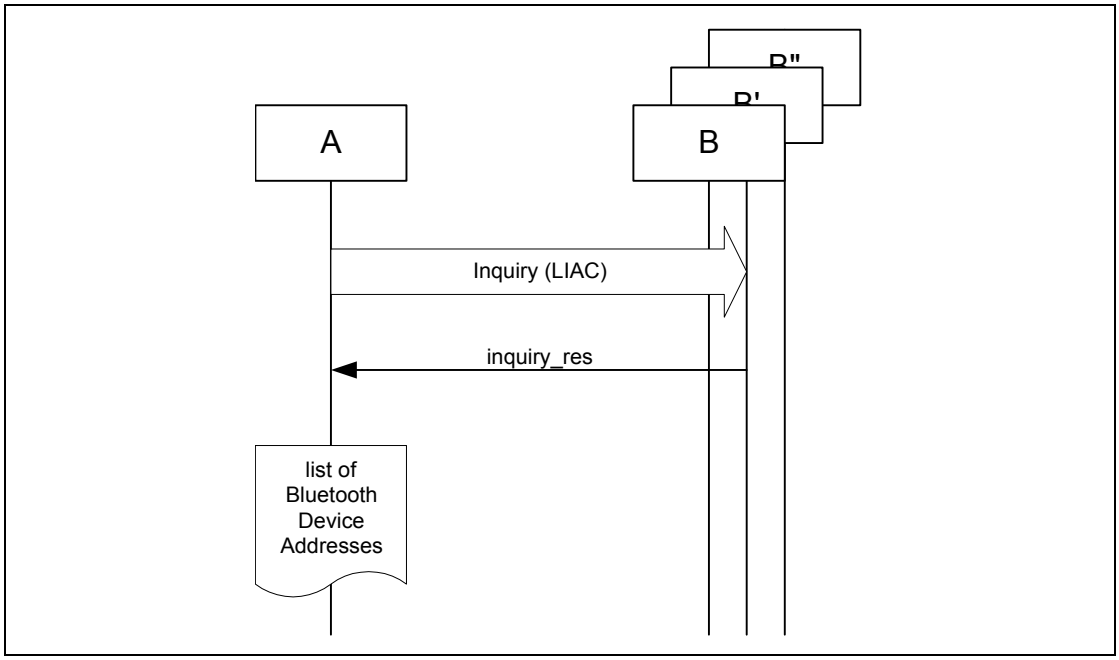


Figure 6.2: Limited inquiry where B is a device in non-discoverable mode, B' is a device in limited discoverable mode and B" is a device in general discoverable mode. (Note that only limited discoverable devices can be discovered using limited inquiry.)

6.2.4 Conditions

When limited inquiry is initiated by a Bluetooth device, the INQUIRY state shall last $T_{GAP}(100)$ or longer, unless the inquirer collects enough responses and determines to abort the INQUIRY state earlier. The Bluetooth device shall perform inquiry using the LIAC.

In order to receive inquiry response, the remote devices in range has to be made limited discoverable.

6.3 NAME DISCOVERY

6.3.1 Purpose

The purpose of name discovery is to provide the initiator with the Bluetooth Device Name of connectable devices (i.e. devices in range that will respond to paging).

6.3.2 Term on UI level

'Bluetooth Device Name Discovery'.

6.3.3 Description

6.3.3.1 Name request

Name request is the procedure for retrieving the Bluetooth Device Name from a connectable Bluetooth device. It is not necessary to perform the full link establishment procedure (see Section 7.1 on page 209) in order to just to get the name of another device.

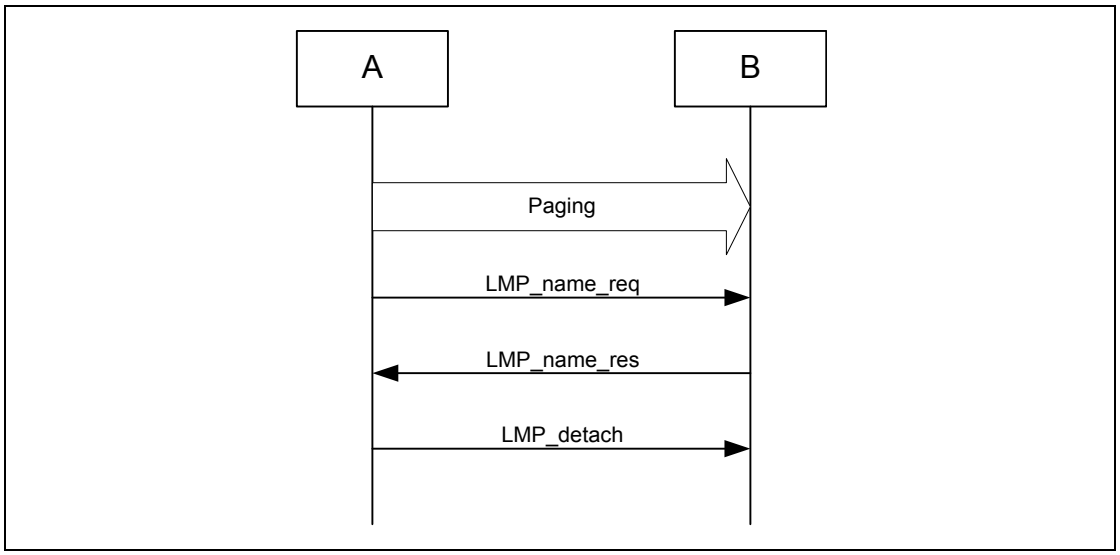


Figure 6.3: Name request procedure.

6.3.3.2 Name discovery

Name discovery is the procedure for retrieving the Bluetooth Device Name from connectable Bluetooth devices by performing name request towards known devices (i.e. Bluetooth devices for which the Bluetooth Device Addresses are available).

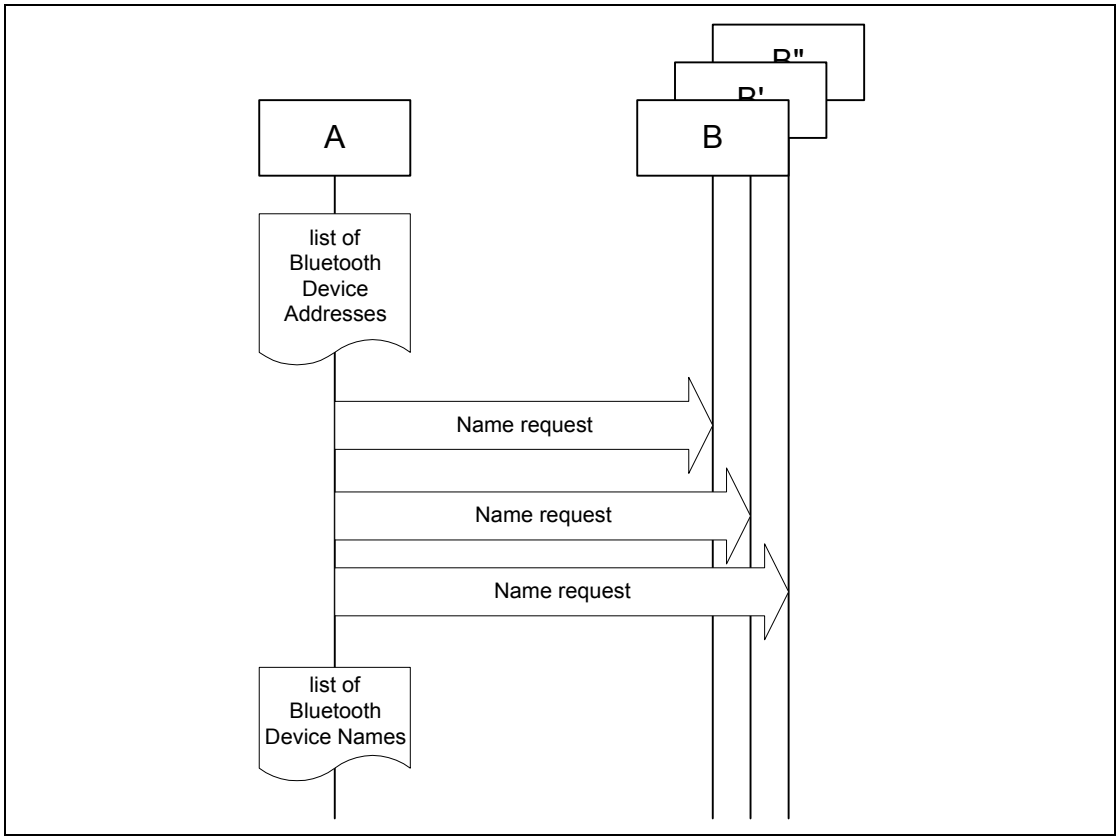


Figure 6.4: Name discovery procedure.

6.3.4 Conditions

In the name request procedure, the initiator will use the Device Access Code of the remote device as retrieved immediately beforehand – normally through an inquiry procedure.

6.4 DEVICE DISCOVERY

6.4.1 Purpose

The purpose of device discovery is to provide the initiator with the Bluetooth Device Address, clock, Class of Device, used page scan mode and Bluetooth device name of discoverable devices.

6.4.2 Term on UI level

'Bluetooth Device Discovery'.

6.4.3 Description

During the device discovery procedure, first an inquiry (either general or limited) is performed, and then name discovery is done towards some or all of the devices that responded to the inquiry.

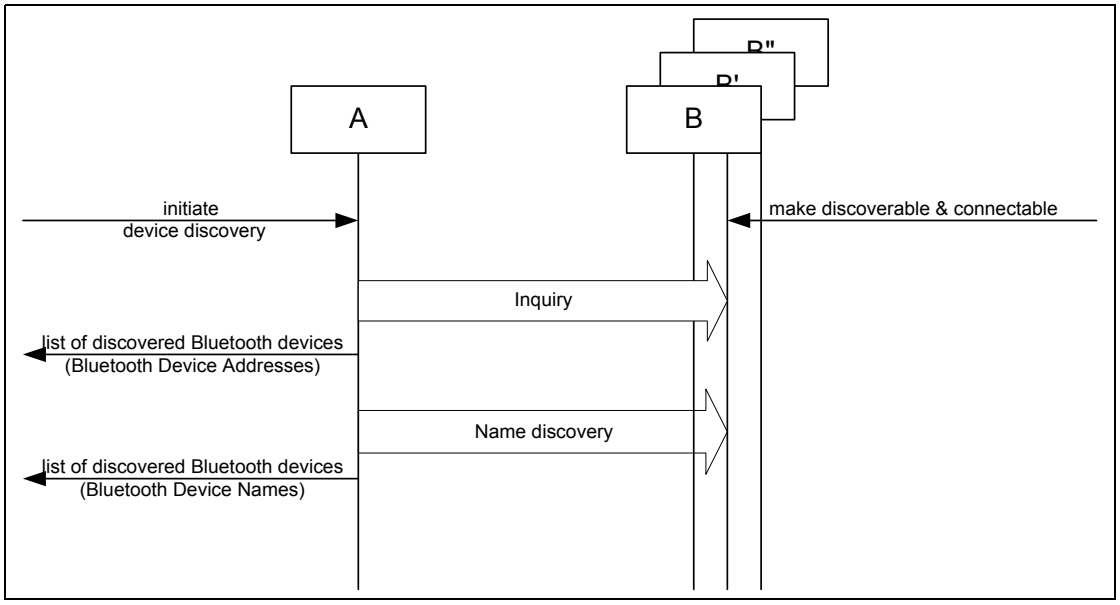


Figure 6.5: Device discovery procedure.

6.4.4 Conditions

Conditions for both inquiry (general or limited) and name discovery must be fulfilled (i.e. devices discovered during device discovery must be both discoverable and connectable).

6.5 BONDING

6.5.1 Purpose

The purpose of bonding is to create a relation between two Bluetooth devices based on a common link key (a bond). The link key is created and exchanged (pairing) during the bonding procedure and is expected to be stored by both Bluetooth devices, to be used for future authentication.

In addition to pairing, the bonding procedure can involve higher layer initialization procedures.

6.5.2 Term on UI level

’Bluetooth Bonding’

6.5.3 Description

Two aspects of the bonding procedure are described here. Dedicated bonding is what is done when the two devices are explicitly set to perform only a creation and exchange of a common link key.

General bonding is included to indicate that the framework for the dedicated bonding procedure is the same as found in the normal channel and connection establishment procedures. This means that pairing may be performed successfully if A has initiated bonding while B is in its normal connectable and security modes.

The main difference with bonding, as compared to a pairing done during link or channel establishment, is that for bonding it is the paging device (A) that must initiate the authentication.

6.5.3.1 General bonding

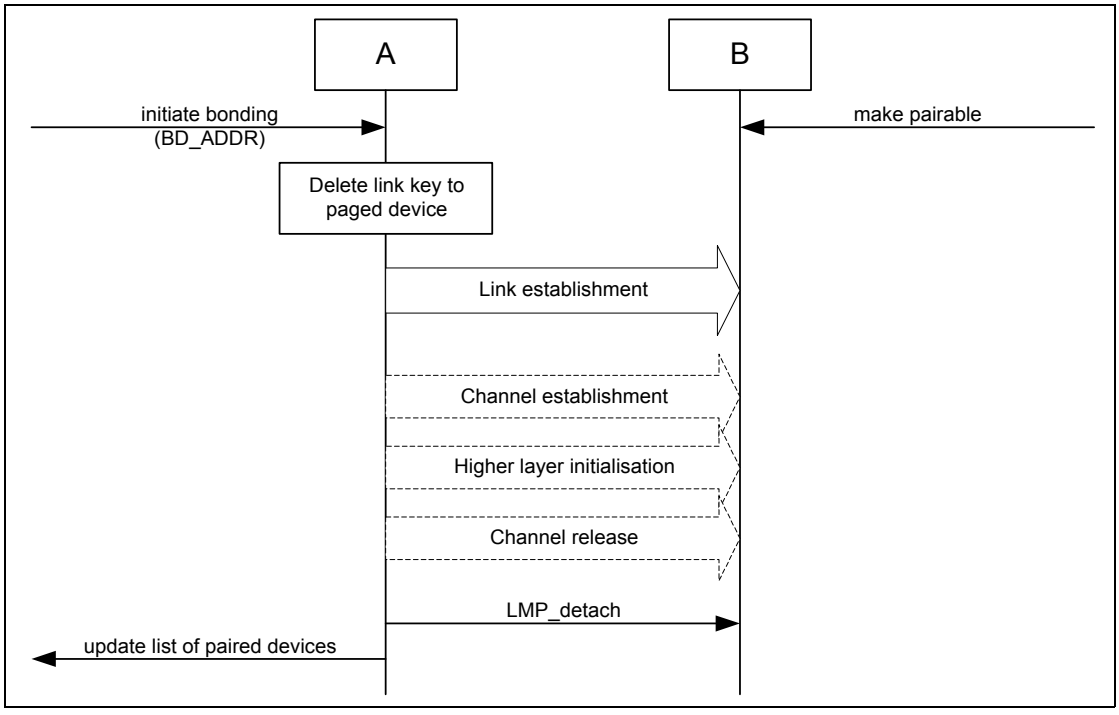


Figure 6.6: General description of bonding as being the link establishment procedure executed under specific conditions on both devices, followed by an optional higher layer initalization process.

6.5.3.2 Dedicated bonding

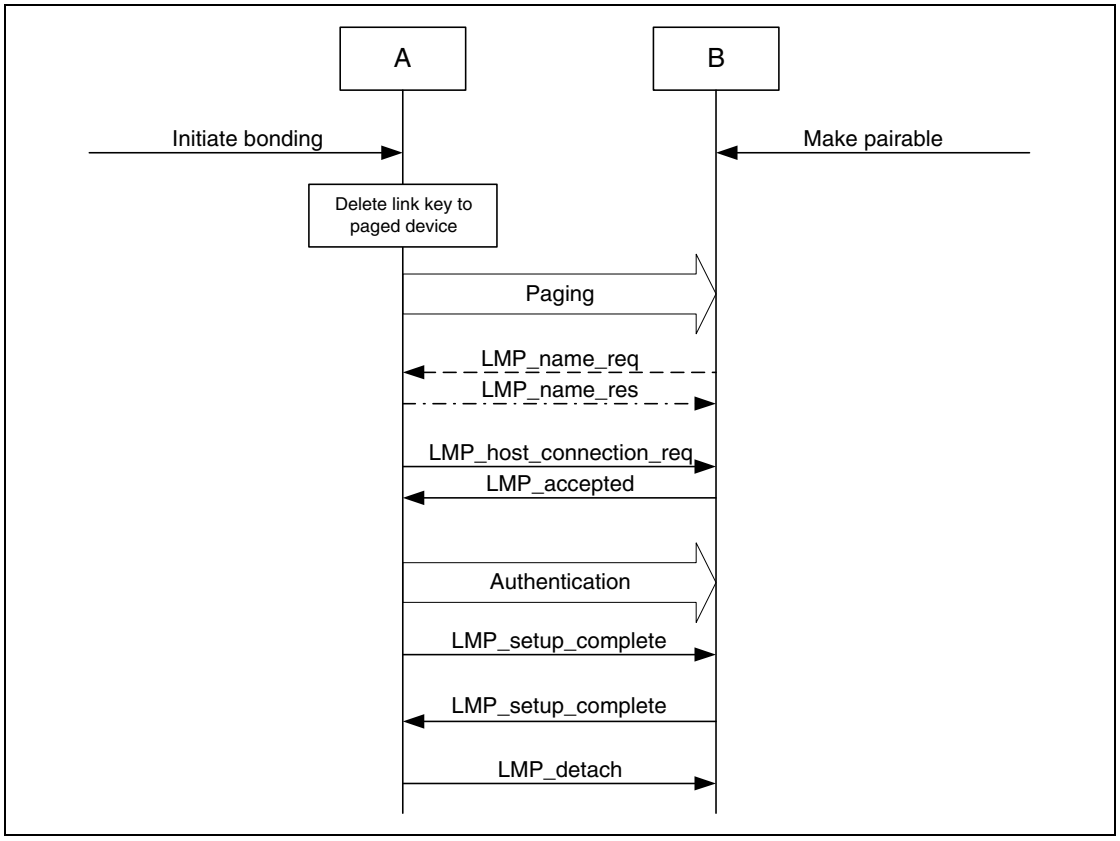


Figure 6.7: Bonding as performed when the purpose of the procedure is only to create and exchange a link key between two Bluetooth devices.

6.5.4 Conditions

Before bonding can be initiated, the initiating device (A) must know the Device Access Code of the device to pair with. This is normally done by first performing device discovery. A Bluetooth Device that can initiate bonding (A) should use limited inquiry, and a Bluetooth Device that accepts bonding (B) should support the limited discoverable mode.

Bonding is in principle the same as link establishment with the conditions:

- The paged device (B) shall be set into pairable mode. The paging device (A) is assumed to allow pairing since it has initiated the bonding procedure.
- The paging device (the initiator of the bonding procedure, A) shall initiate authentication.
- Before initiating the authentication part of the bonding procedure, the paging device should delete any link key corresponding to a previous bonding with the paged device.

7 ESTABLISHMENT PROCEDURES

	Procedure	Ref.	Support in A	Support in B
1	Link establishment	7.1	M	M
2	Channel establishment	7.2	O	M
3	Connection establishment	7.3	O	O

Table 7.1: Establishment procedures

The establishment procedures defined here do not include any discovery part. Before establishment procedures are initiated, the information provided during device discovery (in the FHS packet of the inquiry response or in the response to a name request) has to be available in the initiating device. This information is:

- The Bluetooth Device Address (BD_ADDR) from which the Device Access Code is generated;
- The system clock of the remote device;
- The page scan mode used by the remote device.

Additional information provided during device discovery that is useful for making the decision to initiate an establishment procedure is:

- The Class of device;
- The Device name.

7.1 LINK ESTABLISHMENT

7.1.1 Purpose

The purpose of the link establishment procedure is to establish a physical link (of ACL type) between two Bluetooth devices using procedures from [1] and [2].

7.1.2 Term on UI level

'Bluetooth link establishment'

7.1.3 Description

In this sub-section, the paging device (A) is in security mode 3. The paging device cannot during link establishment distinguish if the paged device (B) is in security mode 1 or 2.

7.1.3.1 B in security mode 1 or 2

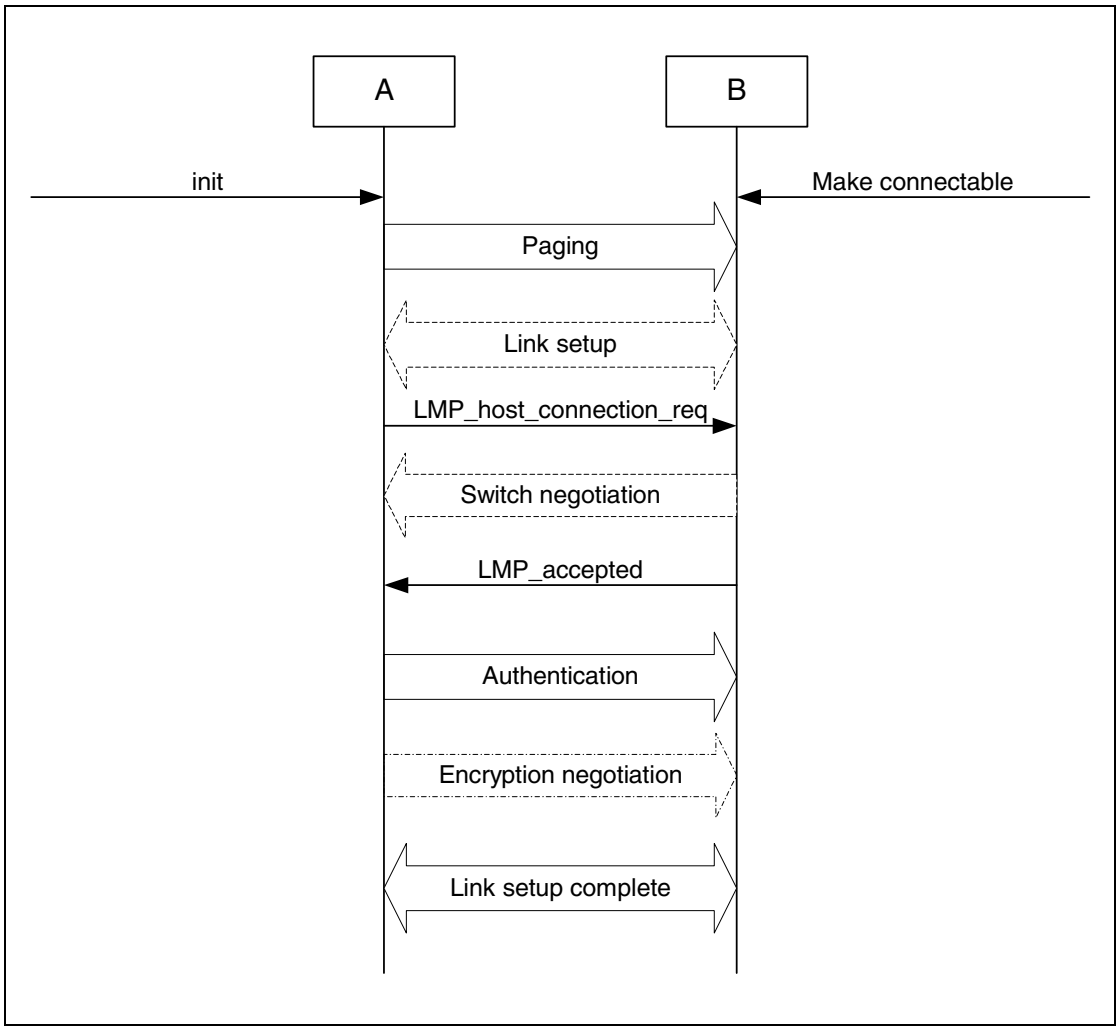


Figure 7.1: Link establishment procedure when the paging device (A) is in security mode 3 and the paged device (B) is in security mode 1 or 2.

7.1.3.2 B in security mode 3

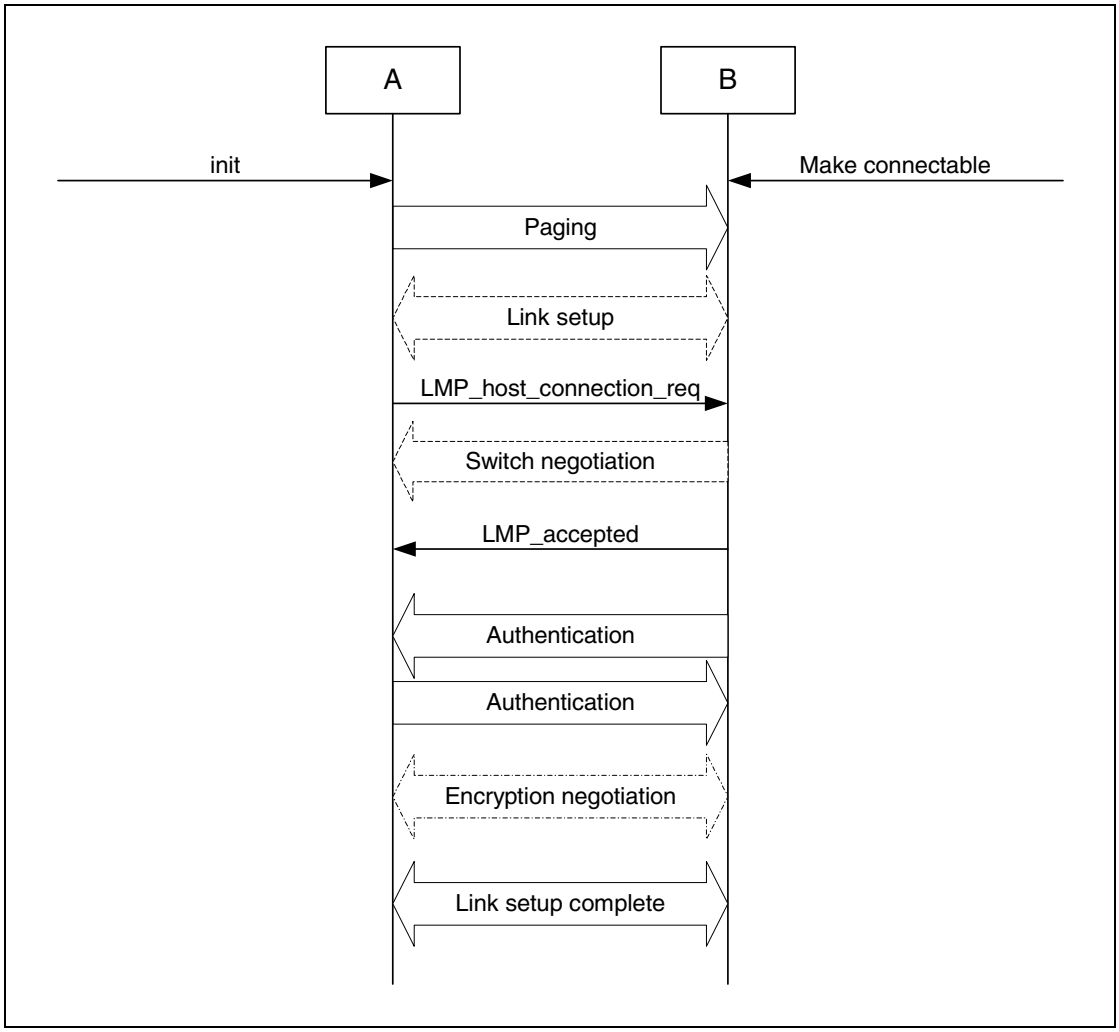


Figure 7.2: Link establishment procedure when both the paging device (A) and the paged device (B) are in security mode 3.

7.1.4 Conditions

The paging procedure shall be according to [1] and the paging device should use the Device access code and page mode received through a previous inquiry. When paging is completed, a physical link between the two Bluetooth devices is established.

If role switching is needed (normally it is the paged device that has an interest in changing the master/slave roles) it should be done as early as possible after the physical link is established. If the paging device does not accept the switch, the paged device has to consider whether to keep the physical link or not.

Both devices may perform link setup (using LMP procedures that require no interaction with the host on the remote side). Optional LMP features can be used after having confirmed (using LMP_feature_req) that the other device supports the feature.

When the paging device needs to go beyond the link setup phase, it issues a request to be connected to the host of the remote device. If the paged device is in security mode 3, this is the trigger for initiating authentication.

The paging device shall send LMP_host_connection_req during link establishment (i.e. before channel establishment) and may initiate authentication only after having sent LMP_host_connection_request.

After an authentication has been performed, any of the devices can initiate encryption.

Further link configuration may take place after the LMP_host_connection_req. When both devices are satisfied, they send LMP_setup_complete.

Link establishment is completed when both devices have sent LMP_setup_complete.

7.2 CHANNEL ESTABLISHMENT

7.2.1 Purpose

The purpose of the channel establishment procedure is to establish a Bluetooth channel (L2CAP channel) between two Bluetooth devices using [3].

7.2.2 Term on UI level

'Bluetooth channel establishment'.

7.2.3 Description

In this sub-section, the initiator (A) is in security mode 3. During channel establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3.

7.2.3.1 B in security mode 2

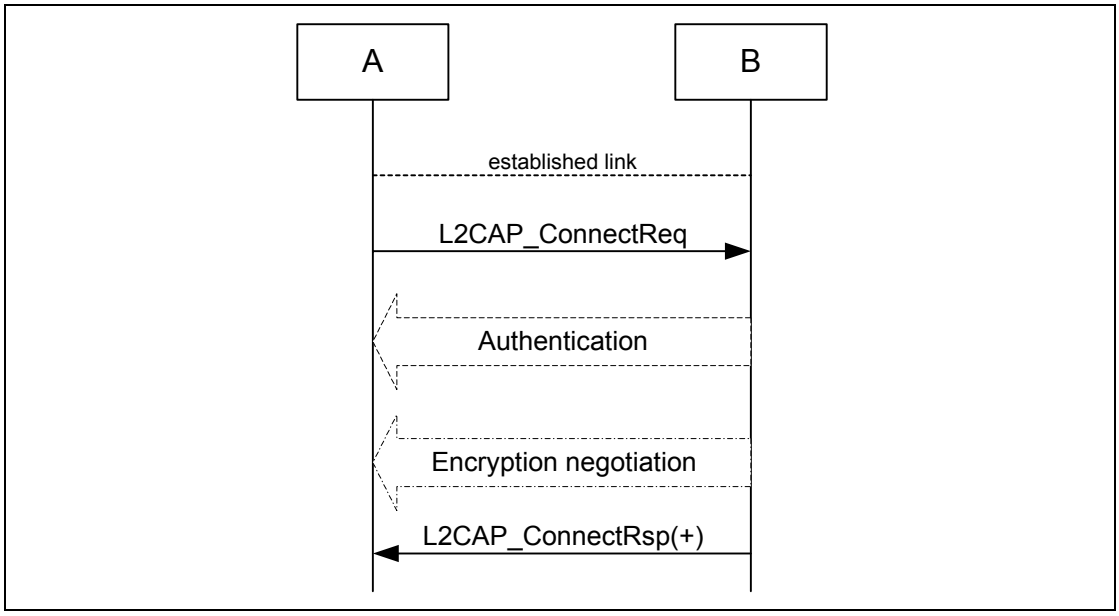


Figure 7.3: Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2.

7.2.3.2 B in security mode 1 or 3

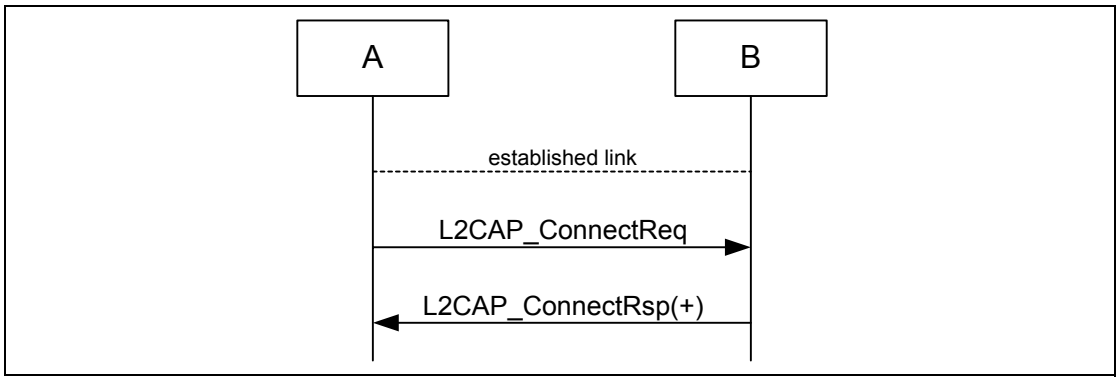


Figure 7.4: Channel establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3.

7.2.4 Conditions

Channel establishment starts after link establishment is completed when the initiator sends a channel establishment request (L2CAP_ConnectReq).

Depending on security mode, security procedures may take place after the channel establishment has been initiated.

Channel establishment is completed when the acceptor responds to the channel establishment request (with a positive L2CAP_ConnectRsp).

7.3 CONNECTION ESTABLISHMENT

7.3.1 Purpose

The purpose of the connection establishment procedure is to establish a connection between applications on two Bluetooth devices.

7.3.2 Term on UI level

'Bluetooth connection establishment'

7.3.3 Description

In this sub-section, the initiator (A) is in security mode 3. During connection establishment, the initiator cannot distinguish if the acceptor (B) is in security mode 1 or 3.

7.3.3.1 B in security mode 2

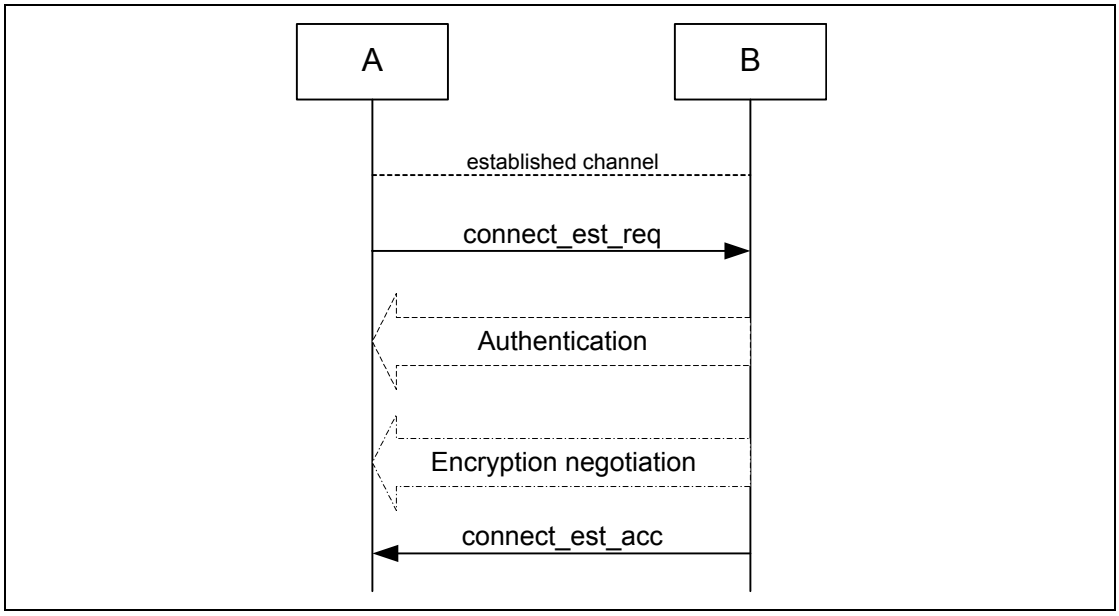


Figure 7.5: Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 2.

7.3.3.2 B in security mode 1 or 3

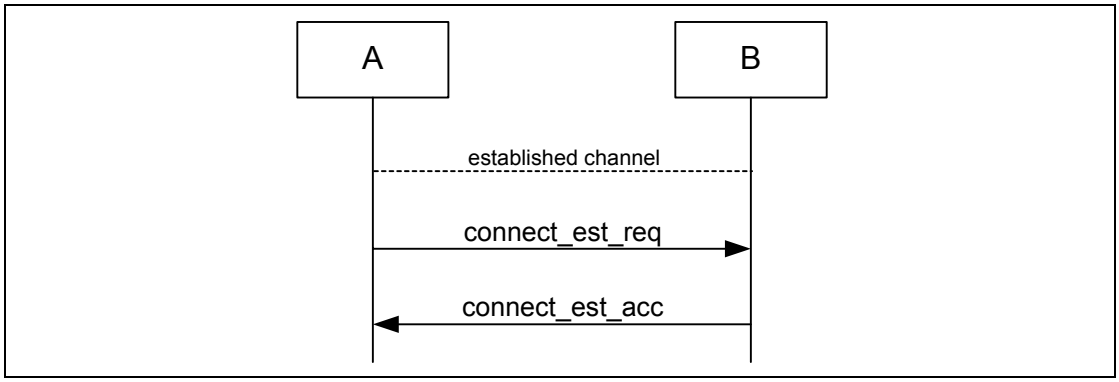


Figure 7.6: Connection establishment procedure when the initiator (A) is in security mode 3 and the acceptor (B) is in security mode 1 or 3.

7.3.4 Conditions

Connection establishment starts after channel establishment is completed, when the initiator sends a connection establishment request (’connect_est_req’ is application protocol-dependent). This request may be a TCS SETUP message [5] in the case of a Bluetooth telephony application Cordless Telephony Profile, or initialization of RFCOMM and establishment of DLC [4] in the case of a serial port-based application Serial Port Profile (although neither TCS or RFCOMM use the term ’connection’ for this).

Connection establishment is completed when the acceptor accepts the connection establishment request (’connect_est_acc’ is application protocol dependent).

7.4 ESTABLISHMENT OF ADDITIONAL CONNECTION

When a Bluetooth device has established one connection with another Bluetooth device, it may be available for establishment of:

- A second connection on the same channel, and/or
- A second channel on the same logical link, and/or
- A second physical link.

If the new establishment procedure is to be towards the same device, the security part of the establishment depends on the security modes used. If the new establishment procedure is to be towards a new remote device, the device should behave according to active modes independent of the fact that it already has another physical link established (unless allowed co-incident radio and baseband events have to be handled).

8 DEFINITIONS

In the following, terms written with capital letters refer to states.

8.1 GENERAL DEFINITIONS

Mode . A set of directives that defines how a device will respond to certain events.

Idle . As seen from a remote device, a Bluetooth device is idle, or is in idle mode, when there is no link established between them.

Bond . A relation between two Bluetooth devices defined by creating, exchanging and storing a common link key. The bond is created through the bonding or LMP-pairing procedures.

8.2 CONNECTION-RELATED DEFINITIONS

Physical channel . A synchronized Bluetooth baseband-compliant RF hopping sequence.

Piconet . A set of Bluetooth devices sharing the same physical channel defined by the master parameters (clock and BD_ADDR).

Physical link . A Baseband-level connection[1] between two devices established using paging. A physical link comprises a sequence of transmission slots on a physical channel alternating between master and slave transmission slots.

ACL link . An asynchronous (packet-switched) connection[1] between two devices created on LMP level. Traffic on an ACL link uses ACL packets to be transmitted.

SCO link . A synchronous (circuit-switched) connection[1] for reserved bandwidth communications; e.g. voice between two devices, created on the LMP level by reserving slots periodically on a physical channel. Traffic on an SCO link uses SCO packets to be transmitted. SCO links can be established only after an ACL link has first been established.

Link . Shorthand for an ACL link.

PAGE . A baseband state where a device transmits page trains, and processes any eventual responses to the page trains.

PAGE_SCAN . A baseband state where a device listens for page trains.

1. The term ’connection’ used here is not identical to the definition below. It is used in the absence of a more concise term.

Page . The transmission by a device of page trains containing the Device Access Code of the device to which the physical link is requested.

Page scan . The listening by a device for page trains containing its own Device Access Code.

Channel . A logical connection on L2CAP level between two devices serving a single application or higher layer protocol.

Connection . A connection between two peer applications or higher layer protocols mapped onto a channel.

Connecting . A phase in the communication between devices when a connection between them is being established. (Connecting phase follows after the link establishment phase is completed.)

Connect (to service) . The establishment of a connection to a service. If not already done, this includes establishment of a physical link, link and channel as well.

8.3 DEVICE-RELATED DEFINITIONS

Discoverable device . A Bluetooth device in range that will respond to an inquiry (normally in addition to responding to page).

Silent device . A Bluetooth device appears as silent to a remote device if it does not respond to inquiries made by the remote device. A device may be silent due to being non-discoverable or due to baseband congestion while being discoverable.

Connectable device . A Bluetooth device in range that will respond to a page.

Trusted device . A paired device that is explicitly marked as trusted.

Paired device . A Bluetooth device with which a link key has been exchanged (either before connection establishment was requested or during connecting phase).

Pre-paired device . A Bluetooth device with which a link key was exchanged, and the link key is stored, before link establishment.

Un-paired device . A Bluetooth device for which there was no exchanged link key available before connection establishment was request.

Known device . A Bluetooth device for which at least the BD_ADDR is stored.

Un-known device . A Bluetooth device for which no information (BD_ADDR, link key or other) is stored.

Authenticated device . A Bluetooth device whose identity has been verified during the lifetime of the current link, based on the authentication procedure.

8.4 PROCEDURE-RELATED DEFINITIONS

Paging . A procedure for establishing a physical link of ACL type on baseband level, consisting of a page action of the initiator and a page scan action of the responding device.

Link establishment . A procedure for establishing a link on LMP level. A link is established when both devices have agreed that LMP setup is completed.

Channel establishment . A procedure for establishing a channel on L2CAP level.

Connection establishment . A procedure for creating a connection mapped onto a channel.

Creation of a trusted relationship . A procedure where the remote device is marked as a trusted device. This includes storing a common link key for future authentication and pairing (if the link key is not available).

Creation of a secure connection. A procedure of establishing a connection, including authentication and encryption.

Device discovery . A procedure for retrieving the Bluetooth device address, clock, class-of-device field and used page scan mode from discoverable devices.

Name discovery . A procedure for retrieving the user-friendly name (the Bluetooth device name) of a connectable device.

Service discovery . Procedures for querying and browsing for services offered by or through another Bluetooth device.

8.5 SECURITY-RELATED DEFINITIONS

Authentication . A generic procedure based on LMP-authentication if a link key exists or on LMP-pairing if no link key exists.

LMP-authentication . An LMP level procedure for verifying the identity of a remote device. The procedure is based on a challenge-response mechanism using a random number, a secret key and the BD_ADDR of the non-initiating device. The secret key used can be a previously exchanged link key.

Authorization . A procedure where a user of a Bluetooth device grants a specific (remote) Bluetooth device access to a specific service. Authorization implies that the identity of the remote device can be verified through authentication.

Authorize . The act of granting a specific Bluetooth device access to a specific service. It may be based upon user confirmation, or given the existence of a trusted relationship.

LMP-pairing . A procedure that authenticates two devices, based on a PIN, and subsequently creates a common link key that can be used as a basis for a trusted relationship or a (single) secure connection. The procedure consists of the steps: creation of an initialization key (based on a random number and a PIN), creation and exchange of a common link key and LMP-authentication based on the common link key.

Bonding . A dedicated procedure for performing the first authentication, where a common link key is created and stored for future use.

Trusting . The marking of a paired device as trusted. Trust marking can be done by the user, or automatically by the device (e.g. when in pairable mode) after a successful pairing.

9 APPENDIX A (NORMATIVE): TIMERS AND CONSTANTS

The following timers are required by this profile.

Timer name	Recommended value	Description	Comment
$T_{GAP}(100)$	10.24 s	Normal time span that a Bluetooth device performs inquiry.	Used during inquiry and device discovery.
$T_{GAP}(101)$	10.625 ms	Minimum time in INQUIRY_SCAN.	A discoverable Bluetooth device enters INQUIRY_SCAN for at least $T_{GAP}(101)$ every $T_{GAP}(102)$.
$T_{GAP}(102)$	2.56 s	Maximum time between repeated INQUIRY_SCAN enterings.	Maximum value of the inquiry scan interval, $T_{inquiry\ scan}$.
$T_{GAP}(103)$	30.72 s	A Bluetooth device shall not be in a discoverable mode less than $T_{GAP}(103)$.	Minimum time to be discoverable.
$T_{GAP}(104)$	1 min.	A Bluetooth device should not be in limited discover-able mode more than $T_{GAP}(104)$.	Recommended upper limit.
$T_{GAP}(105)$	100ms	Maximum time between INQUIRY_SCAN enterings	Recommended value
$T_{GAP}(106)$	100ms	Maximum time between PAGE_SCAN enterings	Recommended value
$T_{GAP}(107)$	1.28s	Maximum time between PAGE_SCAN enterings (R1 page scan)	Recommended value
$T_{GAP}(108)$	2.56s	Maximum time between PAGE_SCAN enterings (R2 page scan)	Recommended value

Table 9.1: Defined GAP timers

10 APPENDIX B (INFORMATIVE): INFORMATION FLOWS OF RELATED PROCEDURES

10.1 LMP-AUTHENTICATION

The specification of authentication on link level is found in [2].

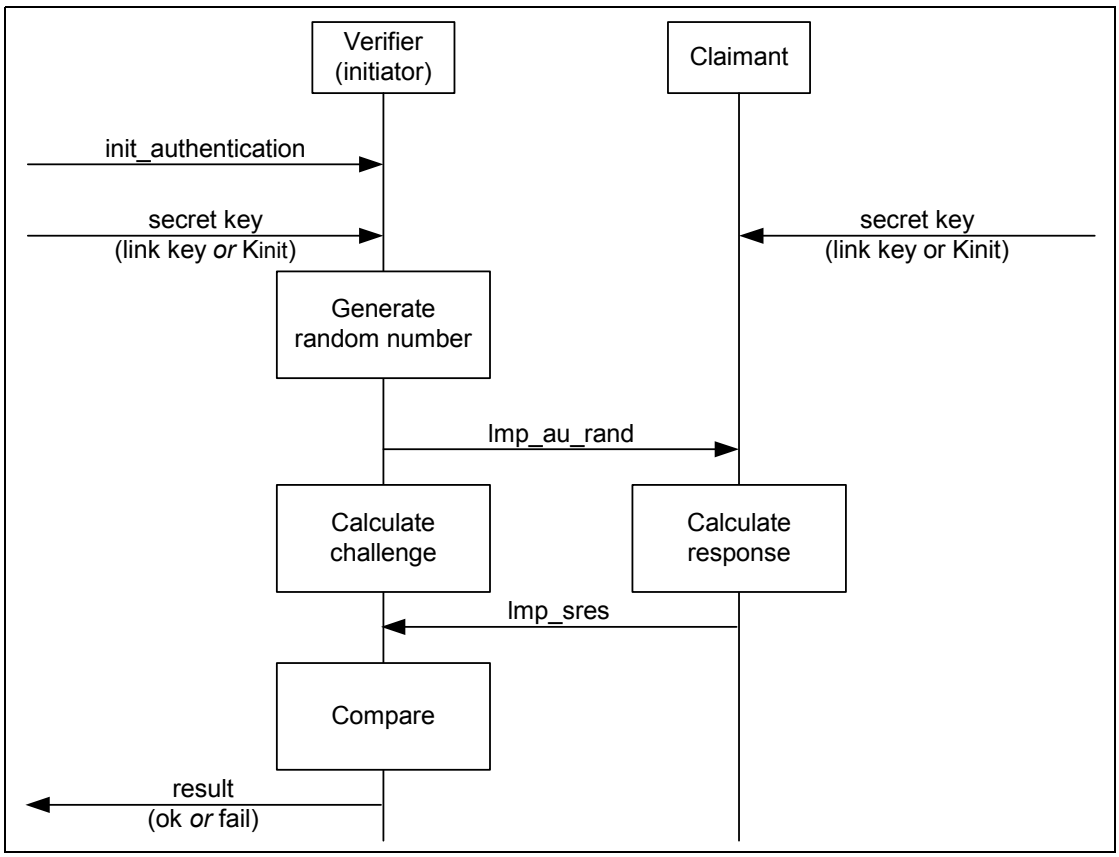


Figure 10.1: LMP-authentication as defined by [2].

The secret key used here is an already exchanged link key.

10.2 LMP-PAIRING

The specification of pairing on link level is found in [2].

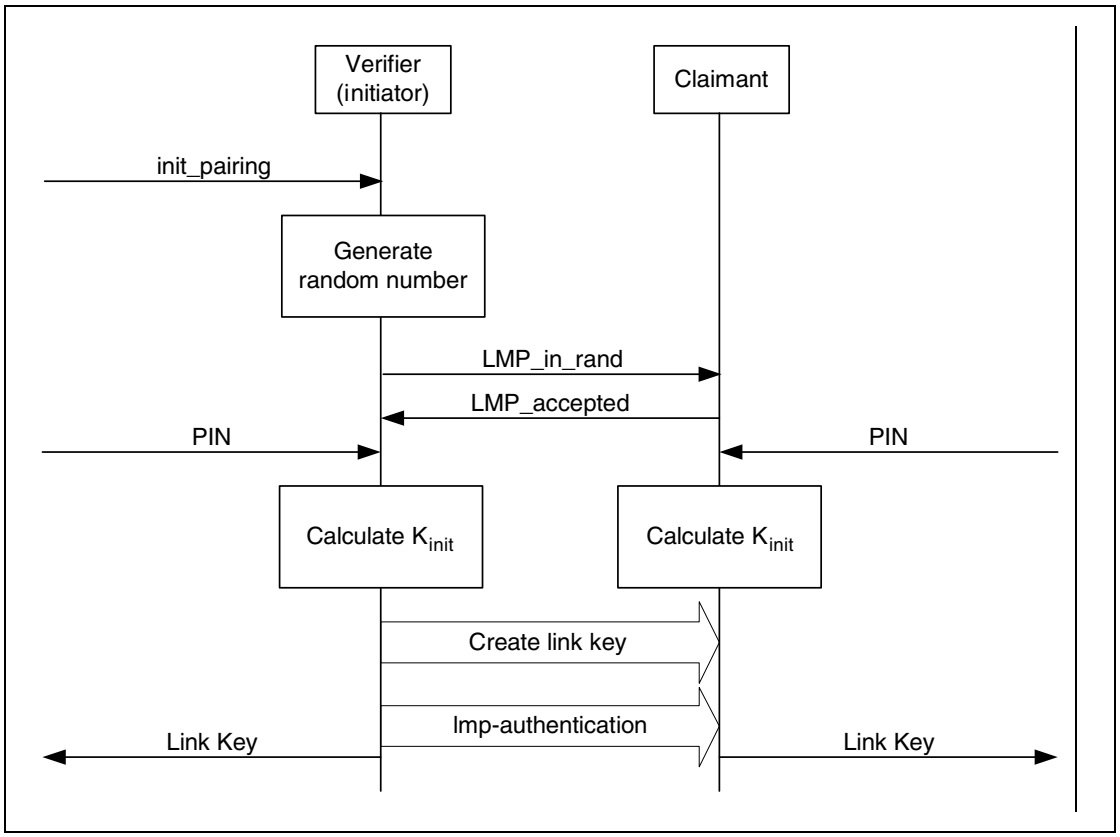


Figure 10.2: LMP-pairing as defined in [2].

The PIN used here is PN_{BB}.

The create link key procedure is described in [Vol. 3, Part C] Section 4.2.2.4 on page 253 and [Vol. 3, Part H] Section 3.2 on page 779. In case the link key is based on a combination key, a mutual authentication takes place and shall be performed irrespective of current security mode.

10.3 SERVICE DISCOVERY

The Service Discovery Protocol [6] specifies what PDUs are used over-the-air to inquire about services and service attributes. The procedures for discovery of supported services and capabilities using the Service Discovery Protocol are described in the Service Discovery Application Profile. This is just an example.

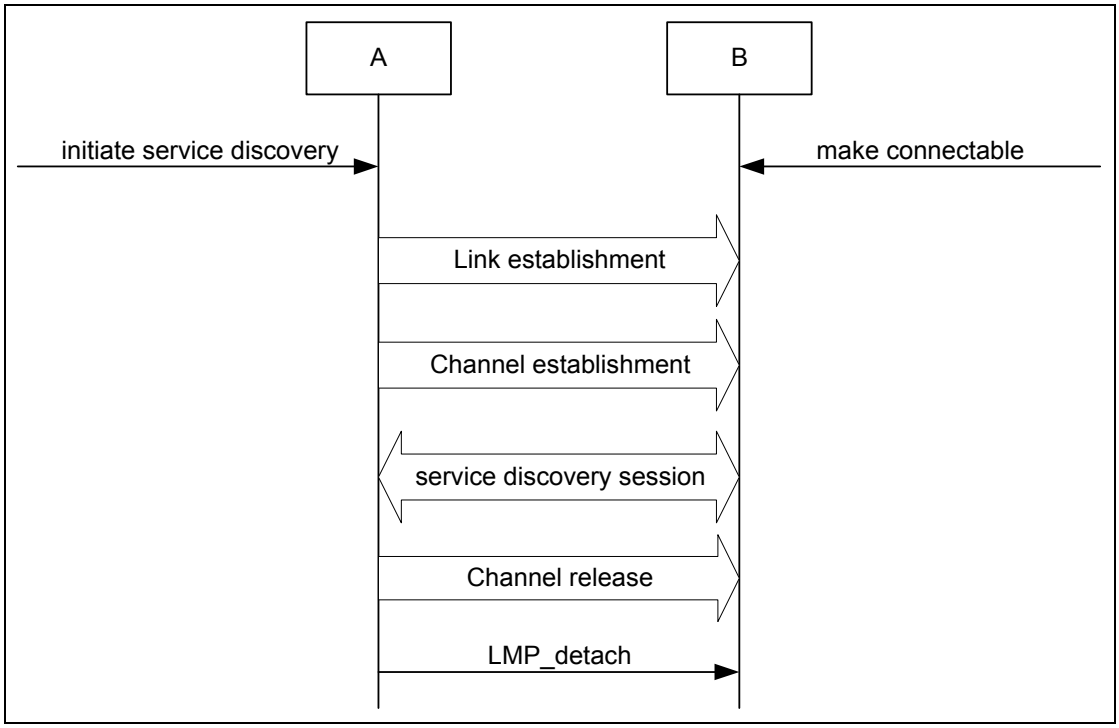


Figure 10.3: Service discovery procedure.

11 REFERENCES

[1] Bluetooth Baseband Specification

[2] Bluetooth Link Manager Protocol

[3] Bluetooth Logical Link Control and Adaptation Protocol

[4] Bluetooth RFCOMM

[5] Bluetooth Telephony Control Specification

[6] Bluetooth Service Discovery Protocol

[7] Bluetooth Service Discovery Application Profile

[8] Bluetooth Cordless Telephony Profile

[9] Bluetooth Serial Port Profile

[10] Bluetooth Security Architecture (white paper)

[11] Bluetooth Assigned Numbers
https://www.bluetooth.org/foundry/assignnumb/document/assigned_numbers

Core System Package [Host volume]
Part D

TEST SUPPORT

CONTENTS

1 TEST METHODOLOGY

This section describes the test mode for hardware and low-level functionality tests of Bluetooth devices. The test mode includes transmitter tests (packets with constant bit patterns) and loop back tests.

The test mode supports testing of the Bluetooth transmitter and receiver. It is intended mainly for certification/compliance testing of the radio and baseband layer, and may also be used for regulatory approval or in-production and after-sales testing.

1.1 TEST SCENARIOS

A device in test mode shall not support normal operation. For security reasons the test mode is designed such that it offers no benefit to the user. Therefore, no data output or acceptance on a HW or SW interface shall be allowed.

1.1.1 Test setup

The setup consists of a device under test (DUT) and a tester. Optionally, additional measurement equipment may be used.

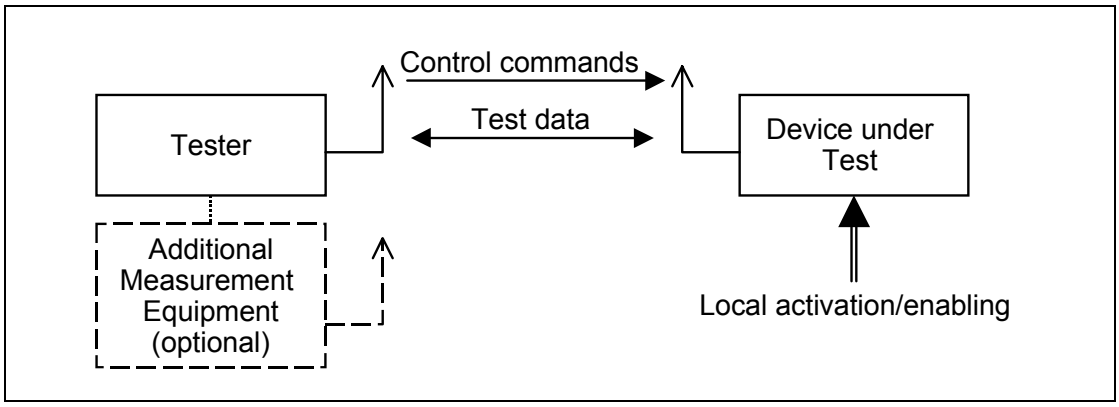


Figure 1.1: Setup for Test Mode

Tester and DUT form a piconet where the tester acts as master and has full control over the test procedure. The DUT acts as slave.

The control is done via the air interface using LMP commands (see [Vol. 3, Part C] Section 4.7.3 on page 291). Hardware interfaces to the DUT may exist, but are not subject to standardization.

The test mode is a special state of the Bluetooth model. For security and type approval reasons, a Bluetooth device in test mode shall not support normal operation. When the DUT leaves the test mode it enters the standby state. After power-off the Bluetooth device shall return to standby state.

1.1.2 Transmitter Test

The Bluetooth device transmits a constant bit pattern. This pattern is transmitted periodically with packets aligned to the slave TX timing of the piconet formed by tester and DUT. The same test packet is repeated for each transmission.

The transmitter test is started when the master sends the first POLL packet. In non-hopping mode agreed frequency is used for this POLL packet.

The tester (master) transmits control or POLL packets in the master-to-slave transmission slots. The DUT (slave) shall transmit packets in the following slave-to-master transmission slot. The tester's polling interval is fixed and defined by the LMP_test_control PDU. The device under test may transmit its burst according to the normal timing even if no packet from the tester was received. In this case, the ARQN bit is shall be set to NAK.

The burst length may exceed the length of a one slot packet. In this case the tester may take the next free master TX slot for polling. The timing is illustrated in Figure 1.2.

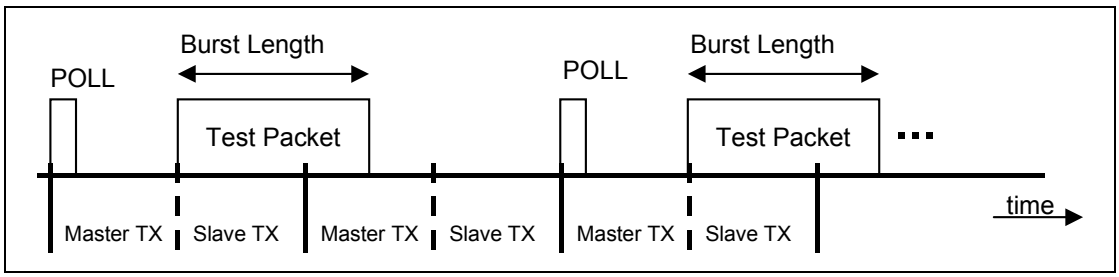


Figure 1.2: Timing for Transmitter Test

1.1.2.1 Packet Format

The test packet is a normal Bluetooth packet, see Figure 1.3. For the payload itself see below.

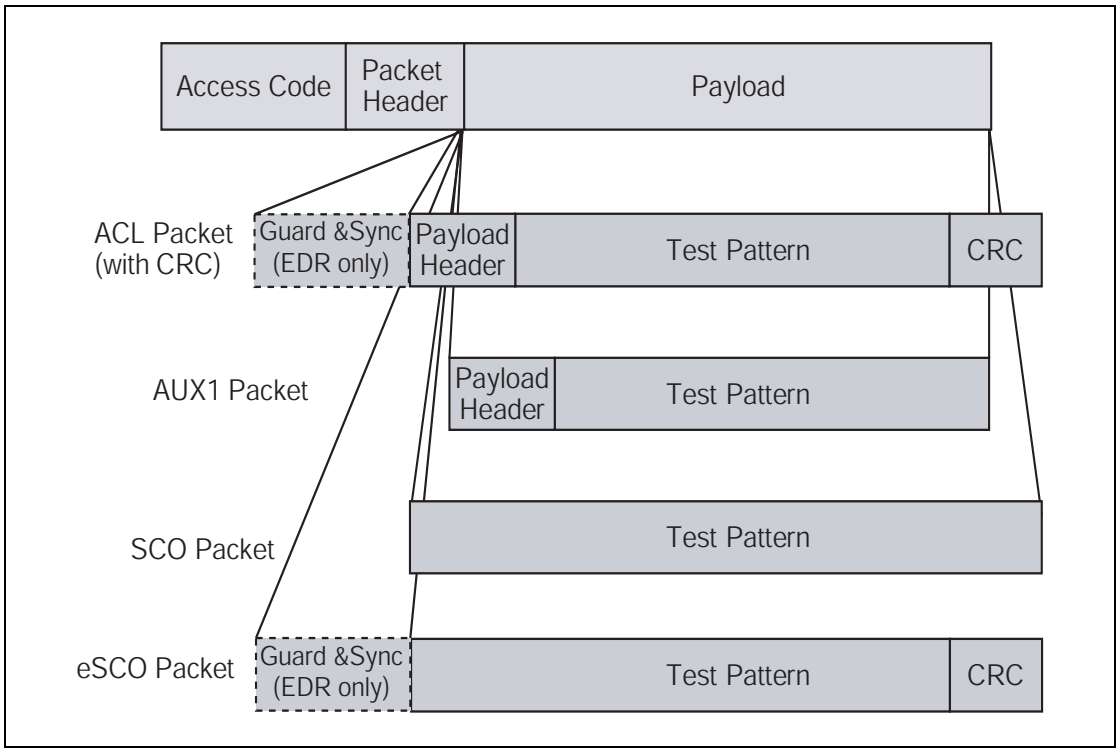


Figure 1.3: General Format of TX Packet

During configuration the tester defines:

- the packet type to be used
- payload length

For the payload length, the restrictions from the baseband specification shall apply (see "Baseband Specification" on page 55[vol. 3]). In case of ACL, SCO and eSCO packets the payload structure defined in the baseband specification is preserved as well, see Figure 1.3 on page 233.

For the transmitter test mode, only packets without FEC should be used; i.e. HV3, EV3, EV5, DH1, DH3, DH5, 2-EV3, 2-EV5, 3-EV3, 3-EV5, 2-DH1, 2-DH3, 2-DH5, 3-DH1, 3-DH3, 3-DH5 and AUX1 packets.

In transmitter test mode, the packets exchanged between the tester and the DUT shall not be scrambled with the whitening sequence. Whitening shall be turned off when the DUT has accepted to enter the transmitter test mode, and shall be turned on when the DUT has accepted to exit the transmitter test mode, see Figure 1.4 on page 234. Implementations shall insure that retransmissions of the LMP_accepted messages use the same whitening status as used in the original LMP_accepted.

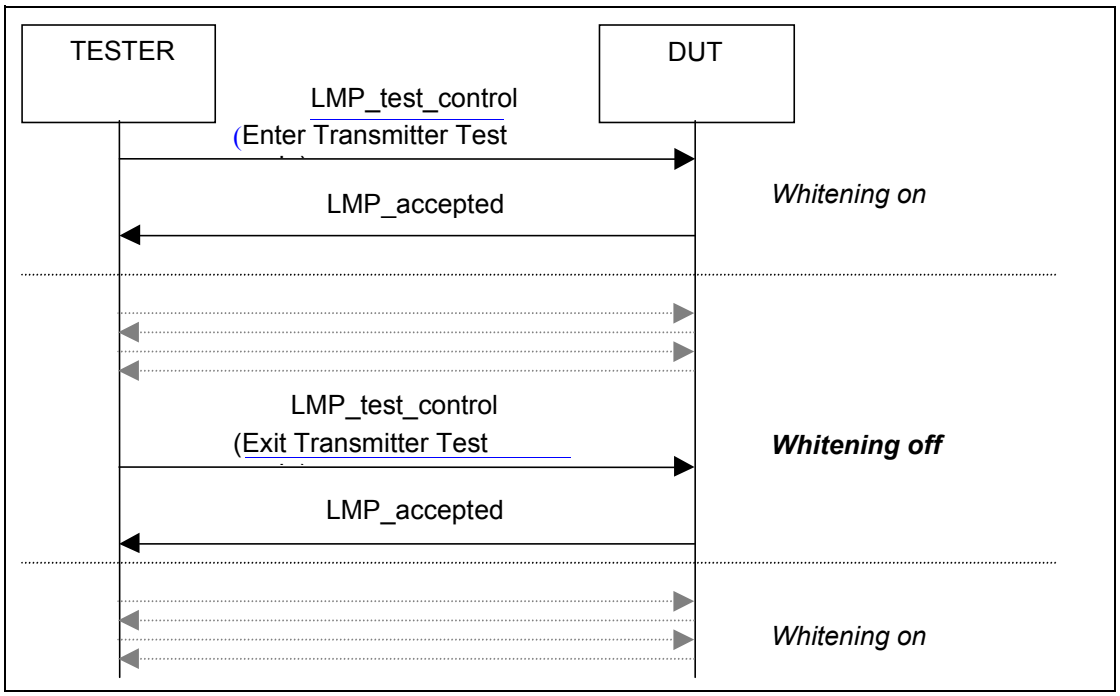


Figure 1.4: Use of whitening in Transmitter mode

1.1.2.2 Pseudorandom Sequence

The same pseudorandom sequence of bits shall be used for each transmission (i.e. the packet is repeated). A PRBS-9 Sequence is used, see [2] and [3].

The properties of this sequence are as follows (see [3]). The sequence may be generated in a nine-stage shift register whose 5th and 9th stage outputs are added in a modulo-two addition stage (see Figure 1.5), and the result is fed back to the input of the first stage. The sequence begins with the first ONE of 9 consecutive ONEs; i.e. the shift register is initialized with nine ones.

- Number of shift register stages: 9
- Length of pseudo-random sequence: 2^9-1 = 511 bits
- Longest sequence of zeros: 8 (non-inverted signal)

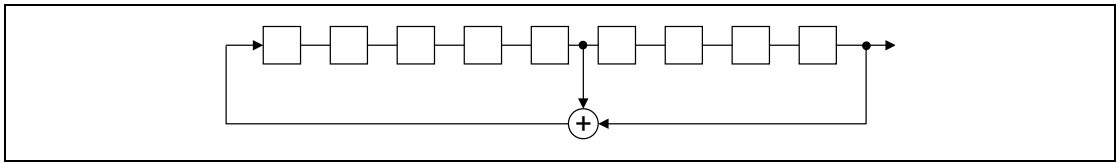


Figure 1.5: Linear Feedback Shift Register for Generation of the PRBS sequence

1.1.2.3 Control of Transmit Parameters

The following parameters can be set to configure the transmitter test:

1. Bit pattern:
   - Constant zero
   - Constant one
   - Alternating 1010...[1]
   - Alternating 1111 0000 1111 0000...[4]
   - Pseudorandom bit pattern
   - Transmission off
2. Frequency selection:
   - Single frequency
   - Normal hopping
3. TX frequency
   - k $\Rightarrow$ f := (2402 + k) MHz
4. Default poll period in TDD frames (n * 1.25 ms)
5. Packet Type
6. Length of Test Sequence (user data of packet definition in “Baseband Specification” on page 55[vol. 3])

1.1.2.4 Power Control

If adaptive power control is tested, the normal LMP commands will be used. The DUT shall start transmitting at the maximum power and shall reduce/ increase its power by one step on every LMP_incr_power_req or LMP_decr_power_req command received.

1.1.2.5 Switch Between Different Frequency Settings

A change in the frequency selection becomes effective when the LMP procedure is completed:

When the tester receives the LMP_accepted it shall then transmit POLL packets containing the Ack for at least 8 slots (4 transmissions). When these transmissions have been completed the tester shall change to the new frequency hop and whitening settings.

After sending LMP_accepted the DUT shall wait for the LC level Ack for the LMP_accepted. When this is received it shall change to the new frequency hop and whitening settings.

1. It is recommended that the sequence starts with a one; but, as this is irrelevant for measurements, it is also allowed to start with a zero.

There will be an implementation defined delay after sending the LMP_accepted before the TX or loopback test starts. Testers shall be able to cope with this.

Note: Loss of the LMP_Accepted packet will eventually lead to a loss of frequency synchronization that cannot be recovered. Similar problems occur in normal operation, when the hopping pattern changes.

1.1.2.6 Adaptive Frequency Hopping

Adaptive Frequency Hopping (AFH) shall only be used when the Hopping Mode is set to 79 channels (e.g. Hopping Mode = 1) in the LMP_test_control PDU. If AFH is used, the normal LMP commands and procedures shall be used. When AFH is enabled prior to entering test mode it shall continue to be used with the same parameters if Hopping Mode = 1 until the AFH parameters are changed by the LMP_set_AFH PDU.

The channel classification reporting state shall be retained upon entering or exiting Test Mode. The DUT shall change the channel classification reporting state in Test Mode based on control messages from the tester (LMP_channel_classification_req) and from the Host (HCI Write_AFH_Channel_Classification_Mode).

1.1.3 LoopBack test

In loopback, the device under test receives normal baseband packets containing payload *Accepted* from the tester. The received packets shall be decoded in the DUT, and the payload shall be sent back using the same packet type. The return packet shall be sent back in either the slave-to-master transmission slot directly following the transmission of the tester, or it is delayed and sent back in the slave-to-master transmission slot after the next transmission of the tester (see Figure 1.7 to Figure 1.9 on page 239).

There is no signalling to determine or control the mode. The device behavior shall be fixed or adjusted by other means, and shall not change randomly.

The tester can select, whether whitening is on or off. This setting holds for both up- and downlink. For switching the whitening status, the same rules as in Section 1.1.2 on page 232 (Figure 1.4) shall apply.

The following rules apply (for illustration see Figure 1.6 on page 238):

- If the synch word was not detected, the DUT shall not reply.
- If the header error check (HEC) fails, the DUT shall either reply with a NULL packet with the ARQN bit set to NAK or send nothing.
- If the packet contains an LMP message relating to the control of the test mode this command shall be executed and the packet shall not be returned, though ACK or NAK shall be returned as per the usual procedure. Other LMP commands are ignored and no packet is returned.
- The payload FEC is decoded and the payload shall be encoded again for transmission. This allows testing of the FEC handling. If the pure bit error rate shall be determined the tester chooses a packet type without FEC.
- The CRC is shall be evaluated. In the case of a failure, ARQN=NAK shall be returned. The payload shall be returned as received.
  A new CRC for the return packet shall be calculated for the returned payload regardless of whether the CRC was valid or not.
- If the CRC fails for a packet with a CRC and a payload header, the number of bytes as indicated in the (possibly erroneous) payload header shall be looped back.

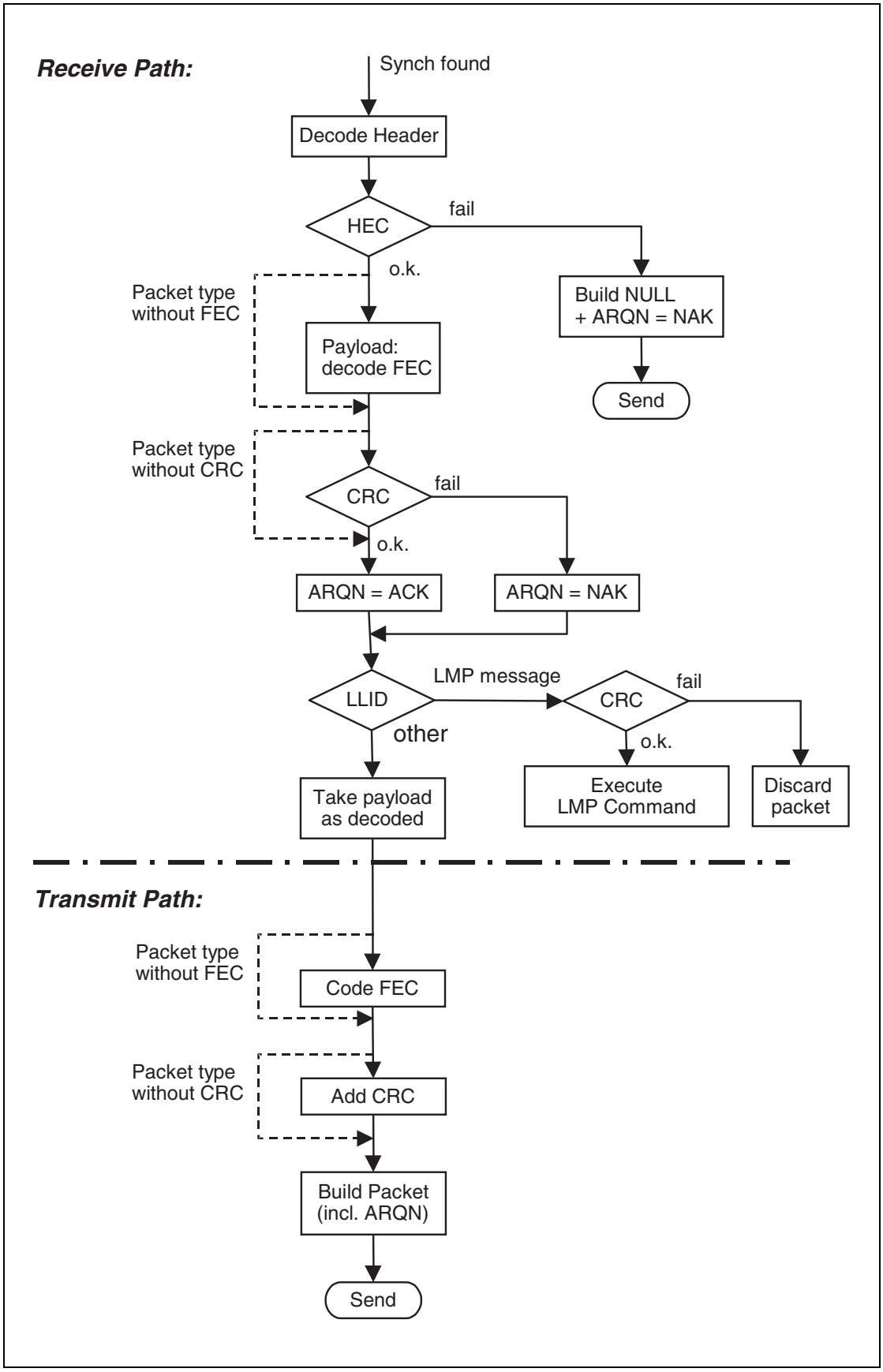


Figure 1.6: DUT Packet Handling in Loop Back Test

The timing for normal and delayed loopback is illustrated in Figure 1.7 to Figure 1.9:

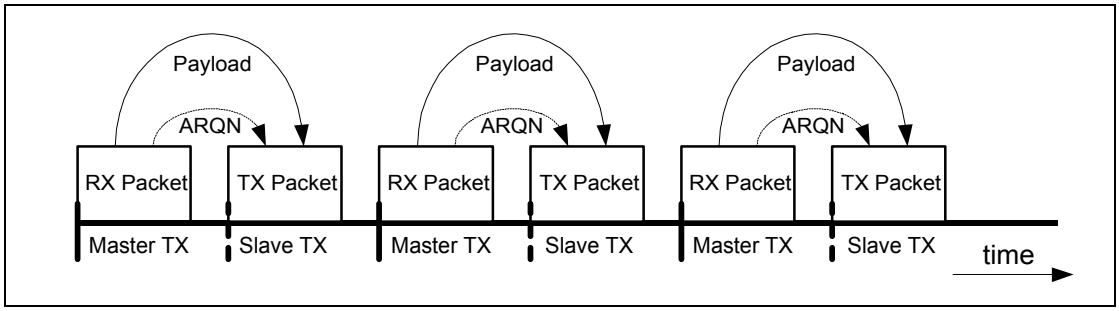


Figure 1.7: Payload & ARQN handling in normal loopback.

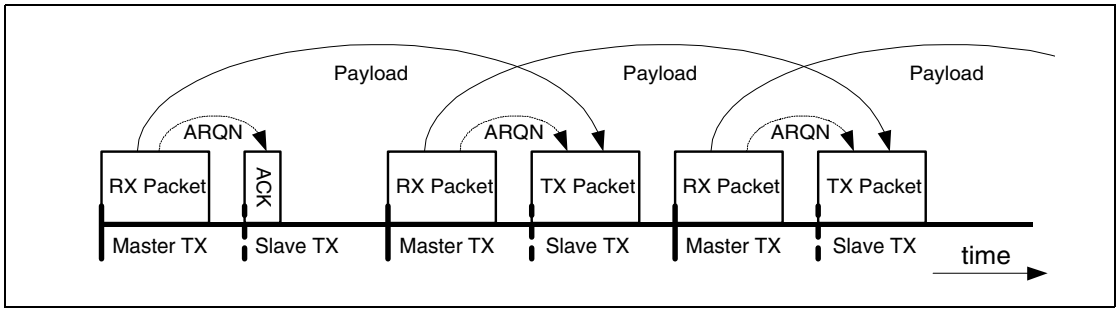


Figure 1.8: Payload & ARQN handling in delayed loopback - start.

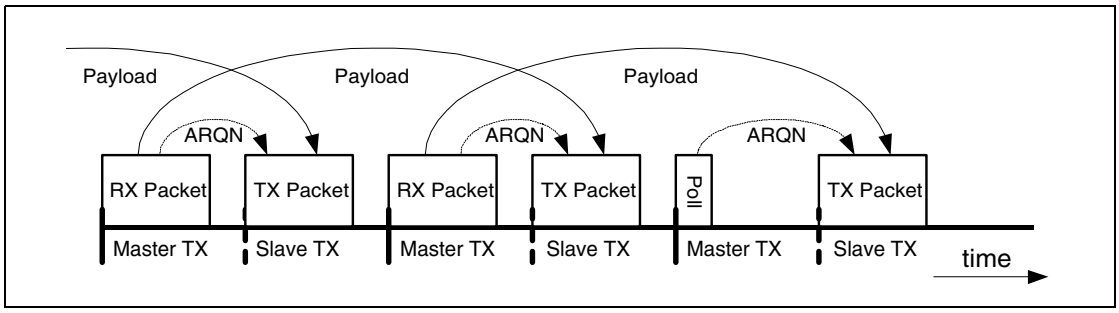


Figure 1.9: Payload & ARQN handling in delayed loopback - end.

The whitening is performed in the same way as it is used in normal active mode.

The following parameters can be set to configure the loop back test:

1. Packet Class[1]
   - ACL Packets
   - SCO Packets
   - eSCO Packets
   - ACL Packets without whitening

1. This is included because, in the future, the packet type numbering may not remain unambiguous.

- SCO Packets without whitening
- eSCO Packets without whitening

2. Frequency Selection
   - Single frequency (independent for RX and TX)
   - Normal hopping
3. Power level: (To be used according radio specification requirements)
   - power control or fixed TX power

The switch of the frequency setting is done exactly as for the transmitter test (see Section 1.1.2.5 on page 235).

1.1.4 Pause test

Pause test is used by testers to put the device under test into Pause Test mode from either the loopback or transmitter test modes.

When an LMP_test_control PDU that specifies Pause Test is received the DUT shall stop the current test and enter Pause Test mode. In the case of a transmitter test this means that no more packets shall be transmitted. While in Pause Test mode the DUT shall respond normally to POLL packets (i.e. responds with a NULL packet). The DUT shall also respond normally to all the LMP packets that are allowed in test mode.

When the test scenario is set to Pause Test all the other fields in the LMP_test_control PDU shall be ignored. There shall be no change in hopping scheme or whitening as a result of a request to pause test.

1.2 REFERENCES

[1] Bluetooth Link Manager Protocol.

[2] CCITT Recommendation O.153 (1992), Basic parameters for the measurement of error performance at bit rates below the primary rate.

[3] ITU-T Recommendation O.150 (1996), General requirements for instrumentation for performance measurements on digital transmission equipment.

[4] Bluetooth Baseband Specification.

2 TEST CONTROL INTERFACE (TCI)

This section describes the Bluetooth Test Control Interface (TCI). The TCI provides a uniform method of accessing the upper interface of the implementation being tested. This facilitates the use of a standardized interface on test equipment used for formal Qualification of implementations.

2.1 INTRODUCTION

2.1.1 Terms used

Conformance testing	Testing according to the applicable procedures given in the Bluetooth Protocol Test Specifications and the Bluetooth Profile Conformance Test Specification when tested against a test system.
HCI	Host Controller Interface
IUT	Implementation Under Test: An implementation of one or more Bluetooth protocols and profiles which is to be studied by testing. This term is used when describing the test concept for products and components equipped with Bluetooth wireless technology as defined in the PRD.
PRD	Bluetooth Qualification Program Reference Document: This document is maintained by the Bluetooth Qualification Review Board and is the reference to specify the functions, organization and processes inside the Bluetooth Qualification program.
TCI	Test Control Interface: The interface and protocol used by the test equipment to send and receive messages to and from the upper interface of the IUT.

2.1.2 Usage of the interface

For all products and components equipped with Bluetooth wireless technology, conformance testing is used to verify the implemented functionality in the lower layers. Conformance testing of the lowest layers requires an upper tester to test the implementation sufficiently well.

In order to avoid that the tester will have to adapt to each and every product and component equipped with Bluetooth wireless technology, the use of the standardized TCI is mandated. This concept puts some burden upon the manufacturer of the IUT in terms of supplying an adapter providing the necessary conversion from/to the IUT's specific interface to the TCI. The adapter can consist of hardware, firmware and software. After qualification testing has been performed the TCI may be removed from the product or component equipped with Bluetooth wireless technology. It is the manufacturer's option to remove it from the qualified product or component equipped with Bluetooth wireless technology.

The TCI is used when qualifying the implemented functionality of the:

- Baseband layer, BB
- Link Manager layer, LM

If support of the Host Controller Interface is claimed by the manufacturer the TCI is used to qualify it.

2.2 TCI CONFIGURATIONS

This section describes the test configurations used when verifying the different Bluetooth requirements. Each layer in the Bluetooth stack is qualified using the procedures described in the layer specific test specification.

2.2.1 Bluetooth RF requirements

For qualification of the Bluetooth Radio Frequency requirements the defined Test Mode is used, see Section 1 on page 231.

Similar to TCI, the specific test mode functionality may be removed from the product or component after qualification, at the discretion of the manufacturer.

2.2.1.1 Required interfaces

For RF qualification only the air interface is required, see Figure 2.1. Depending on the physical design of the IUT it might be necessary to temporarily attach an RF connector for executing the RF tests. As stated in Section 1 on page 231, the Test Mode shall be locally enabled on the IUT for security reasons. The implementation of this local enabling is not subject to standardization.

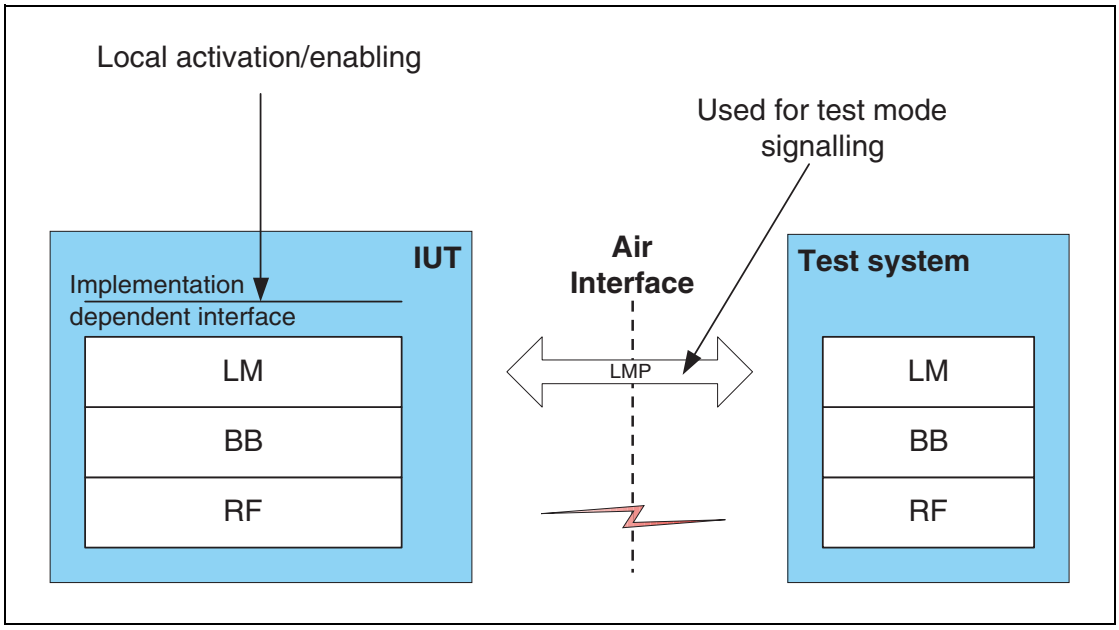


Figure 2.1: General test set-up for RF qualification

2.2.2 Bluetooth protocol requirements

Depending on which of the Bluetooth layers BB, LM or HCI are implemented in the product subject to qualification, the amount of testing needed to verify the Bluetooth protocol requirements differs. Also, the TCI used during the qualification is slightly different. The commands and events necessary for qualification are detailed in the test specifications and only those commands indicated in the test cases to be executed need be implemented.

For other protocols in the Bluetooth stack the TCI is not used. An implementation specific user interface is used to interact with the IUT's upper interface.

2.2.2.1 Required interfaces

For BB, LM and HCI qualification both the air interface of the IUT and the TCI are required. For other protocols both the air interface and the user interface are used as described in the test specification.

2.2.3 Bluetooth profile requirements

For each Bluetooth profile for which conformance is claimed, profile qualification testing is performed to verify the Bluetooth profile requirements. With higher layer protocols the TCI is not used during testing. A user interface specific to the implementation is used to interact with the IUT's upper interface.

2.2.3.1 Required interfaces

For this type of qualification both the air interface and the user interface of the IUT are used as described in the test specification.

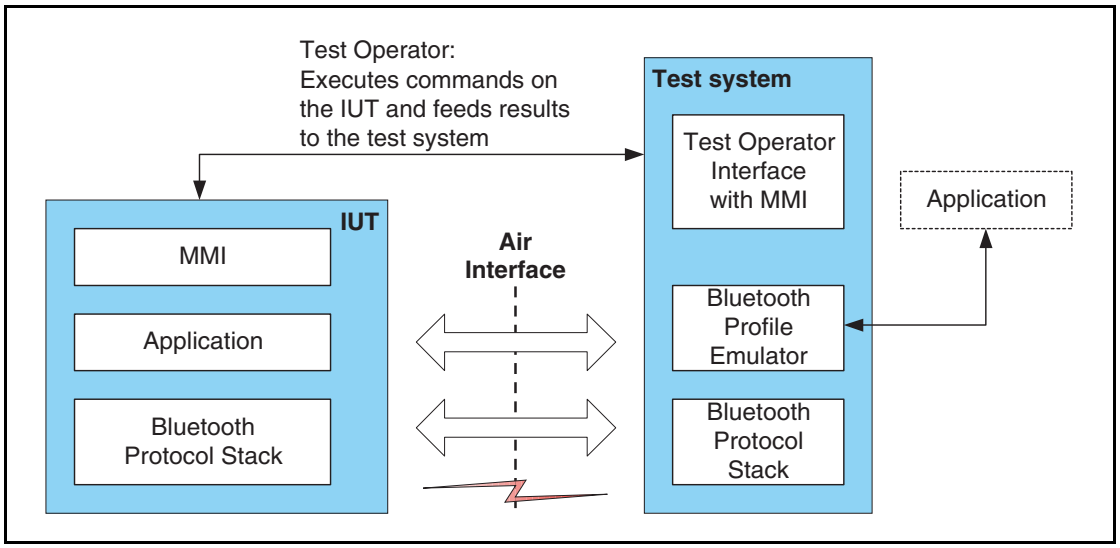


Figure 2.2: General test set-up for profile qualification

2.3 TCI CONFIGURATION AND USAGE

This interface is semantically and syntactically identical to the HCI interface described in “Host Controller Interface Functional Specification” on page 335[vol. 3]. The complete HCI interface is not required in the TCI, but the subset of HCI commands and events necessary for verifying the functionality of the IUT. The exact set of commands and events is specified in the test specifications for the layers subject to testing.

It is worth emphasizing again the TCI is an adapter, logically attached to the upper interface of the IUT. As such the TCI adapts the standardized signalling described here to the implementation specific interface of the IUT.

2.3.1 Transport layers

The method used to convey commands and events between the tester and the IUT’s upper interface can be either physical bearer or “software bearer”.

2.3.1.1 Physical bearer

It is recommended to use one of the transport layers specified for the HCI in. However, other physical bearers are not excluded and may be provided on test equipment. The use of a physical bearer is required for test cases active in category A. Please see the PRD for the definitions of test case categories. Please see the current active Test Case Reference List for the categorization of specific test cases.

2.3.1.2 Software bearer

There is no physical connection between the tester and the IUT’s upper interface. In this case, the manufacturer of the IUT shall supply test software and hardware that can be operated by a test operator. The operator will receive instructions from the tester and will execute them on the IUT. The “software bearer” shall support the same functionality as if using the TCI with a physical bearer. Use of the “software bearer” shall be agreed upon between the manufacturer of the IUT and the test facility that performs the qualification tests. The test facilities can themselves specify requirements placed on such a “software bearer”. Furthermore, the use of a “software bearer” is restricted to test cases active in one of the three lower categories B, C and D.

2.3.2 Baseband and link manager qualification

For the qualification of the link control part of the Baseband layer and for the Link Manager layer, the TCI is used as the interface between the test system and the upper interface of the IUT. The test system accesses the upper interface of the IUT by sending HCI commands and receiving HCI events from the IUT as described in the HCI specification. The required functionality on the TCI depends on the IUT's implemented functionality of the BB and LM layers, and therefore which test cases are executed.

A schematic example in Figure 2.3 shows the test configuration for BB and LM qualification of Bluetooth products which do not support HCI, and use a physical bearer for the TCI. In this example the Test Control (TC) Software represents what the manufacturer has to supply with the IUT when using an external test facility for qualification. The function of the TC Software is to adapt the implementation dependent interface to the TCI.

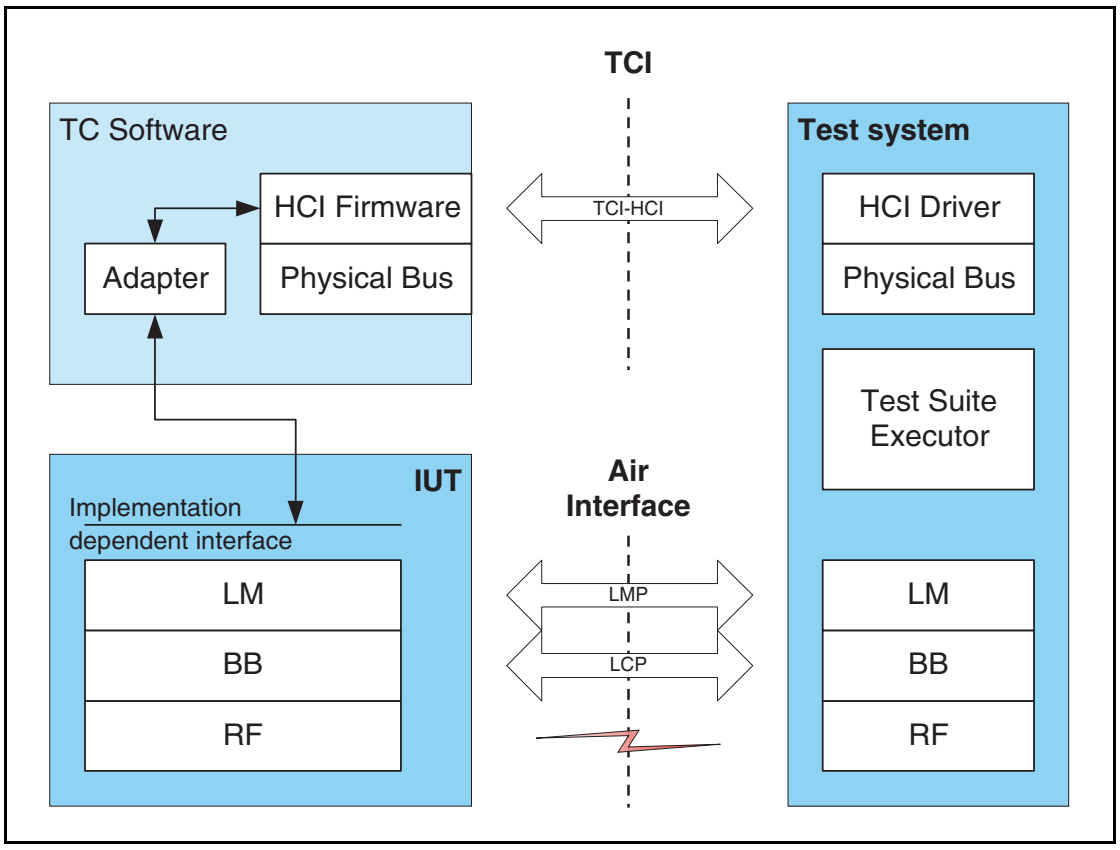


Figure 2.3: BB and LM qualification with TCI physical bearer

Figure 2.4 shows a schematic example of the test configuration for the same Bluetooth product using a “software bearer” for the TCI. Here the function of the Test Control Software is to represent the application that can be controlled by the test operator.

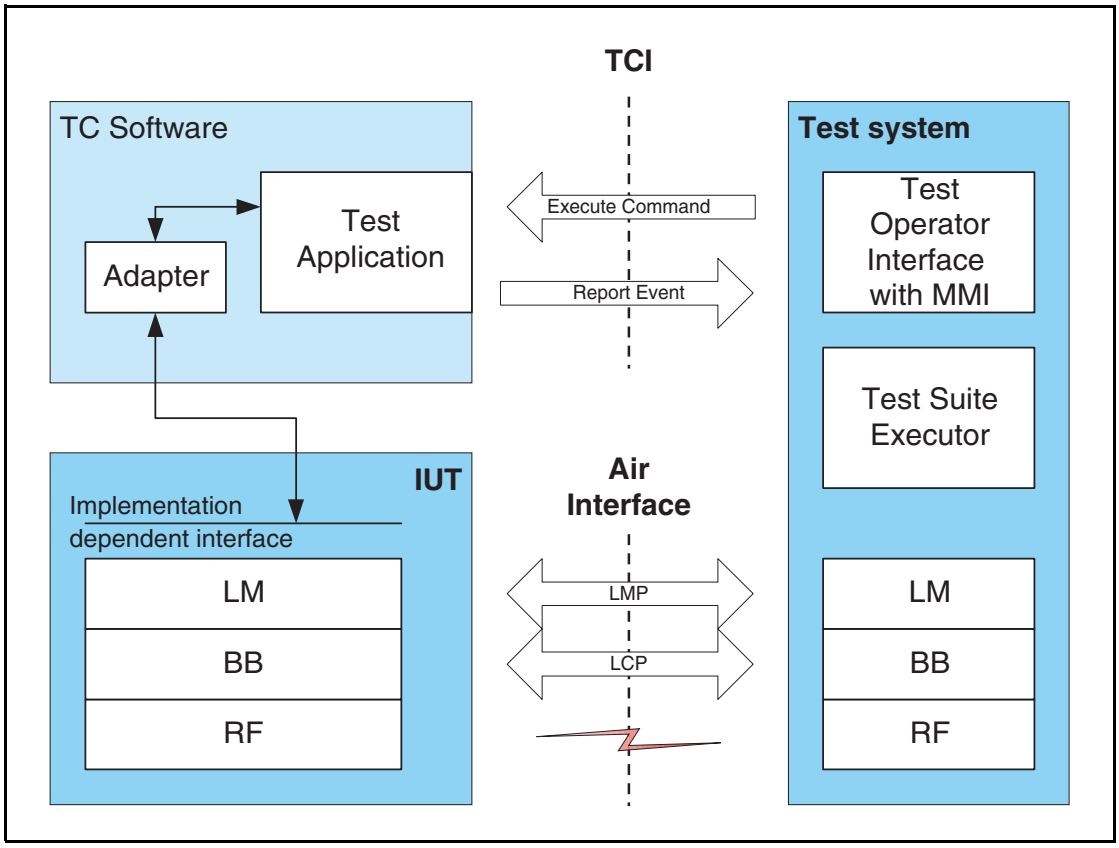


Figure 2.4: BB and LM qualification with “software bearer”

2.3.3 HCI qualification

The TCI may also be used for HCI signalling verification and qualification. The HCI signalling is only verified if support of the HCI functionality is claimed by the manufacturer.

A schematic example in Figure 2.5 shows the test configuration for HCI qualification of Bluetooth products. As can be seen in the figure the implemented HCI is used as the interface to the tester.

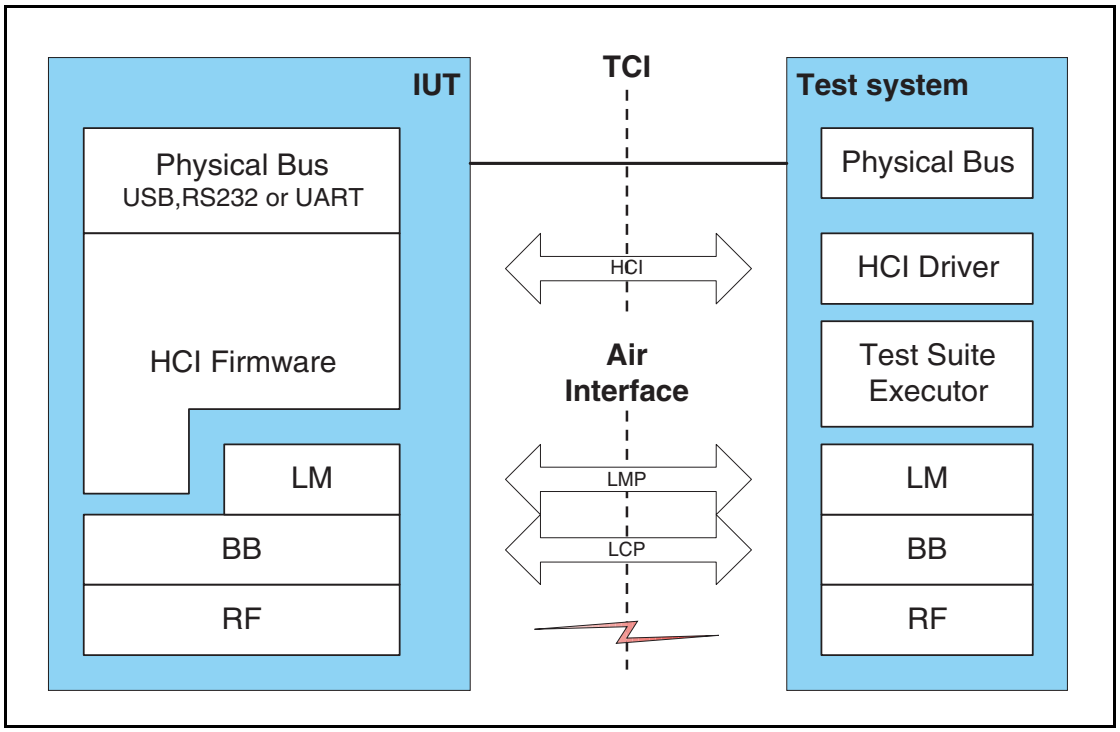


Figure 2.5: General test set-up for HCI qualification

SPECTRUM DIGITAL INCORPORATED

eZdsp™ F2812

Technical Reference

2003

DSP Development Systems

eZdsp™ F2812
Technical Reference

506265-0001 Rev. F
September 2003

SPECTRUM DIGITAL, INC.
12502 Exchange Dr., Suite 440 Stafford, TX. 77477
Tel: 281.494.4505 Fax: 281.494.5310
sales@spectrumdigital.com www.spectrumdigital.com

IMPORTANT NOTICE

Spectrum Digital, Inc. reserves the right to make changes to its products or to discontinue any product or service without notice. Customers are advised to obtain the latest version of relevant information to verify data being relied on is current before placing orders.

Spectrum Digital, Inc. warrants performance of its products and related software to current specifications in accordance with Spectrum Digital's standard warranty. Testing and other quality control techniques are utilized to the extent deemed necessary to support this warranty.

Please be aware, products described herein are not intended for use in life-support appliances, devices, or systems. Spectrum Digital does not warrant, nor is it liable for, the product described herein to be used in other than a development environment.

Spectrum Digital, Inc. assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does Spectrum Digital warrant or represent any license, either express or implied, is granted under any patent right, copyright, or other intellectual property right of Spectrum Digital, Inc. covering or relating to any combination, machine, or process in which such Digital Signal Processing development products or services might be or are used.

WARNING

This equipment is intended for use in a laboratory test environment only. It generates, uses, and can radiate radio frequency energy and has not been tested for compliance with the limits of computing devices pursuant to subpart J of part 15 of FCC rules, which are designed to provide reasonable protection against radio frequency interference. Operation of this equipment in other environments may cause interference with radio communications, in which case the user, at his own expense, will be required to take any measures necessary to correct this interference.

TRADEMARKS

eZdsp is a trademark of Spectrum Digital, Inc.

Contents

List of Figures

List of Tables

About This Manual

This document describes board level operations of the eZdsp™ F2812 based on the Texas Instruments TMS320F2812 Digital Signal Processor.

The eZdsp™ F2812 is a stand-alone module permitting engineers and software developers evaluation of certain characteristics of the TMS320F2812 DSP to determine processor applicability to design requirements. Evaluators can create software to execute onboard or expand the system in a variety of ways.

Notational Conventions

This document uses the following conventions.

The "eZdsp™ F2812" will sometimes be referred to as the "eZdsp".

"eZdsp" will include the socketed or unsocket version

Program listings, program examples, and interactive displays are shown in a special italic typeface. Here is a sample program listing.

equations
!rd = !strobe&rw;

Information About Cautions

This book may contain cautions.
This is an example of a caution statement.
A caution statement describes a situation that could potentially damage your software, hardware, or other equipment. The information in a caution is provided for your protection. Please read each caution carefully.

Related Documents

Texas Instruments TMS320C28x DSP CPU and Instruction Set Reference Guide, literature #SPRU430
Texas Instruments TMS320C28x Assembly Language Tools Users Guide, literature #SPRU513
Texas Instruments TMS320C28x Optimizing C/C++ Compiler User's Guide, literature #SPRU514
Texas Instruments Code Composer Studio Getting Started Guide, literature #SPRU509

Chapter 1

Introduction to the eZdsp™ F2812

This chapter provides a description of the eZdsp™ for the TMS320F2812 Digital Signal Processor, key features, and block diagram of the circuit board.

Topic **Page**

1.0 Overview of the eZdsp™ F2812

The eZdsp™ F2812 is a stand-alone card--allowing evaluators to examine the TMS320F2812 digital signal processor (DSP) to determine if it meets their application requirements. Furthermore, the module is an excellent platform to develop and run software for the TMS320F2812 processor.

The eZdsp™ F2812 is shipped with a TMS320F2812 DSP. The eZdsp™ F2812 allows full speed verification of F2812 code. Two expansion connectors are provided for any necessary evaluation circuitry not provided on the as shipped configuration.

To simplify code development and shorten debugging time, a C2000 Tools Code Composer driver is provided. In addition, an onboard JTAG connector provides interface to emulators, operating with other debuggers to provide assembly language and 'C' high level language debug.

1.1 Key Features of the eZdsp™ F2812

The eZdsp™ F2812 has the following features:

- TMS320F2812 Digital Signal Processor
- 150 MIPS operating speed
- 18K words on-chip RAM
- 128K words on-chip Flash memory
- 64K words off-chip SRAM memory
- 30 MHz. clock
- 2 Expansion Connectors (analog, I/O)
- Onboard IEEE 1149.1 JTAG Controller
- 5-volt only operation with supplied AC adapter
- TI F28xx Code Composer Studio tools driver
- On board IEEE 1149.1 JTAG emulation connector

1.2 Functional Overview of the eZdsp™ F2812

Figure 1-1 shows a block diagram of the basic configuration for the eZdsp™ F2812. The major interfaces of the eZdsp are the JTAG interface, and expansion interface.

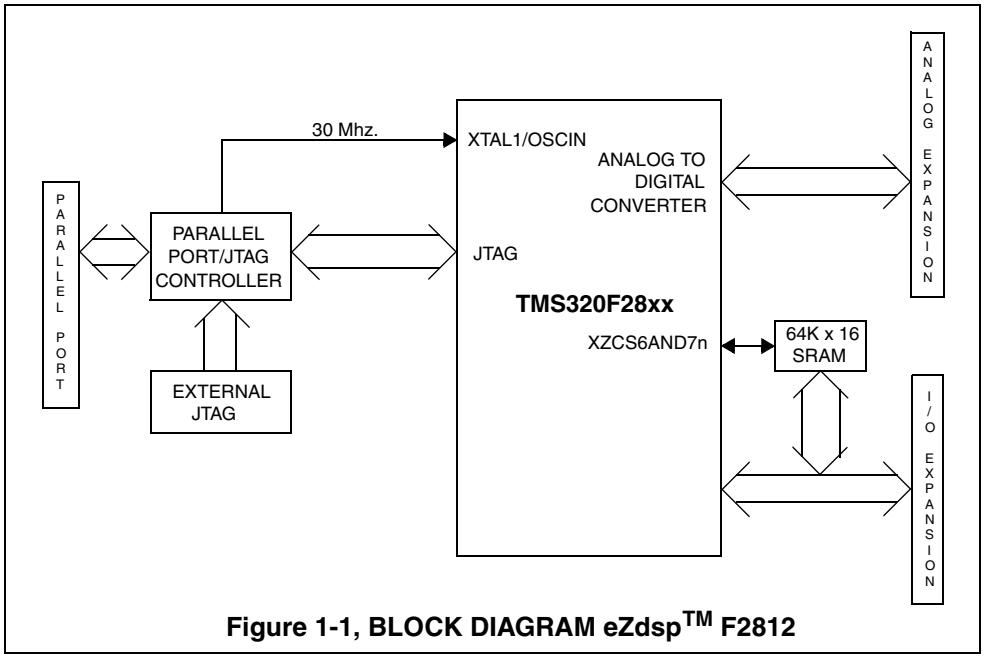


Figure 1-1, BLOCK DIAGRAM eZdsp™ F2812

Chapter 2

Operation of the eZdsp™ F2812

This chapter describes the operation of the eZdsp™ F2812, key interfaces and includes a circuit board outline.

Topic **Page**

2.0 The eZdsp™ F2812 Operation

This chapter describes the eZdsp™ F2812, key components, and operation. Information on the eZdsp's various interfaces is also included. The eZdsp™ F2812 consists of four major blocks of logic:

- Analog Interface Connector
- I/O Interface Connector
- JTAG Interface
- Parallel Port JTAG Controller Interface

2.1 The eZdsp™ F2812 Board

The eZdsp™ F2812 is a 5.25 x 3.0 inch, multi-layered printed circuit board, powered by an external 5-Volt only power supply. Figure 2-1 shows the layout of both the socketed and unsocketed version of the F2812 eZdsp.

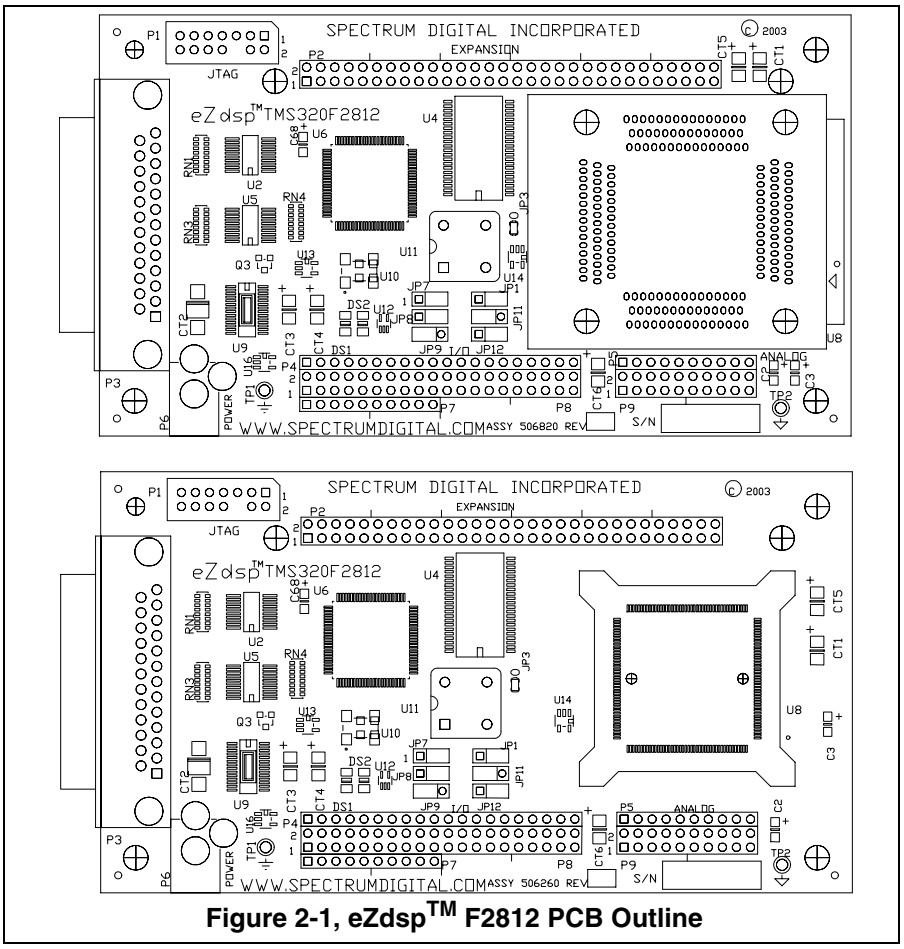


Figure 2-1, eZdsp™ F2812 PCB Outline

2.1.1 Power Connector

The eZdsp™ F2812 is powered by a 5-Volt only power supply, included with the unit. The unit requires 500mA. The power is supplied via connector P6. If expansion boards are connected to the eZdsp, a higher amperage power supply may be necessary. Section 2.3.6 provides more information on connector P6.

2.2 eZdsp™ F2812 Memory

The eZdsp includes the following on-chip memory:

- 128K x 16 Flash
- 2 blocks of 4K x 16 single access RAM (SARAM)
- 1 block of 8K x 16 SARAM
- 2 blocks of 1K x 16 SARAM

In addition 64K x 16 off-chip SRAM is provided. The processor on the eZdsp can be configured for boot-loader mode or non-boot-loader mode.

The eZdsp can load ram for debug or FLASH ROM can be loaded and run. For larger software projects it is suggested to do a initial debug with on eZdsp F2812 module which supports a total RAM environment. With careful attention to the I/O mapping in the software the application code can easily be ported to the F2812.

The table below shows the external chip select signal and its use.

Table 1: External Chip Select and Usage

Chip Select Signal	Use
XZCS0AND1n	Expansion header
XZCS2n	Expansion Header
XZCS6AND7n	External SRAM

2.2.1 Memory Map

The figure below shows the memory map configuration on the eZdsp™ F2812.

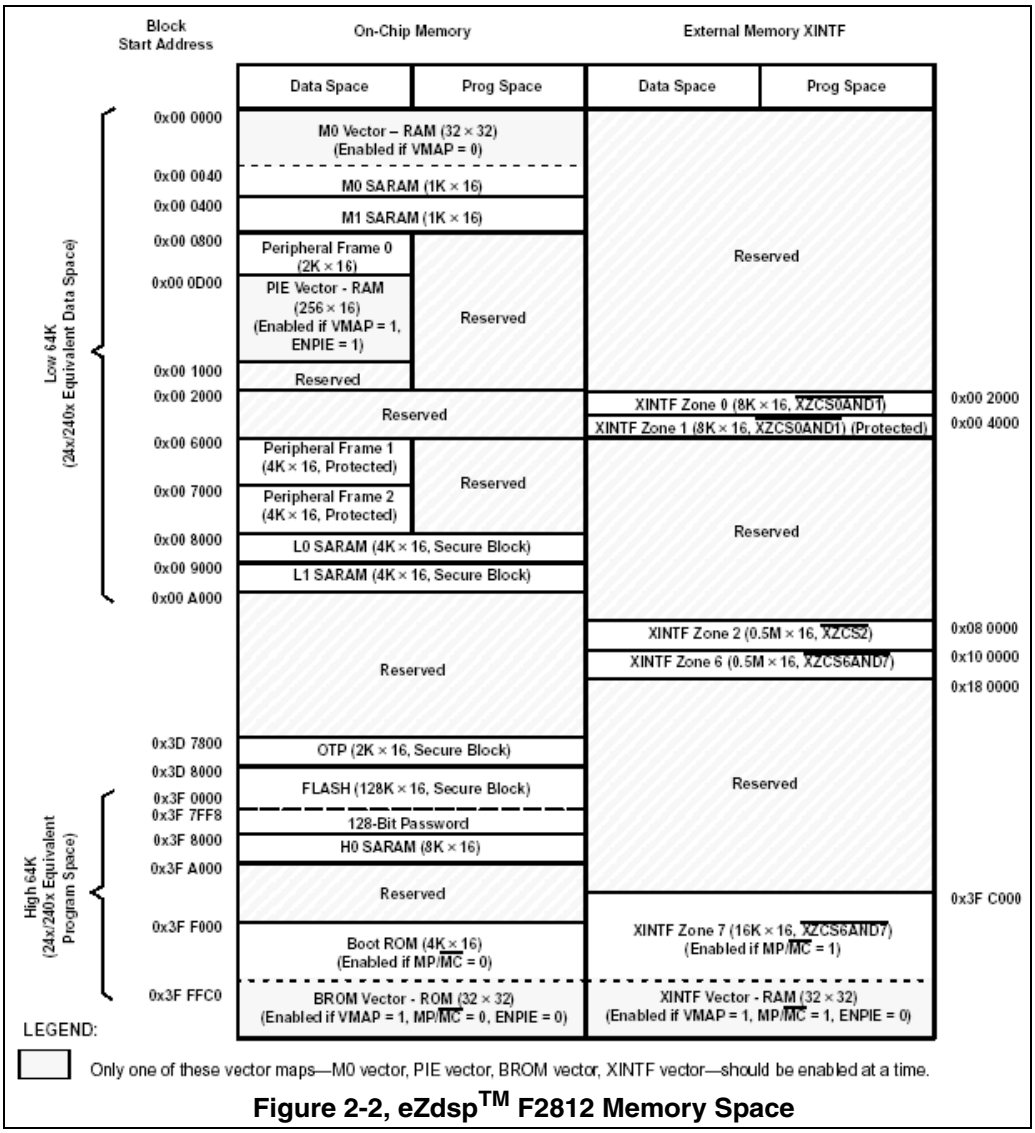


Figure 2-2, eZdsp™ F2812 Memory Space

Note: The on-chip flash memory has a security key which can prevent visibility when enabled.

2.3 eZdsp™ F2812 Connectors

The eZdsp™ F2812 has five connectors. Pin 1 of each connector is identified by a square solder pad. The function of each connector is shown in the table below:

Table 2: eZdsp™ F2812 Connectors

Connector	Function
P1	JTAG Interface
P2	Expansion
P3	Parallel Port/JTAG Controller Interface
P4/P8/P7	I/O Interface
P5/P9	Analog Interface
P6	Power Connector

The diagram below shows the position of each connector

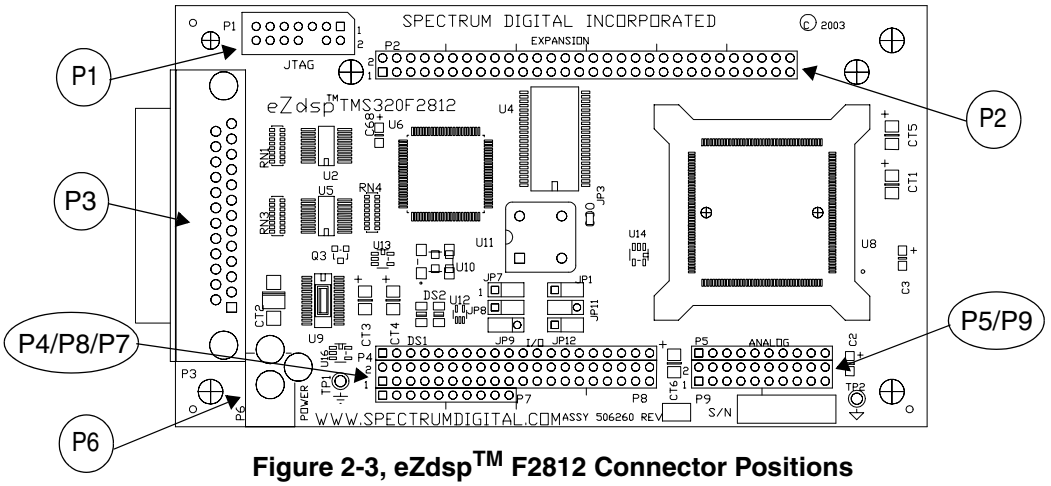


Figure 2-3, eZdsp™ F2812 Connector Positions

2.3.1 P1, JTAG Interface

The eZdsp^{TM} F2812 is supplied with a 14-pin header interface, P1. This is the standard interface used by JTAG emulators to interface to Texas Instruments DSPs.

The positions of the 14 pins on the P1 connector are shown in the diagram below as viewed from the top of the eZdsp.

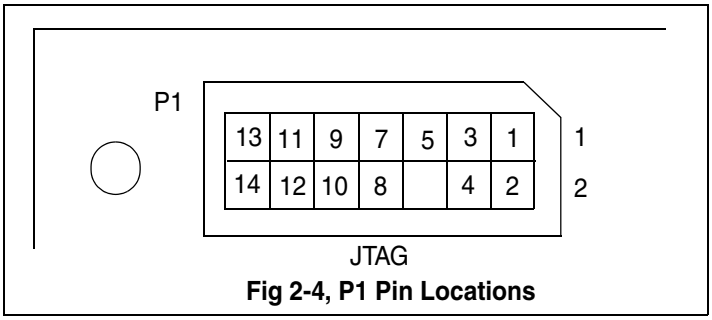


Fig 2-4, P1 Pin Locations

The definition of P1, which has the JTAG signals is shown below.

Table 3: P1, JTAG Interface Connector

Pin #	Signal	Pin #	Signal
1	TMS	2	TRST-
3	TDI	4	GND
5	PD (+5V)	6	**no pin**
7	TDO	8	GND
9	TCK-RET	10	GND
11	TCK	12	GND
13	EMU0	14	EMU1

WARNING !

The TMS320F2812 supports +3.3V Input/Output levels which are NOT +5V tolerant. Connecting the eZdsp to a system with +5V Input/Output levels will damage the TMS320F2812. If the eZdsp is connected to another target then the eZdsp must be powered up first and powered down last to prevent lactchup conditions.

2.3.2 P2, Expansion Interface

The positions of the 60 pins on the P2 connector are shown in the diagram below as viewed from the top of the eZdsp.

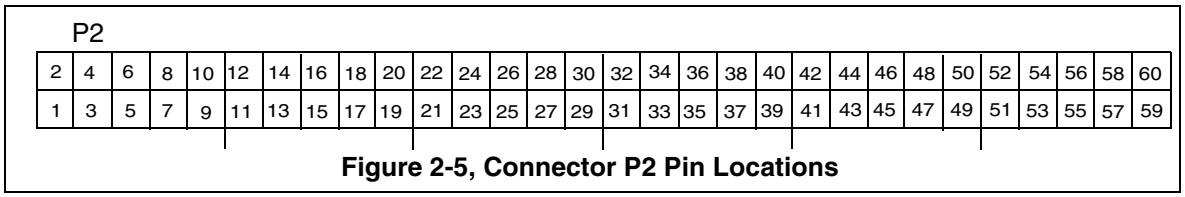


Figure 2-5, Connector P2 Pin Locations

The definition of P2, which has the I/O signal interface is shown below.

Table 4: P2, Expansion Interface Connector

Pin #	Signal	Pin #	Signal
1	+3.3V/+5V/NC *	2	+3.3/+5V/NC *
3	XD0	4	XD1
5	XD2	6	XD3
7	XD4	8	XD5
9	XD6	10	XD7
11	XD8	12	XD9
13	XD10	14	XD11
15	XD12	16	XD13
17	XD14	18	XD15
19	XA0	20	XA1
21	XA2	22	XA3
23	XA4	24	XA5
25	XA6	26	XA7
27	XA8	28	XA9
29	XA10	30	XA11
31	XA12	32	XA13
33	XA14	34	XA15
35	GND	36	GND
37	XZCS0AND1n	38	XZCS2n
39	XREADY	40	10K Pull-up
41	XRnW	42	10K Pull-up
43	XWE	44	XRDn
45	+3.3V	46	XNMI/INT13
47	XRSn/RSn	48	No connect
49	GND	50	GND
51	GND	52	GND
53	XA16	54	XA17
55	XA18	56	XHOLDn
57	XHOLDAn	58	No connect
59	No connect	60	No connect

* Default is No Connect (NC). User can jumper to +3.3V or +5V on backside of eZdsp with JP5.

2.3.3 P3, Parallel Port/JTAG Interface

The eZdsp™ F2812 uses a custom parallel port-JTAG interface device. This device incorporates a standard parallel port interface that supports ECP, EPP, and SPP8/bidirectional communications. The device has direct access to the integrated JTAG interface. Drivers for C2000 Code Composer tools are shipped with the eZdsp modules

2.3.4 P4/P8/P7, I/O Interface

The connectors P4, P8, and P7 present the I/O signals from the DSP. The layout of these connectors are shown below.

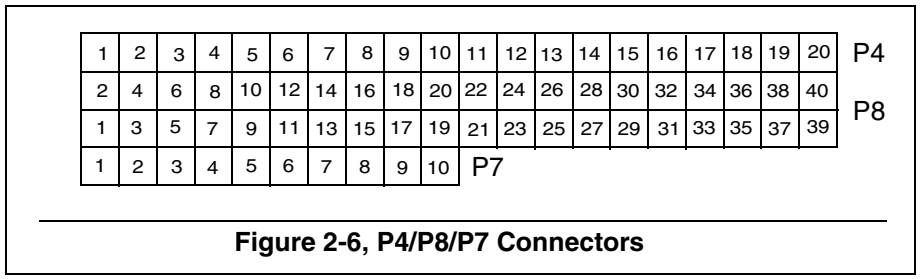


Figure 2-6, P4/P8/P7 Connectors

The pin definition of P4/P8 connectors are shown in the table below.

Table 5: P4/P8, I/O Connectors

P4 Pin #	P4 Signal	P8 Pin #	P8 Signal	P8 Pin #	P8 Signal
1	+3.3V/+5V/NC *	1	+3.3V/+5V/NC *	2	+3.3V/+5V/NC *
2	XINT2/ADCSOC	3	SCITXDA	4	SCIRXDA
3	MCLKXA	5	XINT1n/XBIOn	6	CAP1/QEP1
4	MCLKRA	7	CAP2/QEP2	8	CAP3/QEPI1
5	MFSXA	9	PWM1	10	PWM2
6	MFSRA	11	PWM3	12	PWM4
7	MDXA	13	PWM5	14	PWM6
8	MDRA	15	T1PWM/T1CMP	16	T2PWM/T2CMP
9	No connect	17	TDIRA	18	TCLKINA
10	GND	19	GND	20	GND
11	CAP5/QEP4	21	No connect	22	XINT1N/XBIOn
12	CAP6/QEPI2	23	SPISIMOA	24	SPISOMIA
13	T3PWM/T3CMP	25	SPICLKA	26	SPISTEA
14	T4PWM/T4CMP	27	CANTXA	28	CANRXA
15	TDIRB	29	XCLKOUT	30	PWM7
16	TCLKINB	31	PWM8	32	PWM9
17	XF/XPLLDISn	33	PWM10	34	PWM11
18	SCITXDB	35	PWM12	36	CAP4/QEP3
19	SCIRXDB	37	T1CTRIP/PDPINTAn	38	T3CTRIP/PDPINTBn
20	GND	39	GND	40	GND

* Default is No Connect (NC). User can jumper to +3.3V or +5V on backside of eZdsp with JP4.

The pin definition of P7 connector is shown in the table below.

Table 6: P7, I/O Connector

P7 Pin #	P7 Signal
1	C1TRIPn
2	C2TRIPn
3	C3TRIPn
4	T2CTRIPn/EVASOCn
5	C4TRIPn
6	C5TRIPn
7	C6TRIPn
8	T4CTRIPn/EVBSOCn
9	No connect
10	GND

2.3.5 P5/P9, Analog Interface

The position of the 30 pins on the P5/P9 connectors are shown in the diagram below as viewed from the top of the eZdsp.

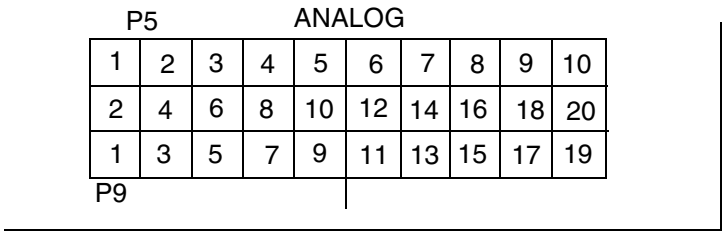


Figure 2-7, Connector P5/P9 Pin Locations

The definition of P5/P9 signals are shown in the table below.

Table 7: P5/P9, Analog Interface Connector

P5 Pin #	Signal	P9 Pin #	Signal	P9 Pin #	Signal
1	ADCINB0	1	GND	2	ADCINA0
2	ADCINB1	3	GND	4	ADCINA1
3	ADCINB2	5	GND	6	ADCINA2
4	ADCINB3	7	GND	8	ADCINA3
5	ADCINB4	9	GND	10	ADCINA4
6	ADCINB5	11	GND	12	ADCINA5
7	ADCINB6	13	GND	14	ADCINA6
8	ADCINB7	15	GND	16	ADCINA7
9	ADCREFM	17	GND	18	VREFLO *
10	ADCREFP	19	GND	20	No connect

* Connect VREFLO to AGND or VREFLO of target system for proper ADC operation.

2.3.6 P6, Power Connector

Power (5 volts) is brought onto the eZdsp^{TM} F2812 via the P6 connector. The connector has an outside diameter of 5.5 mm. and an inside diameter of 2 mm. The position of the P6 connector is shown below.

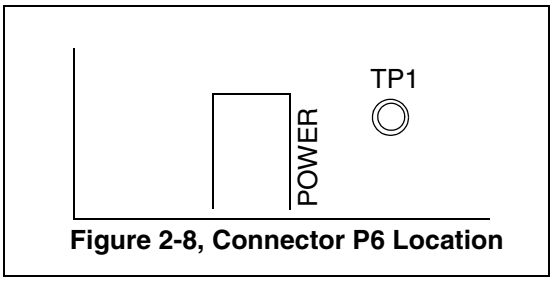


Figure 2-8, Connector P6 Location

The diagram of P6, which has the input power is shown below.

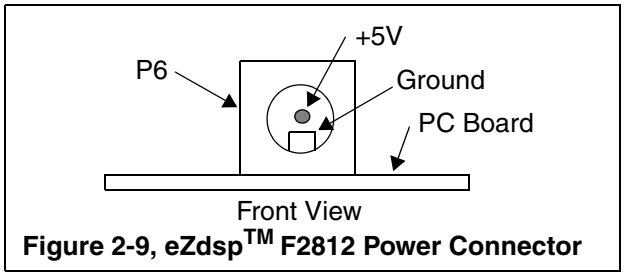


Figure 2-9, eZdsp^{TM} F2812 Power Connector

2.3.7 Connector Part Numbers

The table below shows the part numbers for connectors which can be used on the eZdsp™ F2812. Part numbers from other manufacturers may also be used.

Table 8: eZdsp™ F2812 Suggested Connector Part Numbers

Connector	Male Part Numbers	Female Part Numbers
P1	SAMTEC TSW-1-10-07-G-T	SAMTEC SSW-1-10-01-G-T
P2	SAMTEC TSW-1-20-07-G-T	SAMTEC SSW-1-20-01-G-T

*SSW or SSQ Series can be used

2.4 eZdsp™ F2812 Jumpers

The eZdsp™ F2812 has 8 jumpers available to the user which determine how features on the eZdsp™ F2812 are utilized. The table below lists the jumpers and their function. The following sections describe the use of each jumper.

Table 9: eZdsp™ F2812 Jumpers

Jumper #	Size	Function	Position As Shipped From Factory
JP1	1 x 3	XMP/MCn	2-3
JP4	1 x 3	+3.3/5 Volts to P8,P4	Not connected
JP5	1 x 3	+3.3/5 Volts to P2	Not Connected
JP7	1 x 2	Boot Mode 3	2-3
JP8	1 x 3	Boot Mode 2	2-3
JP9	1 x 3	PLL Disable	1-2
JP11	1 x 3	Boot Mode 1	1-2
JP12	1 x 3	Boot Mode 0	2-3

WARNING!
Unless noted otherwise, all 1x3 jumpers must be installed in either the 1-2 or 2-3 position

The diagram below shows the positions of six jumpers on the component side of the eZdsp™ F2812.

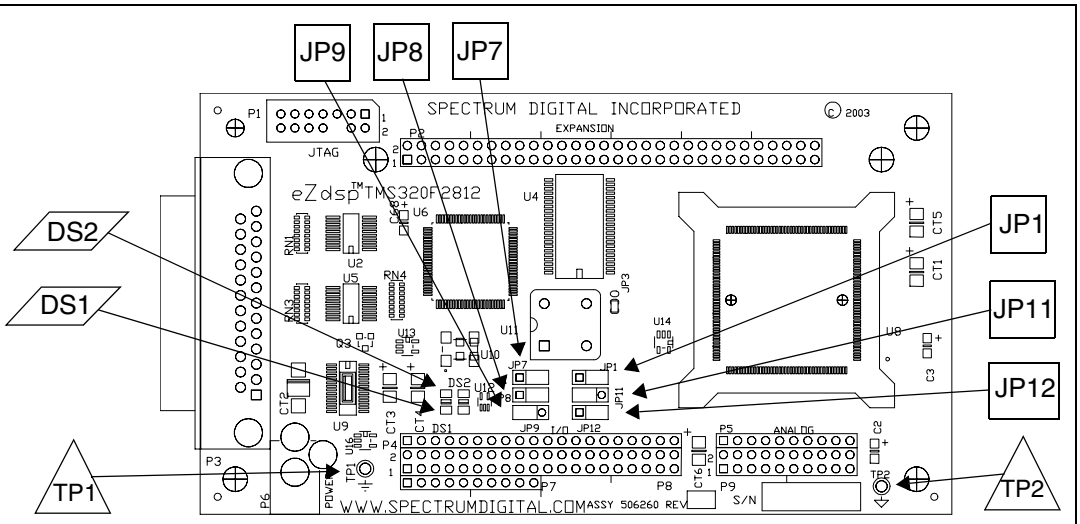


Figure 2-10, eZdsp™ F2812 Jumper Positions

2.4.1 JP1, XMP/MCn Select

Jumper JP1 is used to select the XMP/MCn option. The 1-2 selection allows the DSP to operate in the Microcontroller mode. The 2-3 selection allow the DSP to operate in the Microprocessor mode. The positions are shown in the table below.

Table 10: JP1, XMP/MCn Select

Position	Function
1-2	Microprocessor mode
2-3 *	Microcomputer mode

* as shipped from factory

2.4.2 JP4, JP5 Voltage Jumpers

Jumpers JP4 and JP5 are an unpopulated jumpers on the bottom side of the board that provide either +3.3 volts or +5 volts to pins on the expansion connectors. These jumpers are shipped uninstalled to prevent accidental damage by connecting wires or circuitry to the expansion connector. The user may connect these jumpers by installing a jumper wire or zero ohm resistor. The position of these jumpers are shown in the figure below.

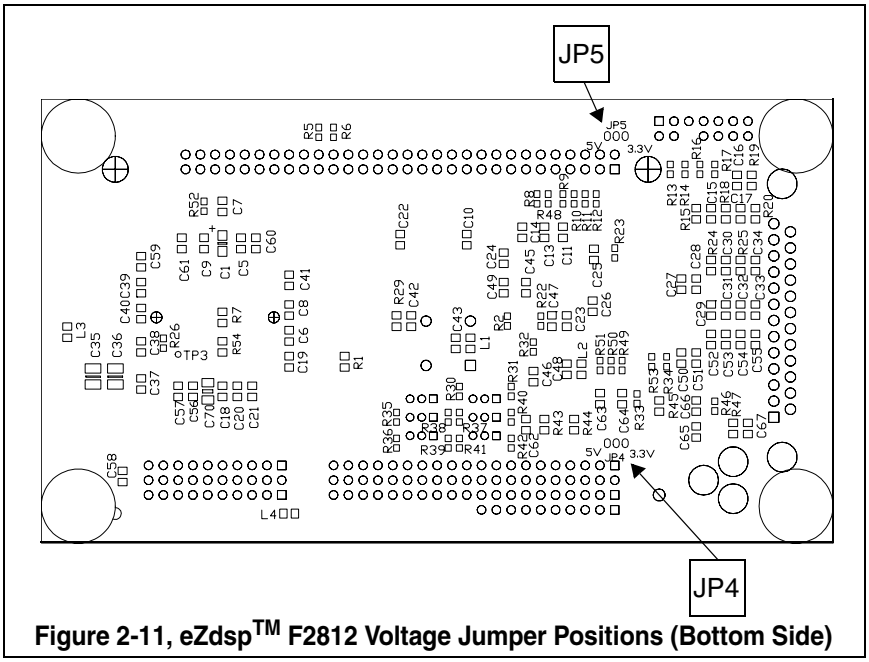
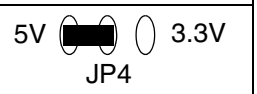
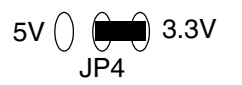
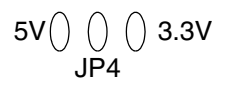


Figure 2-11, eZdsp^{TM} F2812 Voltage Jumper Positions (Bottom Side)

2.4.2.1 JP4, +3.3/5 Volts for P8, P4

Jumper JP4 allows the user to provide either +3.3 or +5 volts to pins 1 and 2 of expansion connector P8, and pin 1 of P4. The settings for this jumper are shown in the table below

Table 11: JP4, +3.3/5 Volts for P8, P4

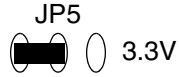
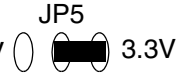
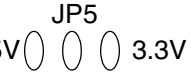
Position	Function	Jumper Position
1-2	Connect +5 Volts to P8, pins 1,2, and P4 pin1	
2-3	Connect +3.3 Volts to P8, pins 1,2, and P4 pin1	
No connect *		

* As shipped from factory

2.4.2.2 JP5, +3.3/5 Volts for P2

Jumper JP5 allows the user to provide either +3.3 or +5 volts to pins 1 and 2 of expansion connector P2. The settings for this jumper are shown in the table below.

Table 12: JP5, +3.3/5 Volts for P2

Position	Function	Jumper Position
1-2	Connect +5 Volts to P2, pins 1,2	
2-3	Connect +3.3 Volts to P2, pins 1,2	
No connect *		

* As shipped from factory

2.4.3 JP7, JP8, JP11, JP12, Boot Mode Select

Jumpers JP7, JP8, JP11, JP12 are used to determine what mode the DSP will use for bootloading on power up. To set a signal high, place the jumper in the 1-2 position. For a low signal, use the 2-3 position.The options are shown in the table below.

Table 13: JP7, JP8, JP11, JP12, Boot Mode Select

JP7, BOOT3 SCITXDA	JP8, BOOT2 MDXA	JP11, BOOT1 SPISTEA	JP12, BOOT0 SPICLKA	MODE
1	X	X	X	FLASH
0	1	X	X	SPI
0	0	1	1	SCI
0	0	1	0	H0 *
0	0	0	1	OTP
0	0	0	0	PARALLEL

* factory default

2.4.4 JP9, PLL Disable

Jumper JP9 is used to enable/disable the use of the Phase Lock Loop (PLL) logic on the DSP. The selection of the 1-2 position enables the use of the PLL. If the 2-3 position is used the PLL is disabled. This signal is latched at reset and may be used as XF after reset. The positions are shown in the table below.

Table 14: JP9, PLL Disable

Position	Function
1-2 *	PLL Enabled
2-3	PLL disabled

***** as shipped from the factory

2.5 LEDs

The eZdsp™ F2812 has two light-emitting diodes. DS1 indicates the presence of +5 volts and is normally ‘on’ when power is applied to the board. DS2 is under software control and is tied to the XF pin on the DSP through a buffer. These are shown in the table below.

Table 15: LEDs

LED #	Color	Controlling Signal
DS1	Green	+5 Volts
DS2	Green	XF bit (XF high = on)

2.6 Test Points

The eZdsp™ F2812 has two test points. The signals they are tied to are shown in the table below.

Table 16: Test Points

Test Point	Signal
TP1	Ground
TP2	Analog Ground

Appendix A

eZdsp™ F2812 Schematics

The schematics for the eZdsp™ F2812 can be found on the CD-ROM that accompanies this board. The schematics were drawn on ORCAD.

The schematics are correct for both the socketed and unsocketed version of the eZdsp™.

WARNING !

The TMS320F2812 supports +3.3V Input/Output levels which are NOT +5V tolerant. Connecting the eZdsp to a system with +5V Input/Output levels will damage the TMS320F2812. If the eZdsp is connected to another target then the eZdsp must be powered up first and powered down last to prevent lactchup conditions.

Design Notes:

1. The TMS320F2812 X1/CLKIN pin is +1.8 volt input. The clock input is buffered with a SN74LVC1G14 whose supply is +1.8 volts. This provides +3.3 volts to the +1.8 volt clock translation. Refer to sheet 4 of the schematics.

REVISIONS

REV	DESCRIPTION	DATE	APPROVED
A	PROTOTYPES	04-April-2002	
B	PRODUCTION RELEASE	07-June-2002	
C	UPDATE PER SPRS174G	20-Jan-2003	

The TMS320F2812 EzDSP design is based on preliminary information(SPRS174G) for the TMS320F2812 device. This schematic is subject to change without notification. Spectrum Digital Inc. assumes no liability for applications assistance, customer product design or infringement of patents described herein.

REVISION STATUS OF SHEETS									
REV									
SH	1	2	3	4	5	6			
REV	C	C	B	C	C	C			

NEXT ASSY | USED ON | APPLICATION

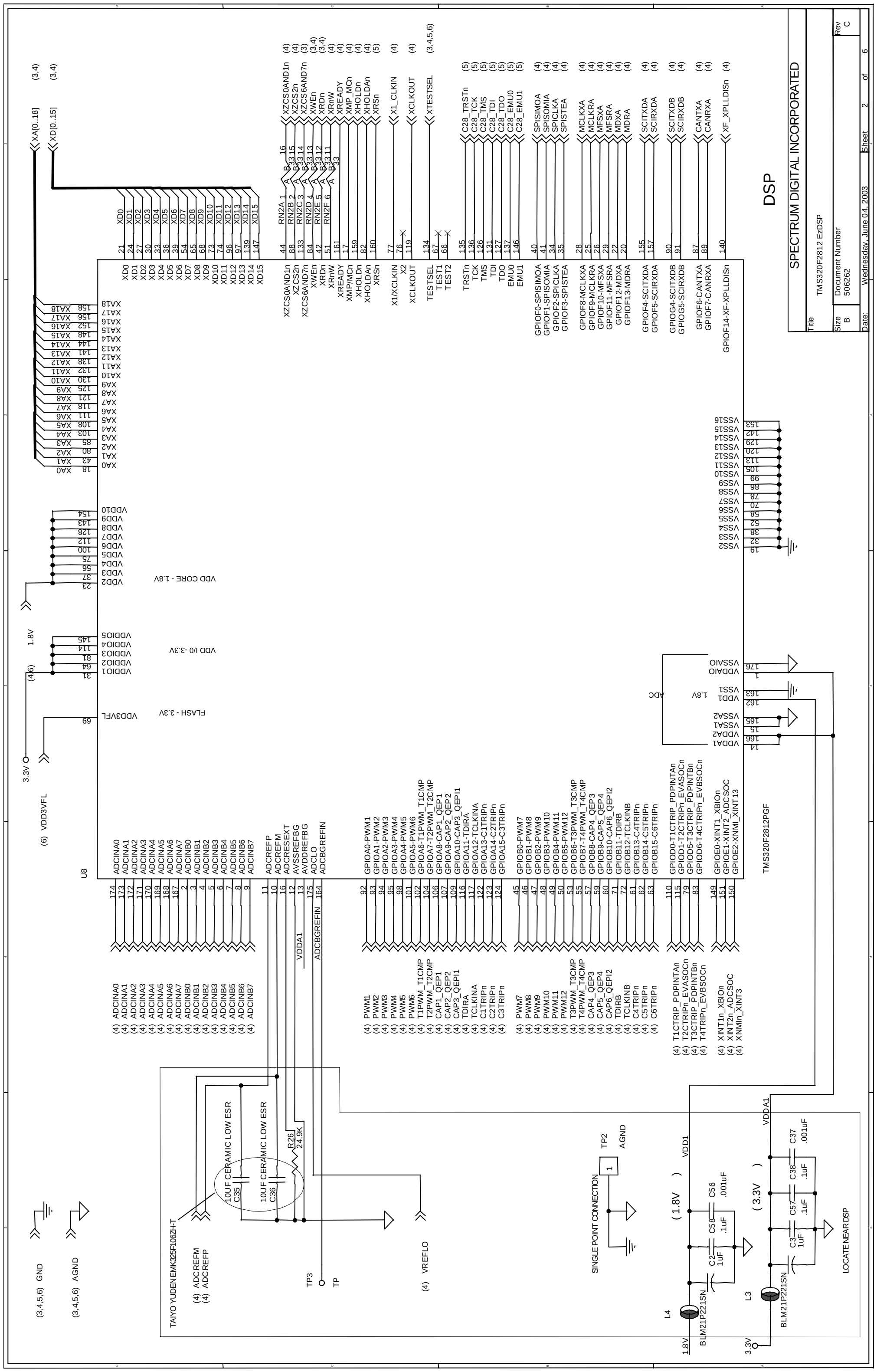
DWN DATE
CHK DATE
ENGR DATE
ENGR MGR DATE
QA DATE
MFG DATE
RLSE DATE

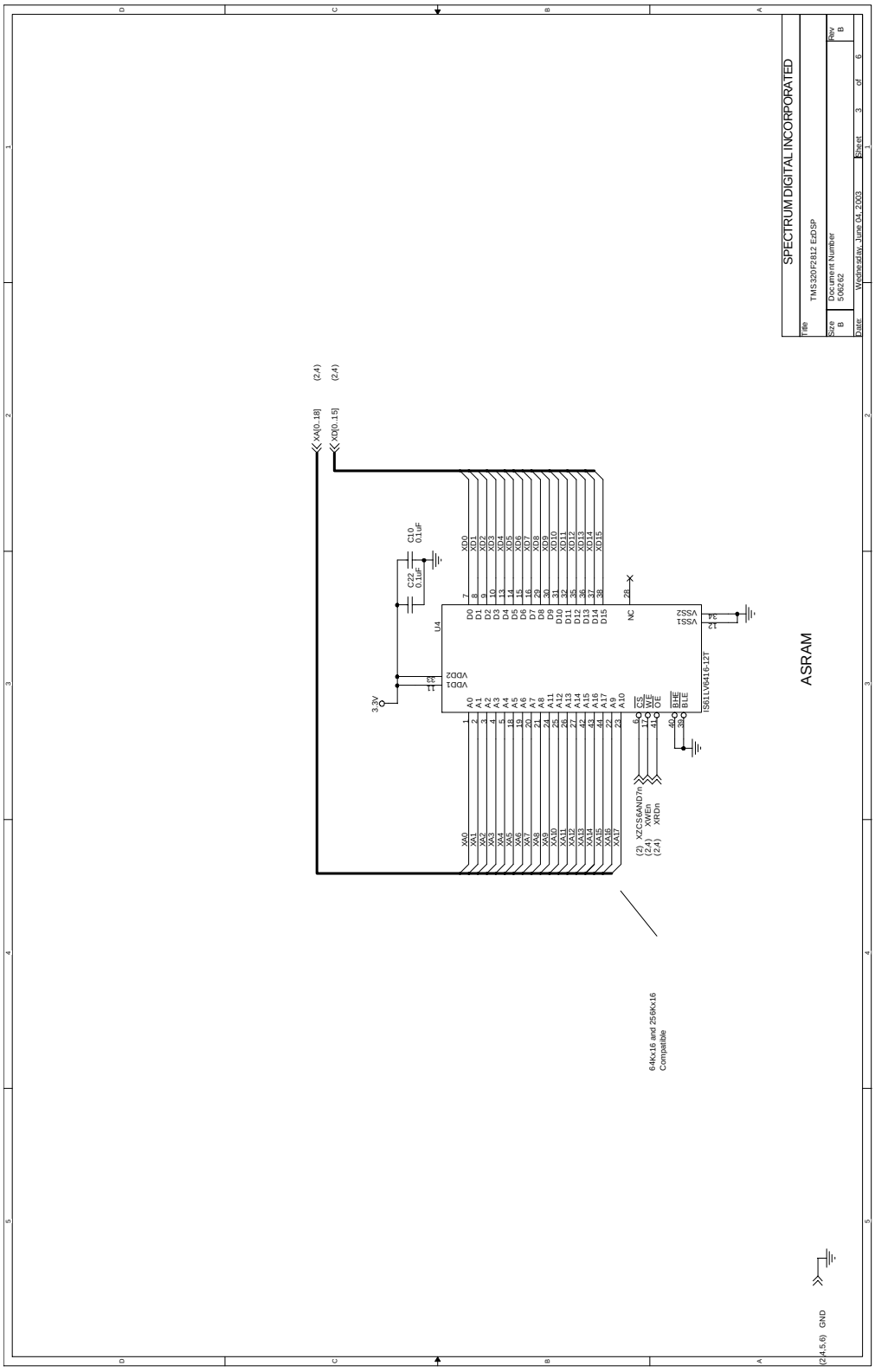
SPECTRUM DIGITAL INCORPORATED

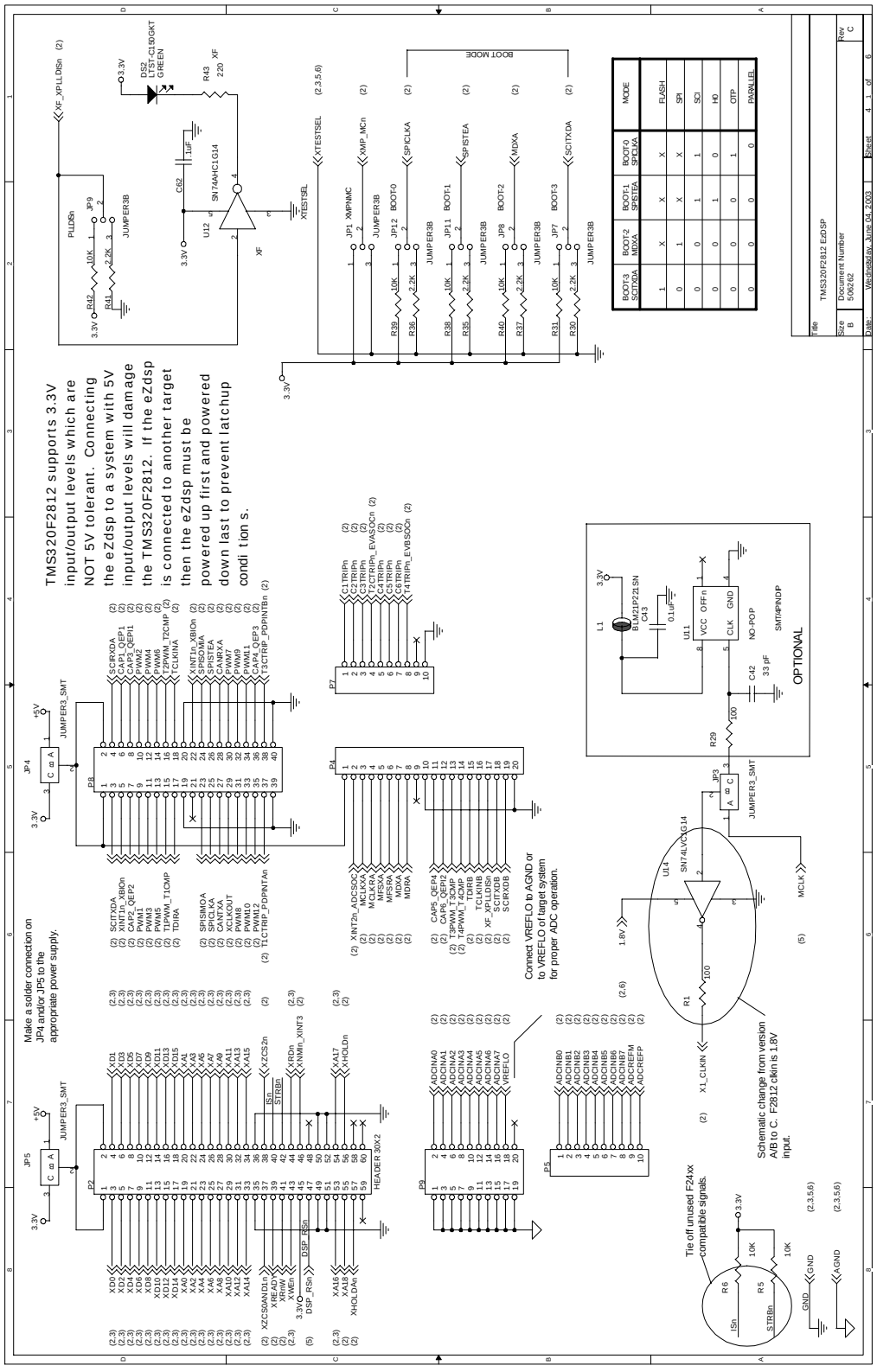
Title: TMS320F2812 EzDSP

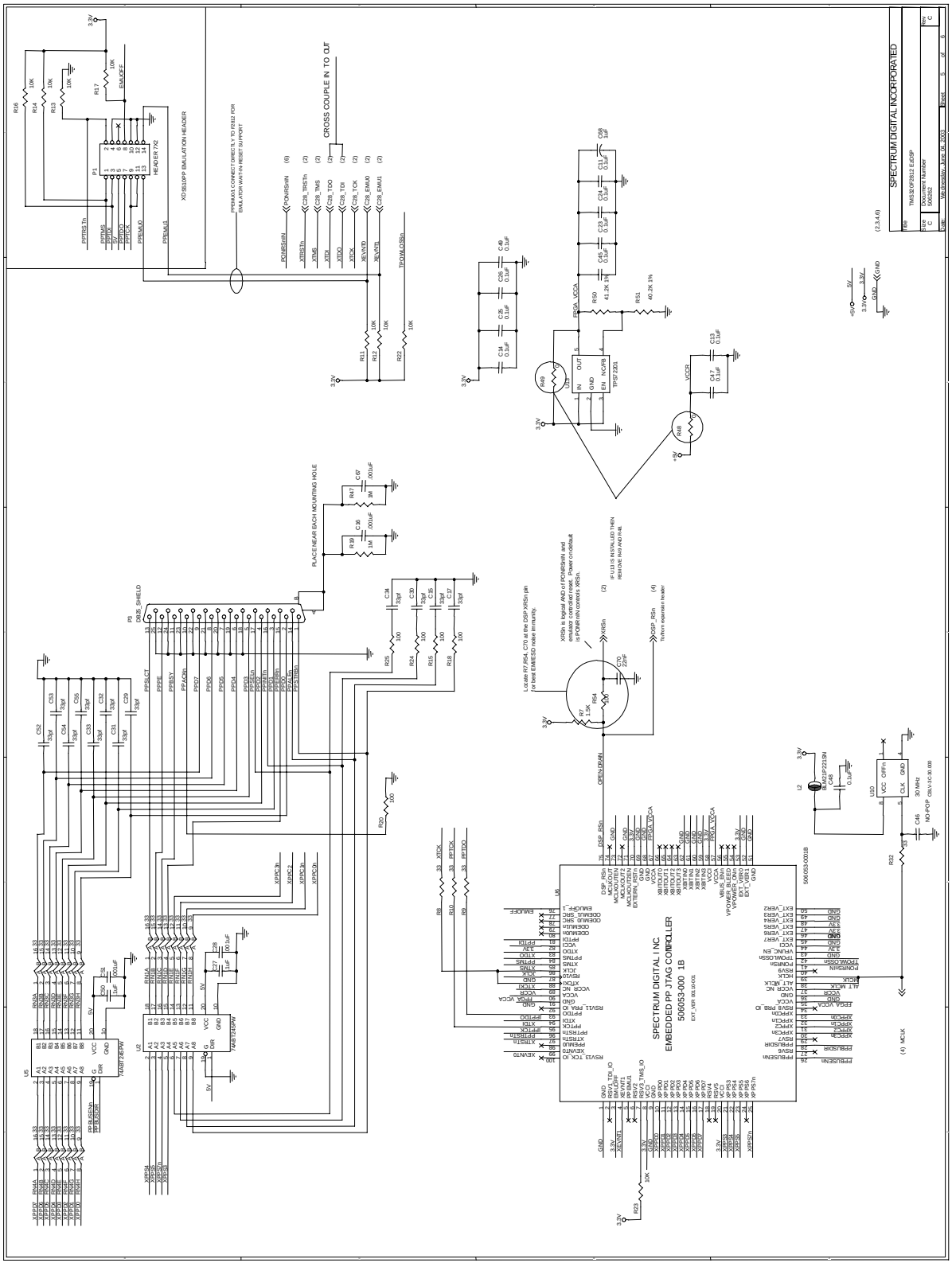
Size: B | Document Number: 506282 | Rev: C

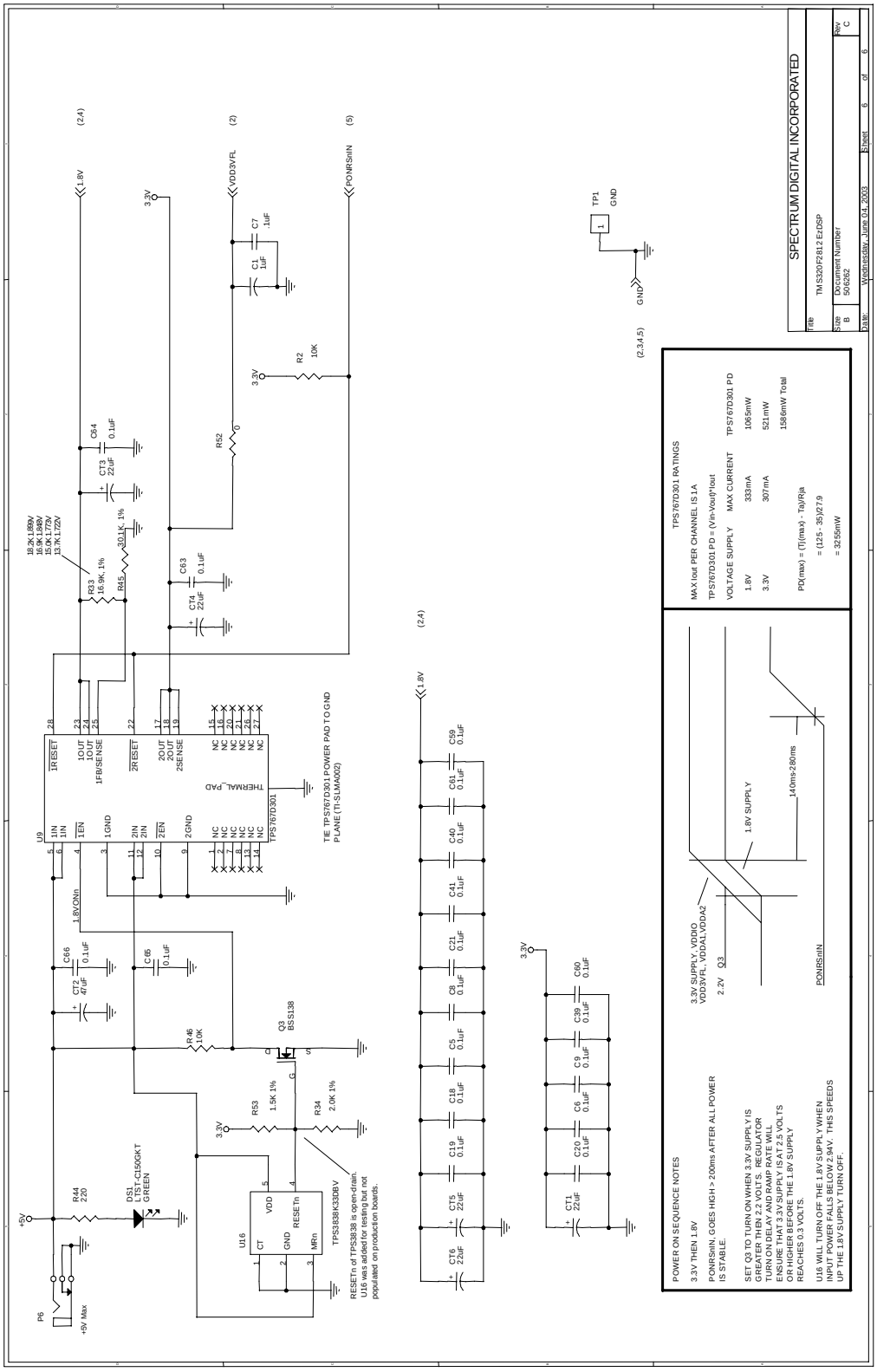
Date: Wednesday, June 04, 2003 | Sheet 1 of 6


DSP
SPECTRUM DIGITAL INCORPORATED
TMS320F2812 EzDSP
506262
Wednesday, June 04, 2003
TMS320F2812PGF


SPECTRUM DIGITAL INCORPORATED
Title
TMS320F2812 EzDSP
Size
B
Document Number
5062262
Rev
B
Date:
Wednesday, June 04, 2003
Sheet
3
of
6
ASRAM
U4
IS61LV6416-12T
C10
0.1uF
C22
0.1uF
3.3V
64Kx16 and 256Kx16
Compatible
(2,4,5,6) GND


TMS320F2812 supports 3.3V input/output levels which are NOT 5V tolerant. Connecting the eZdsp to a system with 5V input/output levels will damage the TMS320F2812. If the eZdsp is connected to another target then the eZdsp must be powered up first and powered down last to prevent latchup condi tion s.
Make a solder connection on JP4 and/or JP5 to the appropriate power supply.
Connect VREFLO to AGND or to VREFLO of target system for proper ADC operation.
Schematic change from version A/B to C. F2812 clkin is 1.8V input.
Tie off unused F24xx compatible signals.
OPTIONAL
BOOT MODE
Title TMS320F2812 EzDSP
Size B
Document Number 506262
Rev C
Date: Wednesday, June 04, 2003
Sheet 4 of 6

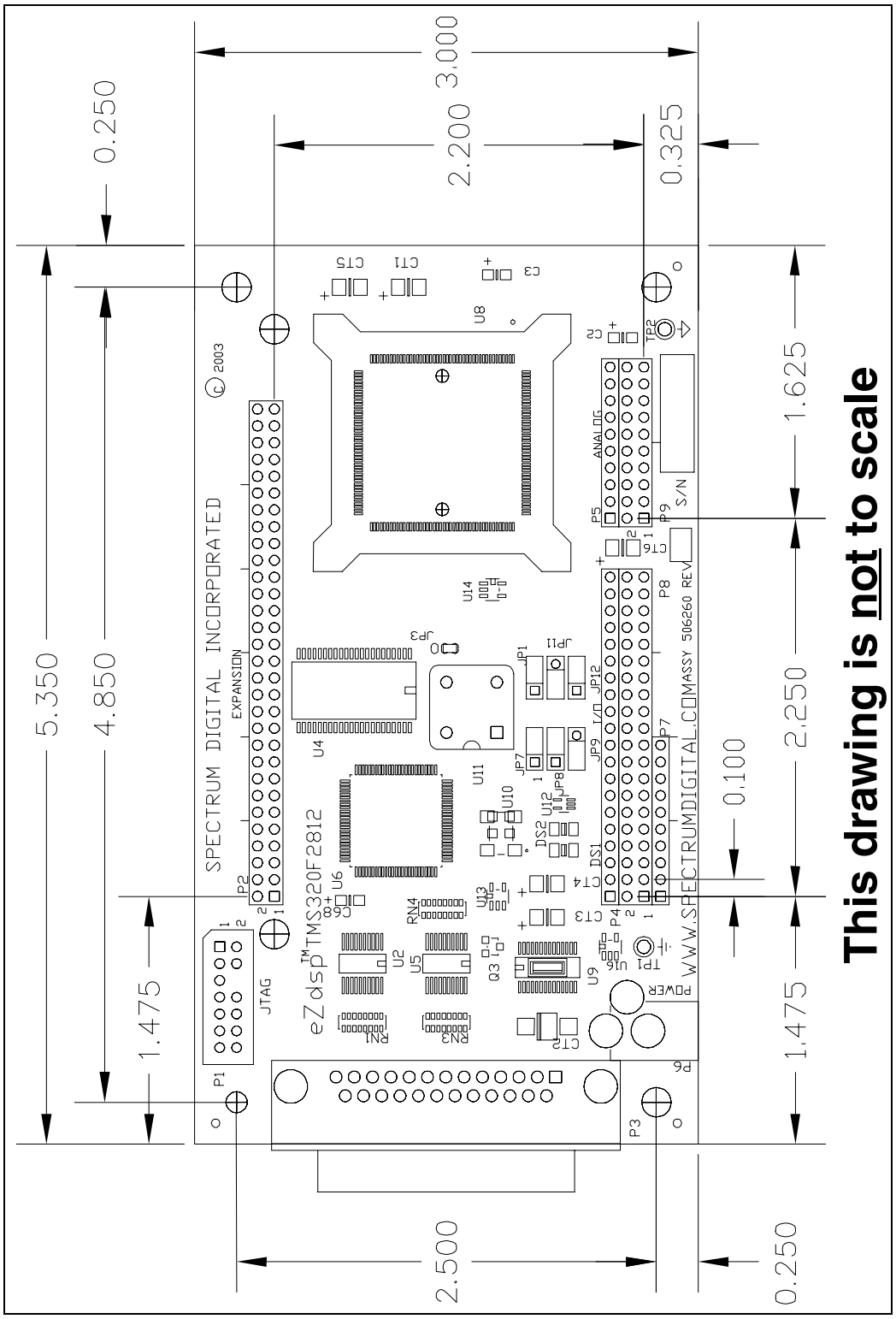


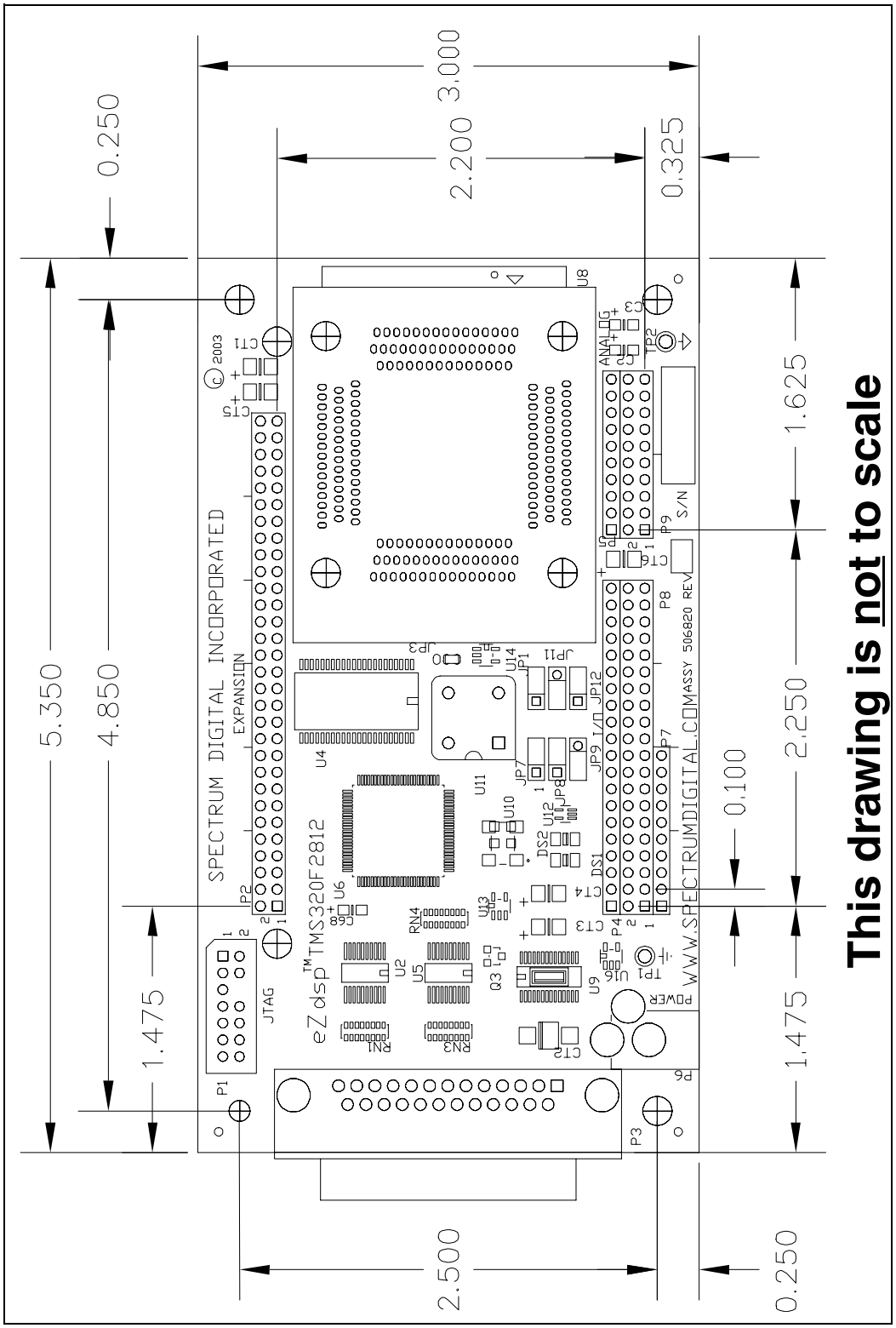

SPECTRUM DIGITAL INCORPORATED
TMS320F2812 ezDSP
Document Number
506202
Wednesday, June 04, 2003
Sheet 6 of 6
Rev C
TPS767D301 RATINGS
MAX Iout PER CHANNEL IS 1A
TPS767D301 PD = (Vin-Vout)*Iout
VOLTAGE SUPPLY MAX CURRENT TPS767D301 PD
1.8V 333mA 1065mW
3.3V 307mA 521mW
1586mW Total
PD(max) = (Tj(max) - Ta)/Rja
= (125 - 35)/27.9
= 3255mW
POWER ON SEQUENCE NOTES
3.3V THEN 1.8V
PONRSnIN, GOES HIGH > 200ms AFTER ALL POWER IS STABLE.
SET Q3 TO TURN ON WHEN 3.3V SUPPLY IS GREATER THEN 2.2 VOLTS. REGULATOR TURN ON DELAY AND RAMP RATE WILL ENSURE THAT 3.3V SUPPLY IS AT 2.5 VOLTS OR HIGHER BEFORE THE 1.8V SUPPLY REACHES 0.3 VOLTS.
U16 WILL TURN OFF THE 1.8V SUPPLY WHEN INPUT POWER FALLS BELOW 2.94V. THIS SPEEDS UP THE 1.8V SUPPLY TURN OFF.
3.3V SUPPLY, VDDIO
VDD3VFL, VDDA1,VDDA2
1.8V SUPPLY
140ms<280ms
PONRSnIN
RESETn of TPS838 is open-drain.
U16 was added for testing but not populated on production boards.
TIE TPS767D301 POWER PAD TO GND PLANE (TI-SLMA002)
TPS838K3308V
TPS767D301

Appendix B

eZdsp™ F2812 Mechanical Information

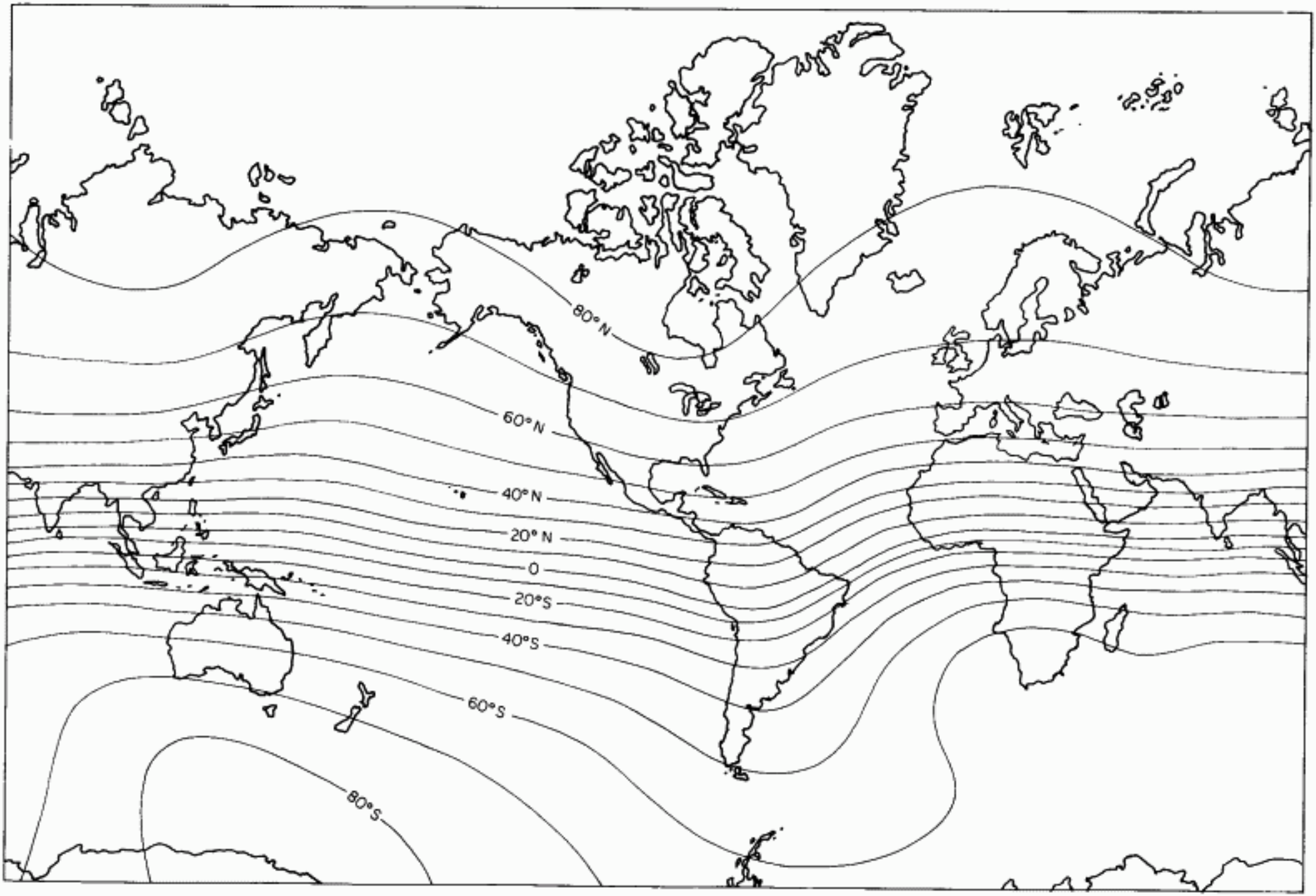
This appendix contains the mechanical information about the socketed and unsocketed versions of the eZdsp™ F2812


3.000
2.200
0.250
0.325
5.350
4.850
1.475
1.625
2.250
0.100
1.475
2.500
0.250
This drawing is not to scale
SPECTRUM DIGITAL INCORPORATED
EXPANSION
JTAG
ANALOG
S/N
POWER
eZdsp™TMS320F2812
WWW.SPECTRUMDIGITAL.COMASSY 506260 REV


This drawing is not to scale
3.000
2.200
0.250
0.325
5.350
4.850
1.475
1.625
2.250
0.100
1.475
2.500
0.250
SPECTRUM DIGITAL INCORPORATED
EXPANSION
JTAG
eZdsp™TMS320F2812
WWW.SPECTRUMDIGITAL.COMASSY 506820 REV
ANALOG
S/N
POWER
© 2003

[redacted]

Printed in U.S.A., September 2003
506265-0001 Rev. F



Main magnetic field inclination (solid contours) on the surface of the earth expressed in degree units. (Modified from "Magnetic Inclination or Dip," Epoch 1975.0, chart published by Defense Mapping Agency Hydrographic Center, Washington, D.C.)

Features

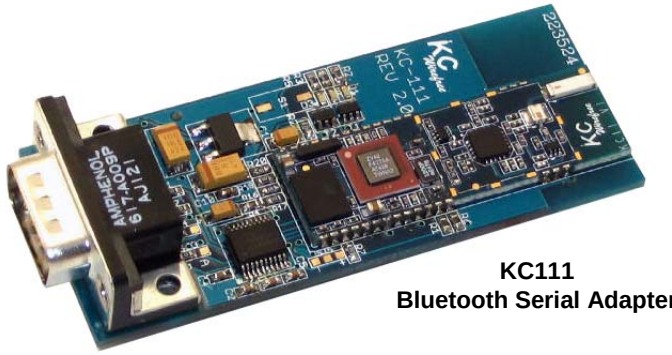
- ***High-performance connections up to 200m***
- ***High-security 128-bit encryption***
- ***High-speed data rates up to 921K baud***
- ***Point-to-point & multi-point connections***
- ***Automatic, self-connecting paired adapters***
- ***Battery & AC power supply options***
- ***Supports SPP - Bluetooth Serial Port Profile***
- ***Supports advanced serial port configurations***
- ***KC Wirefree KC11 Bluetooth OEM Module***
- ***Class 1 radio***

Specifications

- Bluetooth: Version 1.2
- Frequency: 2.4 GHz
- Bandwidth: 1 MHz, 79 Channels
- Frequency Hopping: 1600 hops/sec
- Typical RF Power: +18dB
- Operating Temperature: -20° to +85° C
- Input Voltage: 4 to 10 VDC
- Dimensions: 32mm width, 86mm length
- Power Consumption: typical 50 mA, max 200 mA
- Chipset: Zeevo ZV4002

Providing instant, secure wirefree RS-232 serial connections using Bluetooth® wireless technology.

The KC111 is a complete serial cable replacement solution. No software or computer required.



KC111
Bluetooth Serial Adapter

Bluetooth®

Our KC111 Bluetooth Serial Adapters are a complete wireless solution. Benefits for industrial and office equipment include immediate installation that eliminates the need to dig or route wired serial cables, and allows mobile, portable, or remote communications and control. KC Wirefree Serial Adapters are one of the most versatile solutions for serial data transmissions and control. The 128-bit encryption provides complete security for your wireless data.

Our completely integrated design of an embedded Bluetooth Serial Port Protocol (SPP) allows our serial adapters to work with industrial and other non-computer equipment, which eliminates the need to install any software or device drivers.

KC Wirefree
Bluetooth Serial Adapter
User Guide

Version 2.1

1 Feb. 2005

TABLE OF CONTENTS

1 Overview and Setup

KC Wirefree Serial Adapters are delivered complete with everything the user needs to operate Bluetooth devices. The Serial Cable replacement application embeds a complete upper layer stack, lower layer stack, and Bluetooth profiles.

Because of this high-level of integration, developers can easily use basic serial communication programs to develop host applications/products that interface through a UART port.

The KC Wirefree serial adaptor evaluation kit contains:

- 2 Bluetooth serial adaptors
- 2 Serial cables, Null Modem, DB9 female to female
- 2 AC power adaptors, 6VDC

2 Serial Hardware Types

KC Wirefree Serial Adapters come with two different serial connector types.

2.1 DTE Serial Connector

Adapters that are configured as DTE devices are equipped with 9 pin male gender connectors. In order to connect this type of adapter to a PC, a Null Modem or "twisted" female to female cable is necessary.

2.2 DCE Serial Connector

Adapters that are configured as DCE devices are equipped with 9 pin female gender connectors. This type of adapter is ready to plug into and communicate directly with a PC without additional cables.

3 Launch HyperTerminal

Once the hardware has been setup, the communication program is used to communicate between the two Bluetooth devices. This section describes the launching and configuration of this communication program.

Windows-based computers come with a program called HyperTerminal. Launch and setup this program by following these steps:

- Go to **Start**,
- Select **Programs**,
- Select Accessories,
- Select Communications (Or HyperTerminal in some systems),
- Select Hyper Terminal.

A HyperTerminal window appears (see Figure 1) with a dialogue box. Enter a name (e.g., **test**) and click on **OK**.

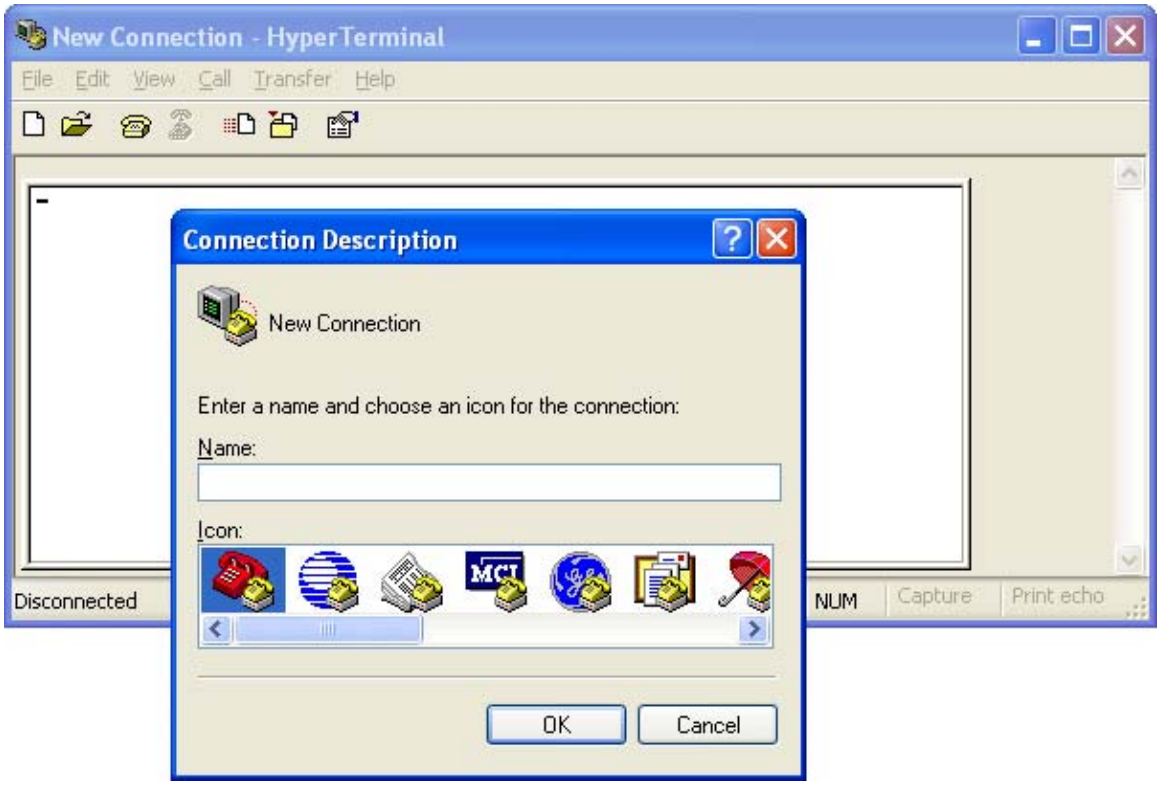


Figure 1. First HyperTerminal Window and Dialogue Box

3.1 Choose Communication Port(s)

The Connect To dialogue box (see Figure 2) appears. Choose the appropriate communication port for each computer and click on **OK**. The "appropriate communication port" is the port to which the serial cable is connected.

Figure 2. Choose Appropriate Com Port

3.2 Set Communication Properties

The Com Port dialogue box (see Figure 3) appears. Choose the appropriate settings for each computer and click on **OK**.

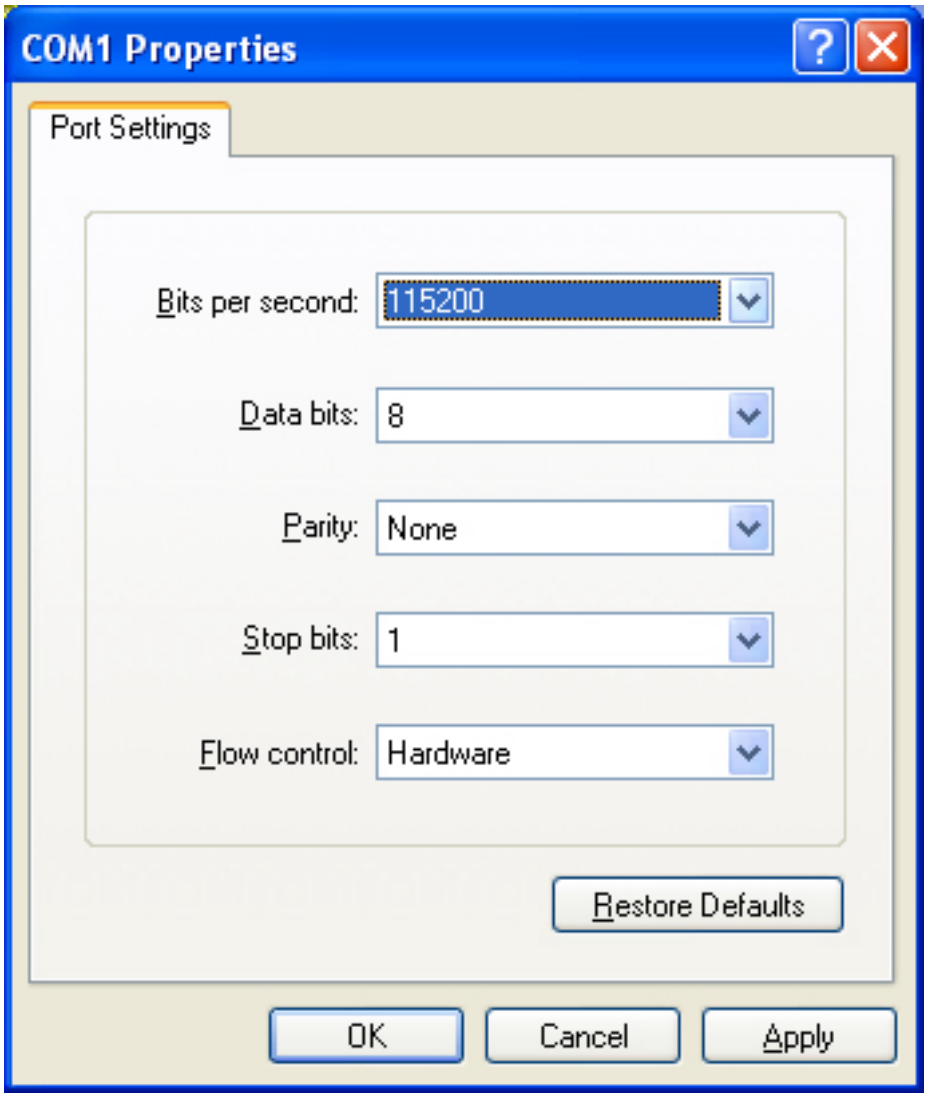


Figure 3. Set Communication Port Settings

The default serial port setup for KC Wirefree adapters is 115K bps, 1 stop bit, no party, 8 data bits, and HW flow control enabled.

4 Establish a Serial Connection

After launching HyperTerminal, a window representing your connection appears. Click on the connect icon to establish a connection from the computer's communication port to the evaluation/development board.

After a connection is established from the computer's communication port to the evaluation/development board, you will see the responses, "AT-ZV -CommandMode-", followed by "AT-ZV BDAddress [bd address of local device]" appear in the text body of the window. Also, on the bottom status bar of the window, the updated status is displayed. See Figure 4.

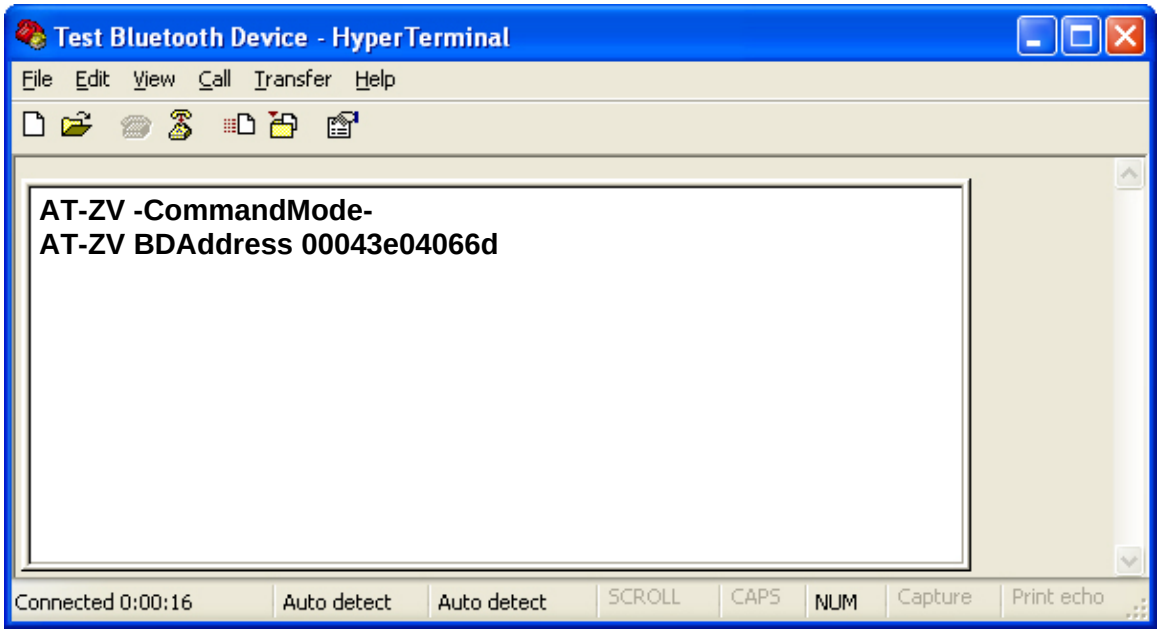


Figure 4. Updated Status Depicts Connection to Evaluation/Development Boards

5 Write Scripts for HyperTerminal

In order to facilitate using HyperTerminal with KC Bluetooth adaptors, the use of pre-written test scripts is recommended. Since the firmware is not able to accept typed command corrections, any errors will require the command to be completely re-entered. To solve this inconvenience, follow the procedure detailed in this section to create HyperTerminal text scripts for AT type commands:

- Right click on the **Desktop**. A drop down menu box appears.
- Select **New** then select **Text Document**.
- Name the document "Connect_Command.txt" or choose an acceptable name.
- Open the document by double clicking on it. Type the following command with the desired remote BD address in place of `[BD_Addr]`:

```
AT+ZV SPPConnect [BD_Addr]
```

- Press **Return/Enter** at the end of the above text. If you forget to include the Return/Enter, the script will not work.

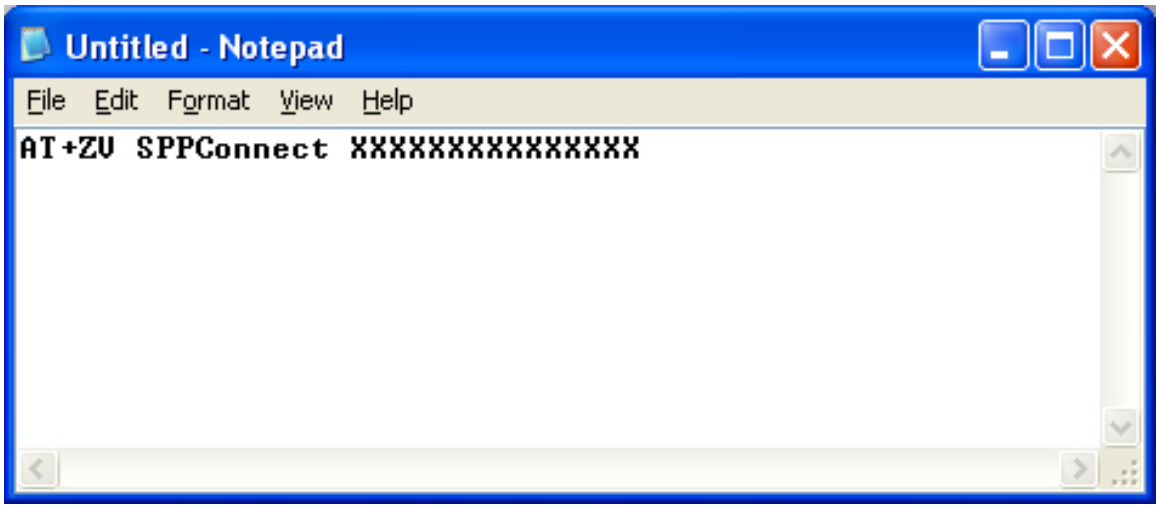


Figure 5. SPP Connect Command with Return Character

- Save the script onto the PC. It will probably be used often.
- Creating scripts for the Change Baud Rate commands is also recommended:

```
AT+ZV ChangeBaud 115200
AT+ZV ChangeBaud 230400
AT+ZV ChangeBaud 460800
AT+ZV ChangeBaud 921600
```

6 Invoke the Connect Command

This section describes how to use the SPP Connect command and demonstrates a file transfer over a Bluetooth link.

Follow the steps below to invoke a command and perform a file transfer:

- On the sending computer's HyperTerminal main menu, click on **Transfer** and then select **Send Text File**. See Figure 6.
- After the browse window appears, locate the previously saved Connect Command script you wish to use. Figure 9.
- Click on **Open**. A connection, via Bluetooth link, will be established to the receiving terminal. See Figure 9.
- On the transmitting terminal, select the **Transfer** menu and then select **Send File**. A Send File window appears. See Figure 8.
- Click on **Browse** to select a file to send or enter the path name of the file to send.
- Choose **Zmodem** from the list of protocols. Click on **Send**. See Figure 8.

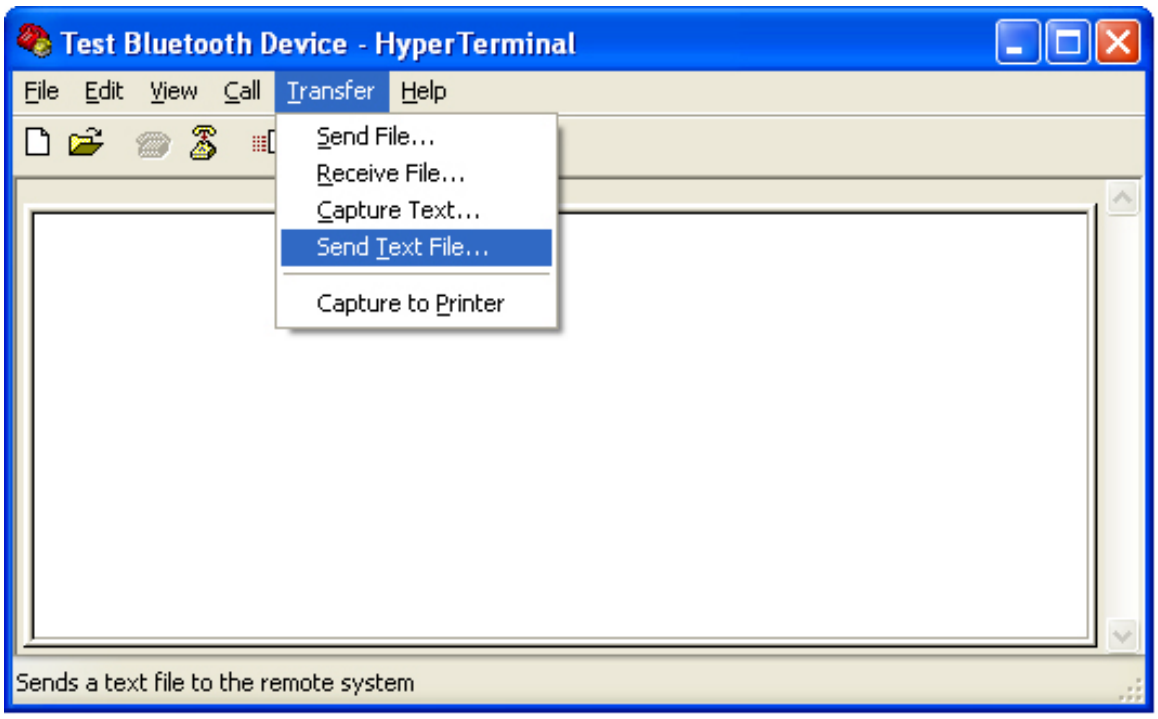


Figure 6. Transfer > Send Text File

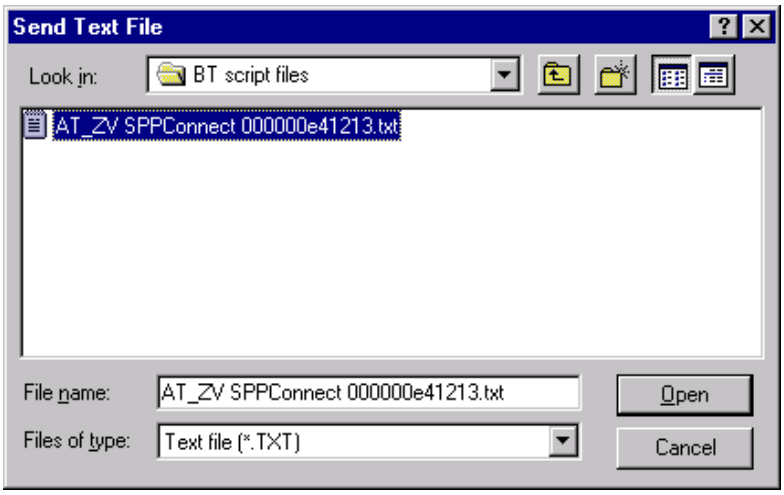


Figure 7. Example of a Script

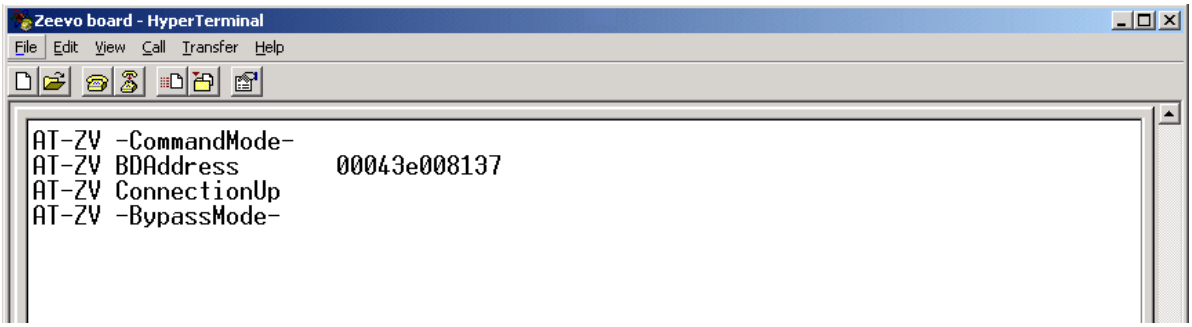


Figure 8. The Connect Event Appears

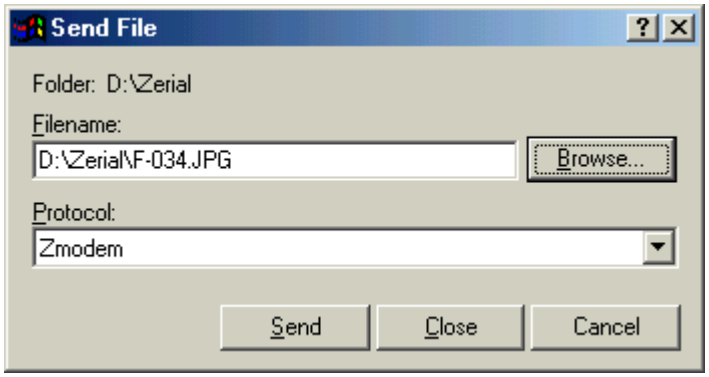


Figure 9. Send File Window

- On the receiving terminal, select the **Transfer** menu, and then select **Receive File**. A dialogue window appears.
- Click on the **Browse** button and choose the directory location of the file transfer.

- Choose **Zmodem** from the list of protocols. Click on **Receive**. See the figure below.

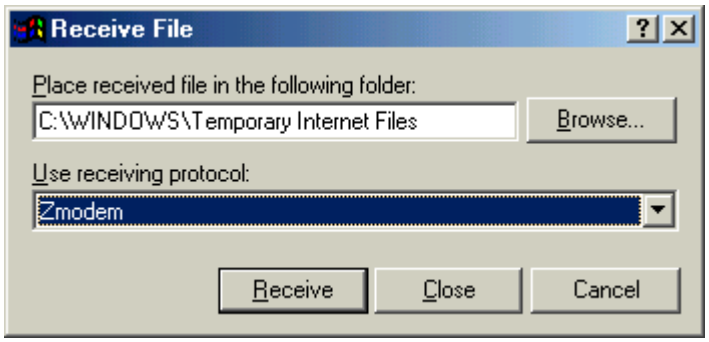


Figure 10. Receive File Dialogue Window

A status window will appear on each terminal. See the two figures below.

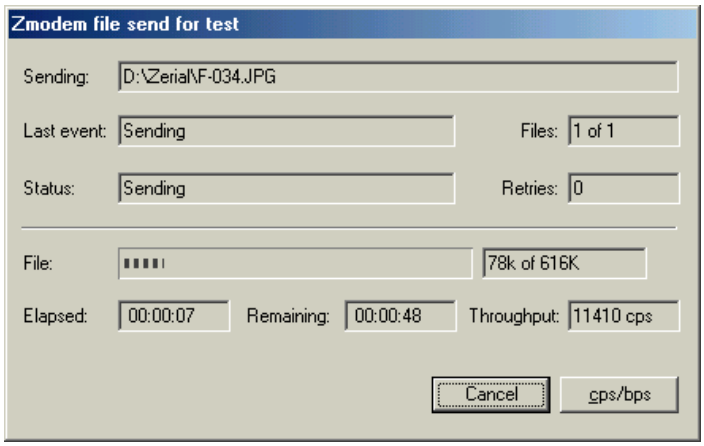


Figure 11. Dialogue Window on Sending Computer

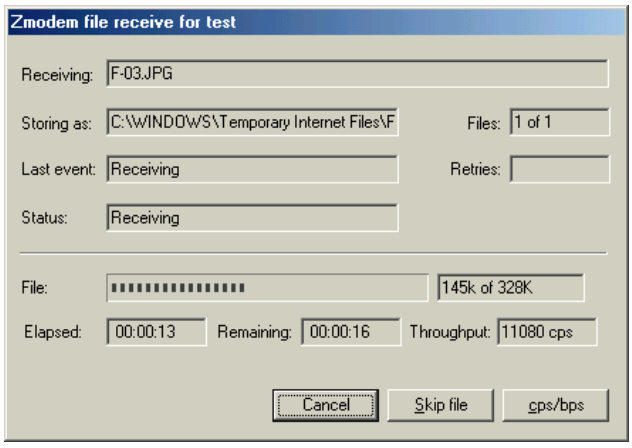


Figure 12. Dialogue Window on Sending Computer

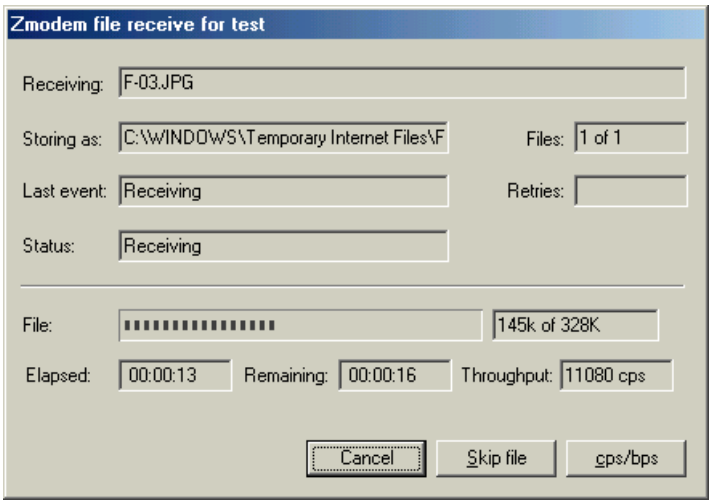


Figure 13. Dialogue Window on Receiving Computer

7 Changing the Baud Rate

Although the adaptors initially have a default baud rate of 115200 bps, multiple baud rates are supported. To change the baud rate for the current session, use the following command:

```
AT+ZV ChangeBaud [new baud rate]
```

Figure 14. Enter the Connect Command

Hit **Return/Enter**.

It is not necessary to reset. If the adaptor is reset, the baud rate and other session configurations will be reset to default.

To permanently change the default baud rate, use the following command:

```
AT+ZV ChangeDefaultBaud [new baud rate]
```

In this case, the new default baud rate will be applied after every reset, but not for the current session.

After the baud rate has been changed, the terminal's baud rate must also be changed to match.

- On HyperTerminal's main menu, select **File** and then select **New Connection**.
- A HyperTerminal window appears with a dialogue box. Enter a name (e.g., **test**) and click on **OK**.
- The Connect To dialogue box appears. Choose the appropriate communication port and click on **OK**.
- The Com Port dialogue box appears. Choose the new baud rate for each terminal and click on **OK**.

7.1 High Speed Serial Cards

In order to increase the UART speed of the PC to values above 115K bps, it is often necessary to use a third-party high speed serial card.

A company called Axxon has a 2-port adapter called the “Soft I/O” Axxon’s contact information is http://www.softio.com/.

8 Streaming Serial Mode

The default UART setting for hardware flow control, CTS/RTS, is enabled. If this setting is not desired, a feature called streaming serial mode may be enable using the Dip Switch block on the back of the KC serial adaptor.

Streaming serial mode activates the following changes:

- The CTS/RTS flow control lines are ignored by the UART
- Data may be discarded when the Bluetooth link is unable to transmit due to poor RF conditions such as out of range.
- The maximum baud rate is limited to 115K bps.

To Enable Streaming Serial mode: place dip switch 1 in the ON position and reset the power to the device.

To Disable Streaming Serial mode: place dip switch 1 in the OFF position and reset the power to the device.

Note: Switches 2 to 4 in the dip switch block are not used.

kcSerial Reference Guide

01 March 2006

TABLE OF CONTENTS

1 Preface

The document describes an embedded application that provides a kcSerial cable replacement service using the Bluetooth Serial Port Profile. This software was originally developed by Zeevo, Inc. under the name Zerial.

1.1 Purpose

This document provides a detailed description of each command supported by the kcSerial interface. Each description explains parameters and the expected behaviors of each command and response.

Errors responses are also detailed in this document.

1.2 Definitions and Acronyms

Table 1. Definitions and Acronyms

Term	Description/Meaning
ASCII	American Standard Code for Information Interchange, a standard describing encoding of characters; the use in this document is strictly US 7-bit
BD	Bluetooth Device
DCD	Modem signal “data carrier detect”; indication from a modem that a connection has been made through, for example, a dialup connection
DTE	Data terminal entity, e.g., a computer
DTR	Modem signal “data terminal ready”; indication to a modem that the data terminal is ready for a connection
DUN	Dialup Networking (Profile)
GPIO	General Purpose Input-Output
LAN	Local Area Network
PIN	Personal Identification Number
SPP	Serial Port Profile
UART	Universal Asynchronous Receiver-Transmitter

1.3 Feedback

We are constantly improving our product and would very much like to get your feedback. Please send your feedback in an email to support@kcwirefree.com.

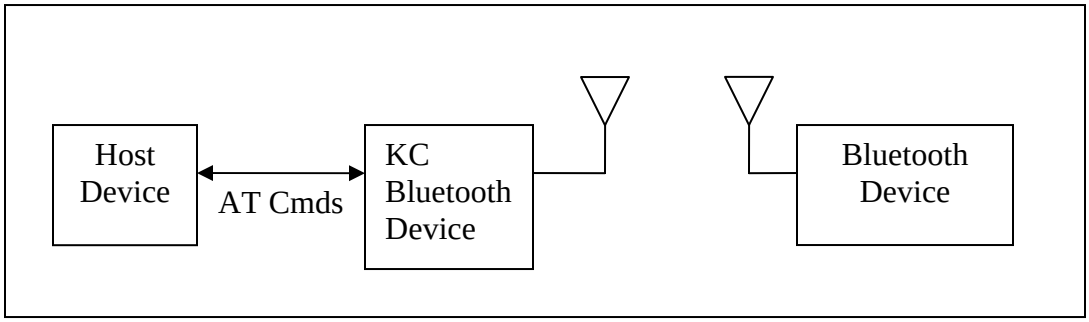
For the latest updates and additional information please visit the KC Wirefree website at: www.kcwirefree.com

2 Overview

This chapter gives a basic overview of the kcSerial interface. For further information, please refer to the *kcSerial User Guide*.

2.1 kcSerial Interface Overview

kcSerial is a cable replacement application that provides point-to-point communication between two Bluetooth devices. A serial port is used to communicate with a host device through an AT command interface as shown below.



kcSerial provides the following basic features:

- Point-to-point connection – kcSerial only supports a connection with one device at a time.
- Serial Port Profile – SPP is supported with kcSerial for both Client and Server application.
- Dial Up Networking – DUN profile support for Client applications (DUN Server is not currently supported).
- Command and Bypass modes – it is possible to switch between Command and Bypass (data transmit/receive) modes during an active connection
- Security – Bonding and data encryption provides a secure link between two devices.
- Multiple Device Bonding – special security keys can be exchanged with multiple devices to allow different devices to securely connect with kcSerial (although not simultaneously)
- Power conservation – use of Park, Sniff, and Hold features to minimize power consumption
- Variable Baud Rates – the serial port can be configured for the following baud rates: 9600, 19.2K, 38.4K, 57.6K, 115.2K (default), 230.4K, 460.8K, 921.6K

3 Commands

This chapter details the each of the kcSerial AT commands including brief descriptions of behavior, syntax of the command, context of the command, and types of responses.

Table Key:

✓ – command is supported in this release

U – command has been updated for this release, see release notes

X – command not supported in this release

Table 3: kcSerial Command Summary

Command	kcSerial v1.0	kcSerial v2.0	kcSerial v2.1
Bond	✓	✓	✓
Bypass	✓	✓	✓
ChangeBaud	✓	✓	✓
ChangeDefaultBaud	✓	✓	✓
DefaultLocalName	X	✓	✓
DeleteSmartCable	✓	✓	✓
DisableBond	✓	✓	✓
Discovery	✓	✓	✓
DunConnect	✓	✓	✓
DunDisconnect	✓	✓	✓
EnableBond	✓	✓	✓
EraseBondTable	✓	✓	✓
ExitPart	✓	✓	✓
ExitSniff	✓	✓	✓
GPIOConfig	✓	✓	✓
GPIORead	✓	✓	✓
GPIOWrite	✓	✓	✓
Hold	✓	✓	✓
HostEvent	✓	✓	✓
LocalName	✓	✓	✓
Park	✓	✓	✓
RemoteCommand	X	X	✓
RemoteCmdDisconnect	✓	✓	✓

Command	kcSerial v1.0	kcSerial v2.0	kcSerial v2.1
Reset	✓	✓	✓
Security	✓	✓	✓
SmartCableSetup	✓	✓	✓
Sniff	✓	✓	✓
SPPConnect	✓	✓	✓
SPPDisconnect	✓	✓	✓
StreamingSerial	X	X	✓
UpdateInquiryScan	✓	✓	✓
UpdatePageScan	✓	✓	✓
Version	✓	U	U

The following subsections describe each of these commands in detail, including a description of behavior, syntax (including possible parameter values), and types of responses.

Some responses will not be “immediate”. Where applicable, these will be noted and will include an approximate delay before response.

For commands with optional parameters, all possible forms will be listed under the syntax subsection.

Error responses are described in Section 4 Error Responses.

3.1 Bond

The `Bond` command is used to initiate bonding with a specified device. A personal identification number (PIN) is also required with this command.

Syntax

```
AT+ZV Bond [BD addr] [PIN]
```

Where `[BD addr]` is the BD Address of the remote device with which to bond and `[PIN]` is the PIN code to use (up to 16 characters).

Responses

If the request is successfully submitted, the response is:

```
AT-ZV BondPending [BD addr]
```

If the operation is successful, the response is:

```
AT-ZV BondOk
```

If the operation fails, the response is:

```
AT-ZV BondFail
```

3.2 Bypass

The `Bypass` command is used to return the kcSerial interface to the bypass mode, if a connection is still available. The possible use for this is to change a setting after a connection has been made (such as the UART baud rate). If the kcSerial interface does not have a connection, it will respond as if the connection is down.

Syntax

```
AT+ZV Bypass
```

Responses

If a connection is still available, the response is:

```
AT-ZV -BypassMode-
```

If there is currently no connection, the response is:

```
AT-ZV ConnectionDown
```

3.3 ChangeBaud

The host sends the `ChangeBaud` command in order to change the local UART speed to a new speed identified by the host. This setting will only remain in effect during the current session - until reset.

Syntax

```
AT+ZV ChangeBaud [rate]
```

where `[rate]` is the new baud rate:

- 9600
- 19,200
- 38,400
- 57,600
- 115,200
- 230,400
- 460,800
- 921,600

Responses

If the change is accepted, the response is:

```
AT-ZV Baudrate Changed
```

The actual change will not occur until the response has been completely transmitted.

3.4 ChangeDefaultBaud

The host sends the `ChangeDefaultBaud` command in order to change the default UART speed to a new speed identified by the host. This command is used to override the default baud rate from the Dynamic Configuration script so that the device does not require reprogramming to update this setting. The new baud rate is updated permanently until the device is either re-programmed or another `ChangeDefaultBaud` command is issued. The baud rate specified in the command will not take effect until the device is reset. To change the baud rate of the current session, use the `ChangeBaud` command.

Syntax

```
AT+ZV ChangeDefaultBaud [rate]
```

where `[rate]` is the new baud rate:

- 9,600
- 19,200
- 38,400
- 57,600
- 115,200
- 230,400
- 460,800
- 921,600

Responses

If the change is accepted, the response is:

```
AT-ZV Baudrate Changed
```

3.5 DefaultLocalName

The `DefaultLocalName` command is used to set the name of the device to the name that is reported during device discoveries. By default, the kcSerial interface uses “KCWirefreeDevice”. Changing the name using this command will permanently change the local name, unlike the `LocalName` command.

Syntax

```
AT+ZV DefaultLocalName [name]
```

Where `[name]` is a string for the new local name (up to 50 characters). The space character is allowed; the name is assumed to be all text up to the end of the command.

Responses

If the operation is successful, the response is:

```
AT-ZV LocalNameOk
```

3.6 DeleteSmartCable

The `DeleteSmartCable` command removes the current Smart Cable settings that were entered using the SmartCableSetup command, but not the setting from the dynamic configuration. The Smart Cable will then be deactivated for the remainder of this session. Upon reset, if a dynamic configuration for a Smart Cable exists, it will be activated. If there is no dynamic configuration Smart Cable setup, then this feature will remain deactivated.

Syntax

```
AT+ZV DeleteSmartCable
```

Responses

If the operation is successful, the response is:

```
AT-ZV DeleteSmartCableDone
```

3.7 DisableBond

The `DisableBond` command is used to disallow new bonding with a device.

This command cannot be used while a connection is active.

Syntax

```
AT+ZV DisableBond
```

Responses

If the operation is successful, the response is:

```
AT-ZV BondDisabled
```

3.8 Discovery

The `Discovery` command is used to initiate a device discovery. The command will return the number of responses of nearby devices and then the individual responses with BD address, name of device, and service names - optional. The number of devices returned is limited to 10; and the number of services per device is limited to 8. Inquiry is performed with an interval of 10.24 seconds. The devices are reported in the same order as the original inquiry results.

Syntax

```
AT+ZV Discovery
```

```
AT+ZV Discovery [CoD]
```

```
AT+ZV Discovery [CoD] [profile] [include service
enable/disable]
```

where devices are filtered on the `[CoD]` class:

- All (default)
- Misc
- Computer

- Phone
- LAN
- Peripheral
- Imaging
- Unclass

where the service name is requested for the profile named `[profile]:`

- All (default)
- SPP
- DUN
- LAP
- FAX

where `[include service enable/disable]: true` returns both the remote services and names, and `false` skips the remote service discovery and only returns the remote names – this completes the discovery process faster.

Responses

When the discovery command has been accepted, the response is:

```
AT-ZV InqPending
```

Once the initial inquiry is complete and discovery has been started, the response is:

```
AT-ZV DiscoveryPending [num]
```

where `[num]` is the number of devices found, in decimal (up to 10 will be reported).

For each name or service name request that is successful, the response uses the returned names in the following format.

```
AT-ZV Device [BD addr] [name] [service name] [...]
```

where `[BD addr]` is in hexadecimal with the most significant byte first. `[name]` is a string in double quotes " ". `[service name]` is a string without quotes and is reported for each service reported.

For each unsuccessful name request, the corresponding name is replaced by `" Unknown"` . The name request may not be successful if unable to make a connection for the request.

```
AT-ZV Device [BD addr] " Unknown"  [service name] [...]
```

For each service request that does not return services, one name is returned as "`NoSvcs"`. A request may not return services if unable to connect to the device for the request or if the device does not contain the requested service.

```
AT-ZV Device [BD addr] [name] NoSvcs
```

For each unsuccessful service request, one name is returned as "`SvFail"`. This will only occur due to an internal error.

```
AT-ZV Device [BD addr] [name] SvFail
```

If the memory allocated for the service discovery buffers is insufficient, then then result may return an "Inval" response. If this occurs, please contact support@kcwirefree.com for assistance.

```
AT-ZV Device [BD addr] [name] [service name 1] [Inval]
[Inval]
```

3.9 DUNConnect

The DUNConnect command is used to initiate a connection with the specified device. The remote BD address must be specified. The remote Service is optional. If not specified, the first registered DUN service will be used by default.

Syntax

```
AT+ZV DUNConnect [BD Addr] [Service]
```

Where [BD Addr] is the remote devices BD Address to page. [Service] is the specific service on the remote device; optional.

Responses

If the connection is successful, the response is:

```
AT-ZV ConnectionUp
AT-ZV -BypassMode-
```

If the connection cannot be completed, the response is:

```
AT-ZV DUNConnectionClosed
```

3.10 DUNDisconnect

The DUNDisconnect command is used to terminate a connection with the remote device.

Syntax

```
AT+ZV DUNDisconnect
```

Responses

If the connection is successful, the response is:

```
AT-ZV DUNConnectionClosed
```

3.11 EnableBond

The EnableBond command is used to enable bonding with another device. The BD Address, PIN and timeout parameters are optional.

When no BD Address is specified, requests from all BD Addresses are allowed.

If a BD Address is specified, bonding requests from devices with BD Addresses other than the one specified will fail and the existing link key will be deleted for that device.

Optionally, a PIN code may be entered with this command. If no PIN code is specified, the default PIN code will be used. The default PIN code is either the last 4 digits of the device's BD address or the dynamically configured PIN code, depending on the default PIN selection in the dynamic configuration file.

Also, a timeout value, in seconds, may be entered after the PIN code. Bonding will be disabled automatically after the requested timeout. If no timeout is specified, bonding is enabled until reset or until the `DisableBond` command is used.

If this command is issued multiple times, only the last PIN and BD address are saved. Also, if this command is issued before the first timeout occurs, the subsequent command will extend the timeout. The timeout is always set to the specified time beyond the last received `EnableBond`.

Syntax

```
AT+ZV EnableBond
AT+ZV EnableBond [BD addr]
AT+ZV EnableBond [BD addr] [PIN]
AT+ZV EnableBond [BD addr] [PIN] [timeout]
```

Where `[BD addr]` is the BD Address of the remote device with which to bond, `[PIN]` is the PIN code to use (up to 16 characters), and `[timeout]` is the duration of the timeout in seconds (1 to 14,400, in decimal).

Responses

If the operation is successful, the response is:

```
AT-ZV BondEnabled
```

If bonding has been initiated by a remote device, the notification is:

```
AT-ZV BondPending [BD addr]
```

where `[BD addr]` is the BD address of the remote device that initiated the bonding.

If bonding has occurred, the notification is:

```
AT-ZV BondOk [BD addr]
```

where `[BD addr]` is the BD address of the remote device with successful bonding.

If bonding was initiated by a remote device but failed, the notification is

```
AT-ZV BondFail
```

When the time limit for bonding has expired, the notification is

```
AT-ZV BondDisabled
```

3.12 EraseBondTable

The `EraseBondTable` command is used to erase all of the bonded device entries. Single devices cannot be erased with this command

Syntax

```
AT+ZV EraseBondTable
```

Responses

If the operation is successful, the response is:

```
AT-ZV BondTableErased
```

3.13 ExitPark

The `ExitPark` command is used to switch a device from park mode to active mode.

Syntax

```
AT+ZV ExitPark [BD address]
```

Where `[BD address]` is the BD address of the device to be switched to active mode.

Responses

If the operation is successful, the response is:

```
AT-ZV ActiveMode
```

3.14 ExitSniff

The `ExitSniff` command is used to switch a device from sniff mode to active mode.

Syntax

```
AT+ZV ExitSniff [BD address]
```

Where `[BD address]` is the BD address of the device to be switched to active mode.

Responses

If the operation is successful, the response is:

```
AT-ZV ActiveMode
```

3.15 GPIOConfig

The `GPIOConfig` command is used to configure a GPIO pin to input or output.

Syntax

```
AT+ZV GPIOConfig [GPIO Pin] [Configuration]
```

Where `[GPIO Pin]` is the Pin number, 0 – 15, of the desired GPIO to configure. `[Configuration]` is “i” or “I” for input and “o” or “O” for output.

Responses

If the operation is successful, the response is:

```
AT-ZV GPIOConfigDone
```

3.16 GPIORead

The `GPIORead` command is used to read a GPIO pin. A GPIO may be read while configured as either an input or output.

Syntax

```
AT+ZV GPIORead [GPIO Pin]
```

Where `[GPIO Pin]` is the Pin number, 0 – 15, of the desired GPIO to read.

Responses

If the operation is successful, the response is:

```
AT-ZV GPIOReadDone [result]
```

Where `[result]` is either a 1 to indicate high, or 0 to indicate low.

3.17 GPIOWrite

The `GPIOWrite` command is used to set a GPIO pin to high or low. A GPIO may only be set when configured as an output.

Syntax

```
AT+ZV GPIOWrite [GPIO Pin] [Setting]
```

Where `[GPIO Pin]` is the Pin number, 0 – 15, of the desired GPIO to read. `[Setting]` is a 1 to set a pin to high and a 0 to set a pin to low.

Responses

If the operation is successful, the response is:

```
AT-ZV GPIOWriteDone
```

3.18 Hold

The `Hold` command is used to switch a device from active mode to hold mode.

Syntax

```
AT+ZV Hold [BD address] [Hold Duration]
```

Where `[BD address]` is the BD address of the device to be switched to active mode. `[Hold Duration]` is given in slots from 4-10,000.

Responses

If the operation is successful, the response is:

```
AT-ZV HoldMode
```

3.19 HostEvent

The `HostEvent` command is used to enable/disable the host notification strings. This will override the default setting in the dynamic configuration only for the current session; until reset.

Syntax

```
AT+ZV HostEvent [Enable/Disable]
```

Where `[Enable/Disable]` is an “e” or “E” character to enable this parameter and a “d” or “D” character to disable it.

Responses

If the feature is successfully enabled, the response is:

```
AT-ZV HostEvent Enabled
```

If the feature is successfully disabled there is no response because the events have been disabled.

3.20 LocalName

The `LocalName` command is used to set the name of the device to the name that is reported during device discoveries. By default, the kcSerial interface uses “KCWirefreeDevice”. Changing the name using this command does not permanently change the local name.

Syntax

```
AT+ZV LocalName [name]
```

Where `[name]` is a string for the new local name (up to 50 characters). The space character is allowed; the name is assumed to be all text up to the end of the command.

Responses

If the operation is successful, the response is:

```
AT-ZV LocalNameOk
```

3.21 Park

The `Park` command is used to switch a device from active mode to park mode.

Syntax

```
AT+ZV Park [BD address] [Beacon Period]
```

Where `[BD address]` is the BD address of the device to be switched to active mode. `[Beacon Period]` is given in slots from 200-10,000.

Responses

If the operation is successful, the response is:

```
AT-ZV ParkMode
```

3.22 RemoteCommand

The `RemoteCommand` command is used to enable/disable the remote command mode. This setting is stored in persistent memory, and will be retained after each reset. Additionally, the new setting will take effect upon the next device reset.

Syntax

```
AT+ZV RemoteCommand [Enable/Disable]
```

Where `[Enable/Disable]` is an “e” or “E” character to enable this parameter and a “d” or “D” character to disable it.

Responses

If the feature is successfully applied, the response is:

```
AT-ZV RemoteCommand [Enabled/Disabled]
```

3.23 RemoteCmdDisconnect

The `RemoteCmdDisconnect` command is used to disconnect a remote command connection. This command only applies to the server side of the link; the remote device. The client or local device, if using the kcSerial interface, should use a SPPDisconnect command since it is not in Remote Command mode.

Syntax

```
AT+ZV RemoteCmdDisconnect
```

Responses

If the operation is successful, the response is:

```
AT-ZV RemoteCmdModeClosed
```

3.24 Reset

The `Reset` command is used to reset the kcSerial interface. This is provided in the event that a host application wants to perform a software reset for error recovery. There is a response prior to reset to verify the command was received by the kcSerial interface.

Syntax

```
AT+ZV Reset
```

Responses

If the operation is successful, the response is:

```
AT-ZV ResetPending
```

3.25 Security

The `Security` command is used to set the security level of the device in use. By default, security level none is used.

- Variable pin type (the pincode request event will always be received by the application from the stack), and
- 128-bit unit key.

Service level security, level 2, is not currently supported. The security setting is not preserved in non-volatile memory.

Syntax

```
AT+ZV Security [level]
```

where `[level]` is the type of security to use:

- None
- Link (default)

Responses

If the operation is successful, the response is:

```
AT-ZV SecurityOk
```

3.26 SmartCableSetup

The `SmartCableSetup` command is used enable and configure a Smart Cable device. A device's BD Address is specified with which to automatically establish a connection; replacing the need for AT connection commands. This command will override the dynamic configuration of a Smart Cable device until the `DeleteSmartCable` command is issued; it is saved in non-volatile memory.

Syntax

```
AT+ZV SmartCableSetup [BD address] [Attempts] [Interval]
```

Where `[BD address]` is the BD address of the remote device to page and attempt to connect. `[Attempts]` 0 – 999 is the number of pages the will be attempted to the specified device until a connection is successful. A value of 0 will not automatically page the remote device, however, GPIO 7 may be asserted to manually send a page. A value of 1000 will perform unlimited pages until connected. be switched to active mode. `[Sniff Interval]` is given in slots from 66-10,000. [Interval] 1-1000 is the number of 100ms intervals (0.1sec to 100 sec) between page attempts. This interval is in addition to the amount of time required by the page attempt itself.

Responses

If the operation is successful, the response is:

```
AT-ZV SmartCableConfigDone
```

3.27 Sniff

The `Sniff` command is used to switch a device from active mode to sniff mode.

Syntax

```
AT+ZV Sniff [BD address] [Sniff Interval]
```

Where `[BD address]` is the BD address of the device to be switched to active mode. `[Sniff Interval]` is given in slots from 66-10,000.

Responses

If the operation is successful, the response is:

```
AT-ZV SniffMode
```

3.28 SPPConnect

The `SPPConnect` command is used to initiate a connection with the specified device. The remote BD address must be specified. The remote Service is optional. If not specified, the first registered SPP service will be used by default.

Syntax

```
AT+ZV SPPConnect [BD Addr] [Service]
```

Where `[BD Addr]` is the remote devices BD Address to page`. [Service]` is the specific service on the remote device; optional.

Responses

If the connection is successful, the response is:

```
AT-ZV ConnectionUp
AT-ZV -BypassMode-
```

If the connection cannot be completed, the response is:

```
AT-ZV SPPConnectionClosed
```

3.29 SPPDisconnect

The `SPPDisconnect` command is used to terminate a connection with the remote device.

Syntax

```
AT+ZV SPPDisconnect
```

Responses

If the connection is successful, the response is

```
AT-ZV SPPConnectionClosed
```

3.30 StreamingSerial

Syntax

```
AT+ZV StreamingSerial [Enable/Disable]
```

Where `[Enable/Disable]` is an “e” or “E” character to enable this parameter and a “d” or “D” character to disable it.

Query

An alternative syntax may be used to query the current StreamingSerial feature status. This syntax is not supported by other commands.

```
AT+ZV StreamingSerial
```

Responses

If the feature is successfully applied or queried, the response is:

```
AT-ZV StreamingSerial [Enabled/Disabled]
```

3.31 UpdateInquiryScan

The `UpdateInquiryScan` command is used to modify the Inquiry scan parameters: mode, duration, and interval.

Syntax

```
AT+ZV UpdateInquiryScan [mode] [duration] [interval]
```

where `[mode]` is the discoverable mode:

- 0: non-discoverable
- 1: limited discoverability – NOT SUPPORTED
- 2: discoverable

`[duration]`

is the scan length in slots; 18 to 4096. The default duration is 18 slots.

`[interval]`

is the period between scans in slots; 18 to 4096. The default interval is 2048 slots

Responses

If the command is successful, the response is:

```
AT-ZV InquiryScanUpdateDone
```

3.32 UpdatePageScan

The `UpdateInquiryScan` command is used to modify the Page scan parameters: mode, duration, and interval.

Syntax

```
AT+ZV UpdatePageScan [mode] [duration] [interval]
```

Where `[mode]` is the connectable mode:

- 0: non-connectable
- 1: connectable

`[duration]` is the scan length in slots; 18 to 4096. The default duration is 18 slots.

`[interval]` is the period between scans in slots; 18 to 4096. The default interval is 2048 slots

Responses

If the command is successful, the response is:

```
AT-ZV PageScanUpdateDone
```

3.33 Version

The `Version` command is used to return the current version of the kcSerial interface.

Syntax

```
AT+ZV Version
```

Responses

If the operation is successful, the response is:

```
AT-ZV kcSerialVer [x.y]
```

where `[x.y]` is the current version of the kcSerial Interface.

4 Error Responses

This chapter details the error responses that occur under specific circumstances.

There are seven error responses that can occur beyond error responses specific to a particular command (e.g., `Discovery`). They are:

`ErrConnect`
`ErrExecution`
`ErrFormat`
`ErrInProgress`
`ErrInvalidParam`
`ErrNumParam`
`ErrUnknownCmd`

The following subsections detail the different error responses. These error responses replace the original error response

`AT-ZV ErrorCommand`

4.1 ErrConnect

The `ErrConnect` error response will be sent if kcSerial has a valid connection established and the command cannot be executed while connected (even in the command mode). Examples of commands that produce this error response are given in the following table.

Table 2. Example Commands that Generate <ErrConnect>

Example	Reason
`AT+ZV Security None`	Changing security level while a connection is up.
`AT+ZV Discovery`	Performing a device discovery while a connection is up.
`AT+ZV SPPConnect 00043e000000`	Establishing a connection to a device while a connection is exists with another.

4.2 ErrExecution

The `ErrExecution` error response will be sent if the command cannot complete for any reason.

Examples of commands that produce this error response are given in the following table.

Table 3. Example Commands that Generate <ErrExecution>

Example	Reason
`AT+ZV Security None`	Execution of command with correct syntax failed.
`AT+ZV Discovery`	Execution of command with correct syntax failed.

4.3 ErrFormat

The `ErrFormat` error response will be sent if kcSerial receives a command (text terminated by a carriage return or line feed) that does not match the expected format of starting with "AT+ZV".

Examples of commands that produce this error response are given in the following table.

Table 4. Example Commands that Generate <ErrFormat>

Example	Reason
`AT-ZV Discovery`	A valid command nam that does not start with the right prefix.
`SPPConnect 00043e000000`	A command does not start with AT+ZV
`abcdef`	A command does not start with AT+ZV

4.4 ErrInvalidParam

The `ErrInvalidParam` error response will be sent if the parameters for the requested command are not correct. The parameter(s) will be echoed back to the user starting from the parameter that was rejected.

Examples of commands that produce this error response are given in the following table.

Table 5. Example Commands that Generate <ErrInvalidParam>

Example	Response	Reason
`AT+ZV SPPConnect 8136`	`AT-ZV ErrInvalidParam 8136`	Numeric parameter not specified with required number of digits (BD address must always be 12 hex characters).
`AT+ZV SPPConnect 00043e008136 GOEP`	`AT-ZV ErrInvalidParam goep`	Unrecognized (or unsupported) symbolic parameter used.
`AT+ZV ChangeBaud 1600`	`AT-ZV ErrInvalidParam 1600`	Numeric parameter is out of range (specified baud rate is not supported by command).
`AT+ZV EnableBond 00043e000000 12345678901234567`	`AT-ZV ErrInvalidParam 12345678901234567`	String parameter (PIN)has too many characters.

4.5 ErrNumParam

The `ErrNumParam` error response will be sent if there are too few parameters for the requested command. A command sent with too many parameters does not generate an error; instead, the extra parameters are ignored.

Note: some commands will accept a variable number of parameters.

Examples of commands that produce this error response are given in the following table.

Table 6. Example Commands that Generate <ErrNumParam>

Example	Reason
`AT+ZV Security`	No parameters were specified.
`AT+ZV Sniff`	The minimum number of parameters was not specified.

4.6 ErrUnknownCmd

The `ErrUnknownCmd` error response will be sent if the requested command is not recognized. The unrecognized command will be echoed back to the host. Any parameters given will be ignored.

```
AT-ZV ErrUnknownCmd [unrecognized command]
```

An accepted command always starts with the command identifier:

```
AT+ZV
```

4.7 ErrInProgress

`ErrInProgress` is sent in response to `Discovery` command being issued when the previous one is still in progress.

```
AT+ZV Discovery [unrecognized command]
```

Examples of commands that produce this error response are given in the following table:

Table 7. Example Commands that Generate <ErrInProgress>

Example	Reason
`AT+ZV Discovery`	Trying to do discovery when the previous one has not completed.

4.8 Commands and Associated Errors

The table below summarizes which of the commands produce specific error responses. If a command can produce an error response, the column will be marked with an X.

Table 8. Possible ErrConnect Error Responses

Example	Err InvalidParam	Err NumParam	Err Execute	Err Connect	Err InProgress
Bond	X	X	X		
Bypass					
ChangeBaud	X	X			
ChangeDefaultBaud	X	X			
DefaultLocalName		X	X		
DeleteSmartCable					
DisableBond					
Discovery	X	X	X		X
DUNConnect	X	X		X	
DUNDisconnect					
EnableBond	X	X	X		
EraseBondTable					
ExitPark	X		X		
ExitSniff	X		X		
GPIOConfig	X	X			
GPIORead	X				
GPIOWrite	X	X			
Hold	X	X	X		
LocalName		X	X		
Park	X	X	X		
RemoteCmdDisconnect					
Reset					
Security	X	X	X		
SmartCableSetup	X	X			
Sniff	X	X	X		
SPPConnect	X	X		X	
SPPDisconnect					
UpdateInquiryScan	X	X	X		
UpdatePageScan	X	X	X		
Version					

5 Other Responses

The following subsections describe the 4 types of responses that occur under specific circumstances, not necessarily as a result of a specific command. They are:

- Reset
- Escape Sequence
- Controlled Disconnect
- Unexpected Disconnect

5.1 Reset

Upon either hardware reset or software reset (such as the `Reset` command), kcSerial will respond as follows after the reset is complete:

```
AT-ZV -CommandMode-
AT-ZV BDAddress [BD addr]
```

Because the BD address of the local device is reported during this response, the response is different than a response to the Escape Sequence.

5.2 Escape Sequence

If the Escape sequence `^#^$^%` is received and no connection is active, kcSerial will immediately respond with:

```
AT-ZV -CommandMode-
```

When the Escape Sequence is received while a connection is still active and there is no data for 2 seconds, kcSerial will respond (after those 2 seconds of no data) with the same string.

kcSerial will now be in command mode.

5.3 Controlled Disconnect

If the local host initiates a disconnect, it must first put the kcSerial interface into command mode (see the section on Escape Sequence directly above). After a successful disconnect command, the following response is made:

```
AT-ZV ConnectionDown
```

5.4 Unexpected Disconnect

Bluetooth connections may be unexpectedly dropped (e.g., in changing RF conditions). Although it is generally assumed that a disconnect will be “negotiated” on the application level, the remote device may initiate a disconnect. When that happens, the disconnect may be unexpected. This section applies to both the general case and the unexpected disconnect.

It is useful for the local host to be notified that a connection has been terminated when it isn't controlling the termination. An unexpected disconnect is essentially defined as a disconnect that occurs while in bypass mode. If this happens, kcSerial will respond with:

```
###NO CARRIER
AT-ZV -CommandMode-
```

It is the responsibility of the host to prevent this string from appearing in the data stream during normal operation.

If a remote disconnect occurs during command mode, this notification string is also sent. It will not be sent, however, if an initial setup cannot be established or if the disconnect is requested by the local device.

Hardware handshaking is not used to indicate a disconnection in this implementation. Modems can use DCD (data carrier detect) to notify the DTE (data terminal entity, e.g., computer) that a connection is either available or unavailable.

Lassen iQ GPS Module

Low-power, high-quality GPS solution for your mobile products

Key Features and Benefits

- **Ultra-low power: 86 mW**
- **Trimble quality at low cost**
- **Aided GPS through TSIP for faster acquisition**
- **Dual sensitivity modes with automatic switching**
- **12-channel simultaneous operation**
- **Supports NMEA 0183, TSIP, TAIP and DGPS**

Trimble's Lassen® iQ module is one smart buy. It adds powerful, 12-channel GPS functionality to your mobile product in a postage-stamp-sized footprint with ultra-low power consumption and extreme reliability—all at a very economical price.

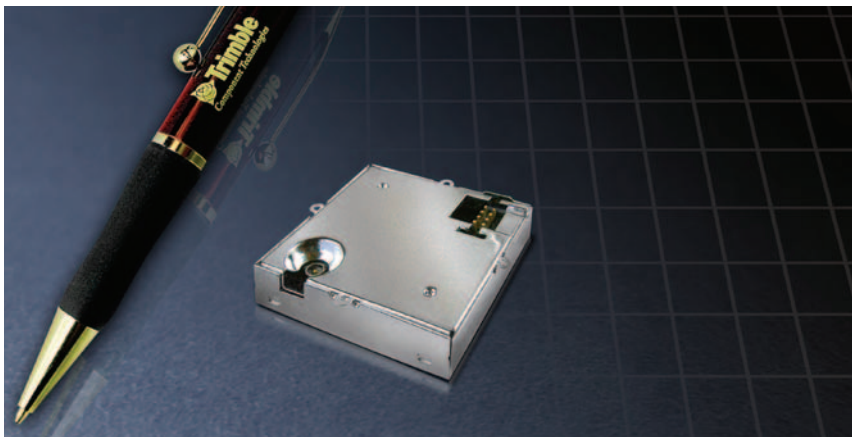
Designed for portable handheld, battery-powered applications such as cell phones, pagers, PDAs, digital cameras, and many others, the module is also ideal for standard GPS applications such as tracking.

The 12-channel Lassen iQ module is fully compatible with Trimble's popular Lassen SQ module. Using Trimble's breakthrough, patented FirstGPS® architecture, the module delivers complete position, velocity and time (PVT) solutions for use in the host application.

Powerful Performance

The Lassen iQ module features two GPS signal sensitivity modes: Standard and Enhanced. With Enhanced mode enabled, the module automatically switches to higher sensitivity when satellite signals are weak.

The module also supports TSIP download of critical startup information for fast acquisition. This aided GPS (A-GPS) startup provides hot start performance for each power-up.



Lassen iQ GPS receiver with metal shield

The Lassen iQ module is the only stamp-sized GPS product that supports the four most popular protocols: DGPS (RTCM), TSIP (Trimble Standard Interface Protocol), TAIP (Trimble ASCII Interface Protocol) and NMEA 0183.

The Lassen iQ module combines Trimble performance and quality with low cost. With an MTBF (mean time between failures) figure of 60 years, it is one of the most reliable GPS receivers on the market.

Hardware

A metal shield encloses the module for protection and ease of handling. The package has a small form factor, (approximately 26 mm x 26 mm, including the shield). It typically requires less than 90 mW of power at 3.3 VDC.

The highly integrated module is a miniature board containing Trimble GPS hardware core based on our Colossus® RF ASIC and IO-TS digital signal processor (DSP), a 32-bit RISC CPU and flash memory.

Antennas

The Lassen iQ module is compatible with active, 3.3-VDC antennas. Three such antennas are available from Trimble and are recommended for use according to your application; see the reverse side for antenna details.

The module provides both antenna open and short detection plus antenna short protection.

Starter Kit

The Lassen iQ Starter Kit provides everything you need to get started integrating state-of-the-art GPS capability into your application.

Lassen iQ GPS Module

Low-power, high-quality GPS solution for your mobile products

KEY FEATURES

- 12-channel simultaneous operation
- Ultra-low power consumption: less than 90 mW (27 mA) @ 3.3 V
- Dual sensitivity modes with automatic switching
- Aided GPS through TSIP
- Antenna open and short circuit detection and protection
- Compact size: 26 mm W x 26 mm L x 6 mm H
- Supports NMEA 0183, TSIP, TAIP, DGPS protocols
- Trimble quality at low cost

PERFORMANCE SPECIFICATIONS

General L1 (1575.42 MHz) frequency, C/A code, 12-channel, continuous tracking receiver

Update Rate TSIP @ 1 Hz; NMEA @ 1 HZ; TAIP @ 1 Hz

Accuracy

Horizontal:	<5 meters (50%), <8 meters (90%)
Altitude:	<10 meters (50%), <16 meters (90%)
Velocity:	0.06 m/sec
PPS (static):	±50 nanoseconds

Acquisition (Autonomous Operation in Standard Sensitivity Mode)

Reacquisition:	<2 sec. (90%)
Hot Start:	<10 sec (50%), <13 sec (90%)
Warm Start:	<38 sec (50%), <42 sec (90%)
Cold Start:	<50 sec (50%), <84 sec (90%)

Cold start requires no initialization. Warm start implies last position, time and almanac are saved by backup power. Hot start implies ephemeris also saved.

Operational (COCOM) Limits

Altitude:	18,000 m
Velocity:	515 m/s

Either limit may be exceeded, but not both

INTERFACE CHARACTERISTICS

Connectors I/O: 8-pin (2x4) 2 mm male header, micro terminal strip
ASP 69533-01
RF: Low-profile coaxial connector
H.FL-R-SMT (10), 50 Ohm

Serial Port 2 serial ports (transmit/receive)

PPS 3.3 V CMOS-compatible TTL-level pulse, once per second

Protocols TSIP, TAIP, NMEA 0183 v3.0, RTCM SC-104

NMEA Messages GGA, VTG, GLL, ZDA, GSA, GSV and RMC
Messages selectable by TSIP command
Selection stored in flash memory

ELECTRICAL CHARACTERISTICS

Prime Power +3.0 VDC to 3.6 VDC (3.3 V typ.)

Power Consumption Less than 90 mW (27 mA) @ 3.3 V

Backup Power +2.5 VDC to +3.6 VDC (3.0V typ.)

Ripple Noise Max 60 mV, peak to peak from 1 Hz to 1 MHz

Antenna Fault Protection Open and short circuit detection and protection

ENVIRONMENTAL SPECIFICATIONS

Operating Temperature –40° C to +85° C

Storage Temperature –55° C to +105° C

Vibration

0.008 g^2/Hz	5 Hz to 20 Hz
0.05 g^2/Hz	20 Hz to 100 Hz
–3 dB/octave	100 Hz to 900 Hz

Operating Humidity 5% to 95% R.H. non-condensing, at +60° C

PHYSICAL CHARACTERISTICS

Enclosure Metal enclosure with solder mounting tabs

Dimensions 26 mm W x 26 mm L x 6 mm H
(1.02" W x 1.02" L x 0.24" H)

Weight 6.5 grams (0.2 ounce) including shield

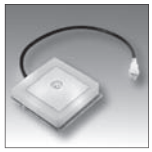
ORDERING INFORMATION & ACCESSORIES

Module Lassen iQ module, in metal enclosure with solder mounting tabs

Starter Kit Includes Lassen iQ module mounted on interface motherboard in a durable metal enclosure, AC/DC power converter, compact magnetic-mount GPS antenna, ultra-compact embedded antenna, serial interface cable, cigarette lighter adapter, TSIP, NMEA, and TAIP protocols, software toolkit and manual on CD-ROM

Antenna Transition Cable, MCX
RF cable for connecting antennas with MCX connector to on-module H.FL-RF connector.
Cable length: 10 cm

Antenna Transition Cable, SMA
RF cable for connecting antennas with SMA connector to on-module H.FL-RF connector.
Cable length: 12.9 cm.



Ultra-Compact Embedded Antenna
3.3V active miniature unpackaged antenna
Cable length: 8 cm
Dim: 22 mm W x 21 mm L x 8 mm H
(0.866" x 0.827" x 0.315")
Connector: HFL; mates directly to on-module RF connector

Compact Unpackaged Antenna
3V active micropatch unpackaged antenna
Cable length: 11 cm
Dim: 34.6 mm W x 29 mm L x 9 mm H
(1.362" x 1.141" x 0.354")
Connector: MCX; mates through the optional RF transition cable to on-module RF connector

Compact Magnetic-Mount Antenna, MCX or SMA
3V active micropatch antenna with magnetic mount
Cable length: 5 m
Dim: 42 mm W x 50.5 mm L x 13.8 mm H
(1.65" x 1.99" x 0.55")
Connectors: MCX or SMA, mates through the optional RF trasition cable to the module RF connector

Specifications subject to change without notice.

Trimble Navigation Limited is not responsible for the operation or failure of operation of GPS satellites or the availability of GPS satellite signals.

Trimble

www.trimble.com/iQ

Trimble Navigation Limited
Corporate Headquarters
645 North Mary Avenue
Sunnyvale, CA 94086
1-800-787-4225
1-408-481-7741
www.trimble.com

Trimble Navigation Europe
Ltd, UK
Phone: 44 1256-760-150

Trimble Export Ltd, Korea
Phone: 82-2-5555-361
korea_sales@trimble.com

Trimble Navigation Ltd,
China
Phone: 86-21-6391-7814

Lassen™ SQ GPS Receiver

System Designer Reference Manual

Part Number 47838-00
Revision A
June 2002

Trimble™

Corporate Office

Trimble Navigation Limited
Components Technologies Division
645 North Mary Avenue
Post Office Box 3642
Sunnyvale, CA 94088-3642
U.S.A.
Phone: +1-408-481-8940, 1-800-545-7762
Fax: +1-408-481-7744
www.trimble.com

Support Offices

Trimble Navigation Limited
Components Technologies Division
645 North Mary Avenue
Post Office Box 3642
Sunnyvale, CA 94088-3642
U.S.A.
Phone: +1-408-481-8940, 1-800-545-7762
Fax: +1-408-481-7744

Trimble Navigation Europe Limited
Trimble House
Meridian Office Park
Osborn Way, Hook
Hampshire RG27 9HX
England
Phone: +44-1256-760-150
Fax: +44-1-256-760-148

Copyright and Trademarks

Release Notice

This is the June 2002 release (Revision A) of the Lassen™ SQ GPS Receiver System Designer Reference Manual, part number 47838-00.

The following limited warranties give you specific legal rights. You may have others, which vary from state/jurisdiction to state/jurisdiction.

Hardware Limited Warranty

Trimble warrants that this Trimble hardware product (the “Product”) shall be free from defects in materials and workmanship and will substantially conform to Trimble’s applicable published specifications for the Product for a period of one (1) year, starting from the date of delivery. The warranty set forth in this paragraph shall not apply to software/firmware products.

Software and Firmware License, Limited Warranty

This Trimble software and/or firmware product (the “Software”) is licensed and not sold. Its use is governed by the provisions of the applicable End User License Agreement (“EULA”), if any, included with the Software. In the absence of a separate EULA included with the Software providing different limited warranty terms, exclusions, and limitations, the following terms and conditions shall apply. Trimble warrants that this Trimble Software product will substantially conform to Trimble’s applicable published specifications for the Software for a period of ninety (90) days, starting from the date of delivery.

Warranty Remedies

Trimble's sole liability and your exclusive remedy under the warranties set forth above shall be, at Trimble’s option, to repair or replace any Product or Software that fails to conform to such warranty (“Nonconforming Product”), or refund the purchase price paid by you for any such Nonconforming Product, upon your return of any Nonconforming Product to Trimble in accordance with Trimble’s standard return material authorization procedures.

Warranty Exclusions and Disclaimer

These warranties shall be applied only in the event and to the extent that: (i) the Products and Software are properly and correctly installed, configured, interfaced, maintained, stored, and operated in accordance with Trimble's relevant operator's manual and specifications, and; (ii) the Products and Software are not modified or misused. The preceding warranties shall not apply to, and Trimble shall not be responsible for defects or performance problems resulting from (i) the combination or utilization of the Product or Software with products, information, data, systems or devices not made, supplied or specified by Trimble; (ii) the operation of the Product or Software under any specification other than, or in addition to, Trimble's standard specifications for its products; (iii) the unauthorized modification or use of the Product or Software; (iv) damage caused by accident, lightning or other electrical discharge, fresh or salt water immersion or spray; or (v) normal wear and tear on consumable parts (e.g., batteries).

THE WARRANTIES ABOVE STATE TRIMBLE'S ENTIRE LIABILITY, AND YOUR EXCLUSIVE REMEDIES, RELATING TO PERFORMANCE OF THE PRODUCTS AND SOFTWARE. EXCEPT AS OTHERWISE EXPRESSLY PROVIDED HEREIN, THE PRODUCTS, SOFTWARE, AND ACCOMPANYING DOCUMENTATION AND MATERIALS ARE PROVIDED "AS-IS" AND WITHOUT EXPRESS OR IMPLIED WARRANTY OF ANY KIND BY EITHER TRIMBLE NAVIGATION LIMITED OR ANYONE WHO HAS BEEN INVOLVED IN ITS CREATION, PRODUCTION, INSTALLATION, OR DISTRIBUTION, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE, TITLE, AND NONINFRINGEMENT. THE STATED EXPRESS WARRANTIES ARE IN LIEU OF ALL OBLIGATIONS OR LIABILITIES ON THE PART OF TRIMBLE ARISING OUT OF, OR IN CONNECTION WITH, ANY PRODUCTS OR SOFTWARE. SOME STATES AND JURISDICTIONS DO NOT ALLOW LIMITATIONS ON DURATION OR THE EXCLUSION OF AN IMPLIED WARRANTY, SO THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

TRIMBLE NAVIGATION LIMITED IS NOT RESPONSIBLE FOR THE OPERATION OR FAILURE OF OPERATION OF GPS SATELLITES OR THE AVAILABILITY OF GPS SATELLITE SIGNALS.

Limitation of Liability

TRIMBLE'S ENTIRE LIABILITY UNDER ANY PROVISION HEREIN SHALL BE LIMITED TO THE GREATER OF THE AMOUNT PAID BY YOU FOR THE PRODUCT OR SOFTWARE LICENSE OR U.S.$25.00. TO THE MAXIMUM EXTENT PERMITTED BY APPLICABLE LAW, IN NO EVENT SHALL TRIMBLE OR ITS SUPPLIERS BE LIABLE FOR ANY INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES WHATSOEVER UNDER ANY CIRCUMSTANCE OR LEGAL THEORY RELATING IN ANY WAY TO THE PRODUCTS, SOFTWARE, AND ACCOMPANYING DOCUMENTATION AND MATERIALS, (INCLUDING, WITHOUT LIMITATION, DAMAGES FOR LOSS OF BUSINESS PROFITS, BUSINESS INTERRUPTION, LOSS OF BUSINESS INFORMATION, OR ANY OTHER PECUNIARY LOSS), REGARDLESS OF WHETHER TRIMBLE HAS BEEN ADVISED OF THE POSSIBILITY OF ANY SUCH LOSS AND REGARDLESS OF THE COURSE OF DEALING WHICH DEVELOPS OR HAS DEVELOPED BETWEEN YOU AND TRIMBLE. BECAUSE SOME STATES AND JURISDICTIONS DO NOT ALLOW THE EXCLUSION OR LIMITATION OF LIABILITY FOR CONSEQUENTIAL OR INCIDENTAL DAMAGES, THE ABOVE LIMITATION MAY NOT APPLY TO YOU.

Contents

Glossary

About this Manual

Welcome to System Designer Reference Manual for the Lassen SQ GPS receiver. This manual describes how to integrate and operate the Lassen SQ GPS receiver.

If you are not familiar with GPS, visit Trimble's website, www.trimble.com, for an interactive look at Trimble and GPS.

Trimble assumes that you are familiar with Microsoft Windows and know how to use a mouse, select options from menus and dialogs, make selections from lists, and refer to online help.

Technical Assistance

If you have a problem and cannot find the information you need in the product documentation, contact the Trimble Technical Assistance Center at 800-767-4822.

Your Comments

Your feedback about the supporting documentation helps us to improve it with each revision. To forward your comments, send an e-mail to ReaderFeedback@trimble.com.

CHAPTER

1

Starter Kit

- Product Overview
- Starter Kit
- Receiver Performance
- Interface Protocols
- Ordering Starter Kit Components
- Starter Kit Interface Unit
- Power
- Hardware Setup
- Software Toolkit

Product Overview

The Lassen SQ GPS receiver is a full featured, ultra low power receiver on a miniature form factor, suitable for a variety of mobile, embedded applications. The Lassen SQ GPS receiver incorporates Trimble's FirstGPS™ architecture in the form of two ASICS: Colossus RF down converter and IO-TS-C33 baseband chip. The IO-TS-C33 integrates Trimble's IO digital signal processor with the Epson C33 RISC processor, real-time clock, UART, and 1Mbit memory. Together with the colossus RF, this implementation of FirstGPS technology makes possible one of the smallest (26 mm x 26 mm x 6mm) and lowest power (100mW) GPS modules available.

The Lassen SQ GPS receiver outputs a complete position, velocity, and time (PVT) solution in the NMEA Version 3.0 ASCII protocol, and the Trimble TSIP binary protocol. A Pulse-Per-Second signal is available for very accurate timing applications.

Starter Kit

The Starter Kit makes it simple to evaluate the Lassen SQ GPS receiver's exceptional performance. The Starter Kit can be used as a platform for configuring the receiver module and as a platform for troubleshooting your design. The Starter Kit includes:

- Shielded Lassen SQ GPS module mounted on an interface motherboard in a durable metal enclosure. The motherboard accepts 9 - 32 VDC power and provides regulated +3.3V power to the Lassen SQ GPS receiver. The motherboard also contains:
  - 3.6V lithium battery that provides back-up power to the receiver.
  - Circuitry to convert the TTL output to RS-232, enabling the user to connect the RS-232 port in the Starter Kit to the PC COM port via an RS-232 cable connection.
- Compact Magnetic-Mount GPS Antenna with a 5 meter cable.
- Ultra-Compact Embedded Antenna with an 8 cm cable.
- 9-pin RS-232 interface cable.
- AC/DC power supply adapter (input: 100-240VAC, output: 12 VDC).
- DC power cable.
- Cigarette lighter adapter power cable.
- CD containing software tools used to communicate with the receiver, the System Designer Reference Manual, and "C" programming source routines to be used as a template for communicating with the receiver.

Removing the Lassen SQ GPS Module

The Lassen SQ GPS module is secured to the motherboard with double-sided adhesive tape allowing for easy removal and integration with the user's application. (The adhesive tape used by Trimble is 3M Scotch, part number 4945).

Follow these steps to remove the module from the motherboard:

- Unplug the I/O cable and the RF cable from the module.
- Use a small flat-head screw driver to pry the Lassen SQ GPS receiver module off the motherboard.

Warning – Once the Lassen SQ GPS receiver module is removed from the motherboard, the double-sided tape looses some of it's adhesive quality. This adhesive tape may only be re-used for laboratory testing. The original adhesive tape should not be re-used for drive testing the Starter Kit interface unit because the module could loosen and cause short circuit when contacting other motherboard components. If drive testing is required, use a new piece of double-sided adhesive tape to re-attach the Lassen SQ GPS receiver module to the motherboard.

Receiver Performance

The Lassen SQ GPS receiver is a complete 8-channel parallel tracking GPS receiver designed to operate with the L1 frequency, Standard Position Service, Coarse Acquisition code. Using two highly integrated Trimble custom integrated circuits, the receiver is designed in a modular format especially suited for embedded applications where small size and extremely low power consumption are required. The receiver features Trimble's latest signal processing code, a high-gain RF section for compatibility with standard 27 dB active gain GPS antennas, and a CMOS TTL level pulse-per-second (PPS) output for timing applications or for use as a general purpose synchronization signal.

The Lassen SQ GPS receiver acquires a position fix with minimal delay after power cycling. The battery back-up RAM is used to keep the Real Time clock (RTC) alive, and to store the following:

- Almanac
- Ephemeris
- Last position

User settings such as port parameters and NMEA settings can be stored in the receiver's non-volatile (Flash) memory. These settings are retained without application of main power or battery back-up power.

The Lassen SQ GPS receiver has a single configurable serial I/O communication port.

Warning – When customizing port assignments or characteristics, confirm that your changes do not affect your ability to communicate with the receiver (see Chapter 3, Software Interface).

Interface Protocols

The Lassen SQ GPS receiver operates using one of two protocols — Trimble Standard Interface Protocol (TSIP) or NMEA 0183. The factory default setting for the I/O port is TSIP bi-directional. Protocol selection and port characteristics are user configurable.

TSIP

TSIP is a powerful binary packet protocol that allows the system designer maximum configuration control over the GPS receiver for optimum performance in any number of applications. TSIP supports over 20 commands and their associated response packets for use in configuring the Lassen SQ GPS receiver to meet user requirements.

NMEA

NMEA 0183 is an industry standard protocol common to marine applications. NMEA provides direct compatibility with other NMEA-capable devices such as chart plotters, radars, etc. The Lassen SQ GPS receiver supports most NMEA messages for GPS navigation. NMEA messages and output rates can be user selected as required.

Ordering Starter Kit Components

The Lassen SQ GPS receiver is available in a Starter Kit or as an individual module and associated antenna. The Starter Kit (PN 47225-00) includes all the components necessary to quickly test and integrate the module:

- Compact Magnetic-Mount Antenna with 5m cable
- Ultra-Compact Embedded Antenna with 8cm cable
- AC/DC power supply adapter
- DC Power cable (3-wire)
- RS-232 interface cable DB9M/DB9F (pin to pin)
- Cigarette lighter adapter power cable
- CD-ROM containing software tools and the System Designer Reference Manual

Table 1.1 provides ordering information for the Lassen SQ GPS module and the associated antennas and cables.

Table 1.1 Lassen SQ GPS Receiver Ordering Information

Products	Part Number
Lassen SQ GPS receiver Module	46240-00
Lassen SQ GPS receiver Starter Kit	47225-00
Lassen SQ GPS receiver antenna transition cable	47274
Ultra-Compact Embedded Antenna, 3.3V, 8cm cable	45336-00
Compact Unpackaged Antenna, 3V, 11cm cable	39265-51
Compact Magnetic Mount Antenna, 3V, 5m cable	39265-50

Note – *Part numbers are subject to change. Confirm part numbers with your Trimble representative when placing your order.*

Starter Kit Interface Unit

The Starter Kit interface unit consists of a Lassen SQ GPS module attached to an interface motherboard, housed in a sturdy metal enclosure. This packaging simplifies testing and evaluation of the module by providing an RS-232 serial interface which is compatible with most PC communication ports. Power (9-32 VDC) is supplied through the power connector on the front of the interface unit. The motherboard features a switching power supply which converts this voltage input to the 3.3 volts required by the module. The DB9 connector allows for an easy connection to a PC serial port using the serial interface cable provided in the Starter Kit. The metal enclosure protects the module and the motherboard for testing outside of the laboratory environment.

The Lassen SQ GPS receiver is a single module encased in a sturdy metal enclosure. The dimensions of the receiver in this enclosure are 26 mm H x 26 mm L x 6 mm H (1.02” W x 1.02” L x 0.24” H). A straight-in, panel-mount RF connector (J1) supports the GPS antenna connection. The center conductor of the coaxial connector also supplies +3.3 VDC for the Low Noise Amplifier of the active antenna. An 8-pin (2x4), 0.09 inch header (J2) supports the serial interface (CMOS TTL level), the pulse-per-second (PPS) signal (CMOS TTL level), and the input power (+3.3 VDC). Figur e1.1 illustrates the module in the metal enclosure.

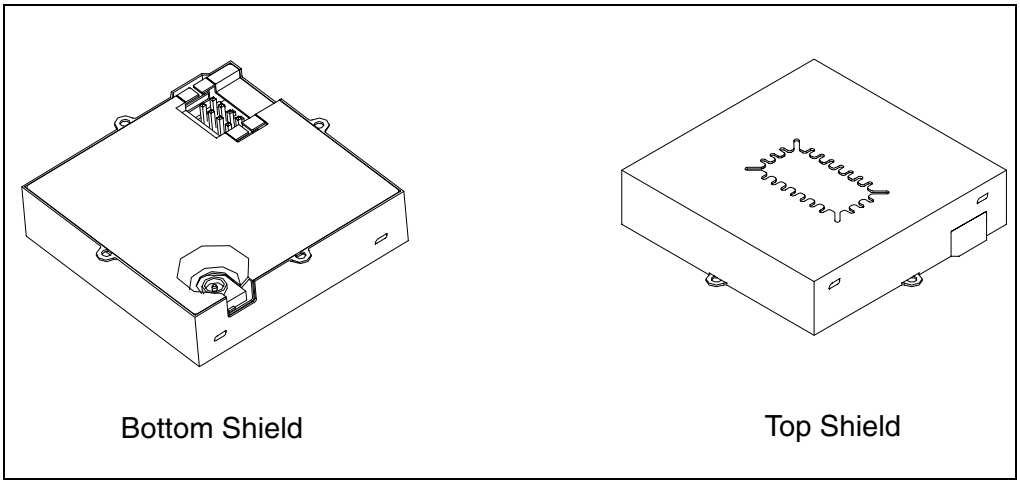


Figure 1.1 Lassen SQ GPS receiver Module

The interface motherboard includes a 9 to 32 VDC switching power supply which provides regulated +3.3 VDC power to the receiver, and contains circuitry which provides an RS-232 interface to a computer. A 3.6V lithium backup battery enables quick hot starts. The TTL level PPS is brought directly out to Pin 9 of the Port 2 DB9 connector on the front of the interface unit.

The Starter Kit includes an AC/DC converter for powering the module from an AC wall socket. The metal enclosure (see Figure 1.2.) provides 2 DB9 interface port connectors, an antenna connector, and a power connector. Port 1 is for serial I/O.

The mounting plate is secured to the metal enclosure with four screws. The eight pin I/O header on the receiver module connects to a mating connector on a ribbon cable. The ribbon cable is attached to a mating I/O connector on the interface motherboard. Figure 1.2 illustrates the Starter Kit interface unit.

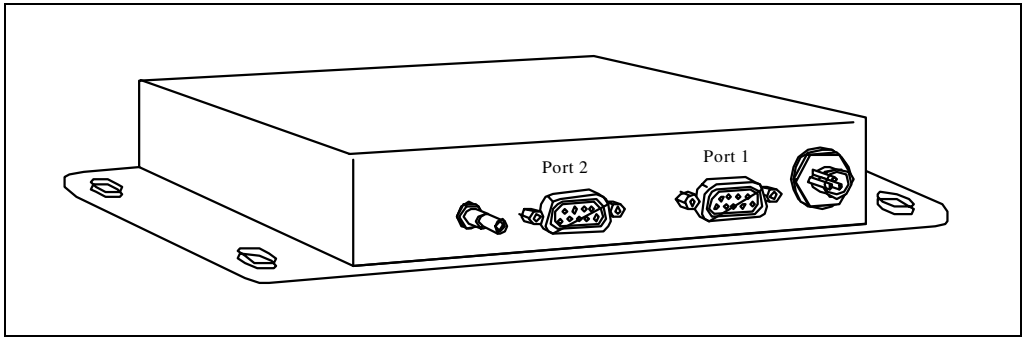


Figure 1.2 Starter Kit Interface Unit

Serial Port Interface

The Starter Kit interface unit is a DCE (Data Communication Equipment) device. To connect to a host computer, or DTE (Data Terminal Equipment) device, use a straight through cable. To connect a Differential Radio (DCE device) to the receiver (DCE Device) use a cross over cable or null modem cable.

Table 1.2 Port 1 Pinouts

Pin	Description
1	NC
2	TX
3	RX
4	NC
5	GND
6	NC
7	NC
8	NC
9	NC

Table 1.3 Port 2 Pinouts

Pin	Description
1	NC
2	NC
3	NC
4	NC
5	GND
6	NC
7	NC
8	NC
9	PPS Out

Pulse-Per-Second (PPS)

The Lassen SQ GPS receiver provides a four microsecond wide, CMOS compatible TTL level Pulse-Per-Second (PPS). The PPS is a positive pulse available on pin 9 of the port 2 DB9 connector of the interface unit (see Table 1.3). The rising edge of the PPS pulse is synchronized with respect to UTC. The timing accuracy is ±95 nanoseconds when valid position fixes are being reported.

The rising edge of the pulse is typically less than 20 nanoseconds. The distributed impedance of the attached signal line and input circuit can affect the pulse shape and rise time. The PPS can drive a load up to 5mA without damaging the module. The falling edge of the pulse should not be used. The PPS is always on (early PPS) and is driven by the Real Time Clock (RTC) until the receiver acquires GPS time from the satellite and generates position fixes. The PPS is output immediately after main power is applied, and continues even if the receiver loses GPS lock. The drift of the PPS, when the receiver is not tracking satellites, is unspecified and should not be used for synchronization.

Note – *Trimble has measured better than 50 nanosecond accuracy on the Lassen SQ GPS receiver's PPS signal in static mode. For more information on use of the Lassen SQ GPS receiver in timing applications, contact your Trimble sales representative.*

Power

The Lassen SQ GPS receiver receiver is designed for embedded applications and requires a regulated +3.3 VDC input (+3.0 to +3.6 VDC). The receiver provided in the Starter Kit is installed on a motherboard, providing a DC power regulator which converts a 9 to 32 VDC input to the regulated 3.3 VDC required by the receiver. Power can be applied to the interface unit using one of three options: the DC power cable (Figure 1.3), the AC/DC power converter (Figure 1.4), or the cigarette lighter adapter.

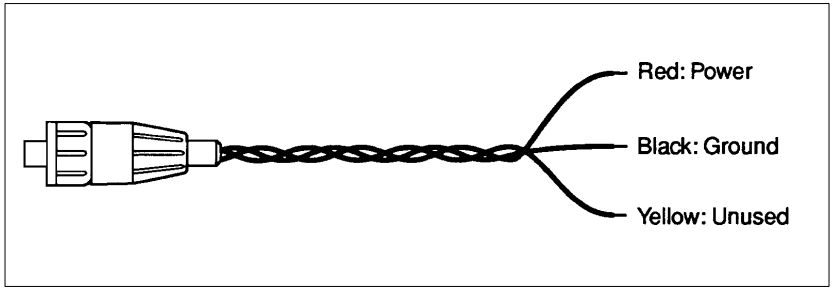


Figure 1.3 DC Power Cable

The DC power cable is ideal for bench-top or automotive testing environments. The power cable is terminated at one end with a 3-pin plastic connector which mates with the power connector on the metal enclosure. The un-terminated end of the cable provides easy connection to a DC power supply. Connect the red power lead to a source of DC positive +9 to +32 VDC, and connect the black power lead to ground. This connection supplies power to both the receiver and the antenna. The combined power consumption of the interface unit with the receiver and the antenna is 133 - 145 milli-amps.

Note – *To ensure compliance with CE conducted emissions requirements when using the DC power cable, the Starter Kit interface unit must be bonded to a ground plane.*

Note – *The yellow wire of the DC power cable is not used. Battery back-up power is provided by a factory installed 3.6V lithium battery on the motherboard.*

The AC/DC power converter may be used as an alternate power source for the interface unit. The AC/DC power converter converts 110 or 220 VAC to a regulated 12 VDC compatible with the interface unit. The AC/DC power converter output cable is terminated with a 3-pin connector compatible with the power connector on the metal enclosure. The AC power cable is not provided in the kit, since this cable is country-specific. The input connector is a standard 3-prong connector used on many desktop PCs.

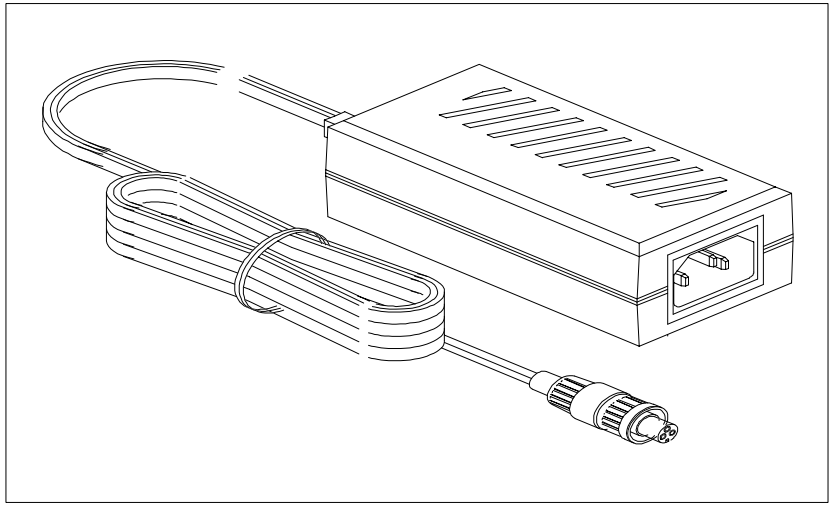


Figure 1.4 AC/DC Power Converter

Hardware Setup

The Lassen SQ GPS receiver supports the TSIP and NMEA protocols. A single port supports both the input/output of TSIP messages and the output of NMEA messages. Follow the steps below to setup the Starter Kit interface unit. Figure 1.5 illustrates the setup.

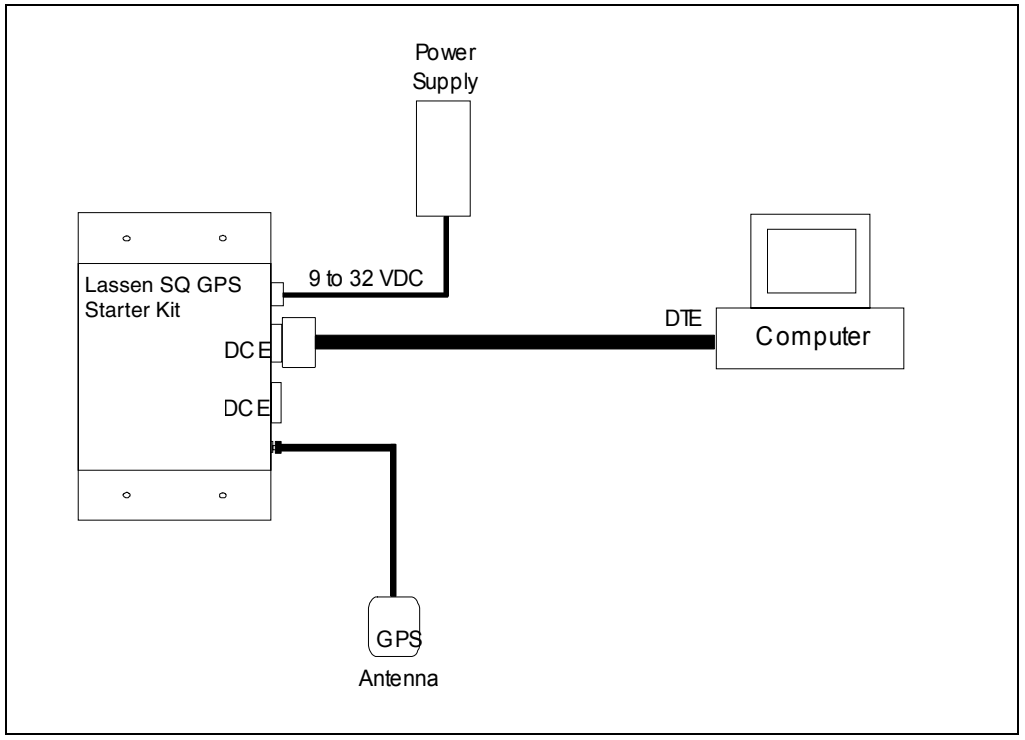


Figure 1.5 Starter Kit Interface Unit

1. When using the TSIP protocol, connect one end of the 9-pin serial interface cable to Port 1 of the interface unit. Connect the other end of the cable to COM1 or COM2 on a PC. A 9-pin-to-25-pin adapter may be required for the serial interface connection to a PC, if your PC has a 25-pin communication port.
2. Connect the antenna cable to the interface unit. This connection is made by pushing the antenna cable connector onto the MCX connector on the module. Place the antenna so that it has a clear view of the sky.

Note – *To remove the antenna cable, grasp the antenna mating MCX connector and pull from the MCX connector mounted on the interface unit.*

3. Using either the DC power cable or an AC/DC power converter, connect to the 3-pin power connector on the interface unit.
   - DC Power Cable — connect the terminated end of the power cable to the power connector on the interface unit. Connect the red lead to DC positive voltage (+9 to +32 VDC) and black power lead to DC ground. The yellow wire is not used. Switch on the DC power source.
   - AC/DC Power Converter — connect the output cable of the converter to the 3-pin power connector on the interface unit. Using the appropriate 3-prong AC power cable (not provided), connect the converter to an AC wall socket (110 VAC or 220 VAC). The AC power cable is not provided in the Starter Kit.

Software Toolkit

The CD provided in the Starter Kit contains the SQ_Monitor and the TSIPCHAT interface programs used to monitor GPS performance and to assist system integrators in developing a software interface for the GPS module. These applications are described in detail in Appendix B, *TSIP User's Guide*.

SQ_Monitor runs on the Windows 95/98/2000 platforms. TSIPCHAT runs under the DOS operating system on a 386 or higher processor.

Following are quick start instructions for using the SQ_Monitor application to monitor the receiver's performance.

1. Connect one end of the serial interface cable to Port 1 of the interface unit. Connect the other end of the cable to the COM port of your PC.
2. Turn on the DC power source or plug in the AC/DC converter.
3. Insert the CD in the computer's CD-ROM drive.
4. The SQ_Monitor program may be run directly off the CD or it may be copied onto your computer's hard drive. To run the program off the CD, initiate the SQ_Monitor.exe file.
5. When the SQ_Monitor screen appears, the TX and RX indicators appear in the lower left corner of the status bar. A blinking TX indicates that the PC is transmitting commands to the receiver; a blinking RX indicates that the PC is receiving reports from the receiver. If either of these indicators stop blinking, there is no activity. The PC COM port settings appear in the lower right corner of this same status bar.
6. After a GPS antenna is connected to the receiver and the receiver has achieved a position fix, the transmitted position reports, time, velocity, satellites tracked, and GPS receiver status appear on the screen. The receiver also sends a health report every few seconds, even if satellites are not being tracked.

Note – *If the SQ_Monitor program displays a question mark (?) in a data field, the receiver has not reported a status for this field. If a (?) remains in the data field, the GPS module may not be communicating with the computer. Re-check the interface cable connections and verify the serial port selection and settings. If the communication failure continues after checking all connections and settings, please call the Trimble Technical Assistance Center (TAC) at 1 (800) 767-4822.*

CHAPTER

2

Hardware Integration

In this chapter:

- General Description
- Connectors
- Power Requirements
- Serial Interface
- Pulse-Per-Second (PPS)
- Mounting
- GPS Antennas

General Description

Trimble's new Lassen SQ GPS receiver adds complete GPS functionality to mobile products, in a postage-stamp-sized footprint with ultra-low power consumption. Using Trimble's breakthrough FirstGPS™ architecture, the module delivers complete position, velocity and time (PVT) solutions for use in mobile, battery-powered applications such as cell phones, pagers, PDAs, and digital cameras.

The Lassen SQ GPS module is packaged in a tiny form factor (26 mm x 26 mm x 6 mm, including the metal shield). It typically requires only 100 mW of power (at 3.3 VDC). Total typical power usage, including the Trimble 3.3 VDC miniature antenna, is 133 mW. The module includes flash memory for field upgrades and for storing the user configuration.

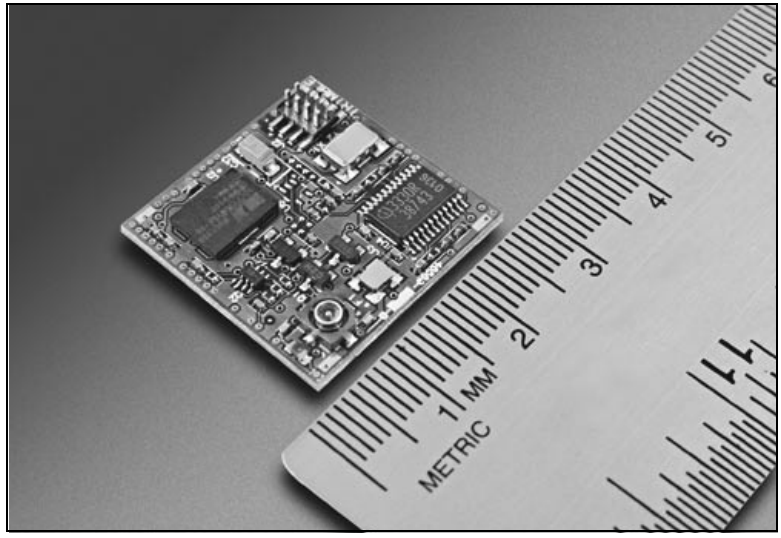


Figure 2.1 Lassen SQ GPS Receiver Board without Shield

Connectors

Digital IO/Power Connector

The Lassen SQ GPS module uses a single 8-pin (2x4) male header connector for both power and data I/O. The power and I/O connector, J2, is a surface mount micro terminal strip. This connector uses 0.09 inch (2.286mm) high pins on 0.05 inch (1.27mm) spacing. The manufacturer of this connector is Samtec, part number ASP 69533-01.

Note – *See Appendix D for mechanical drawings and specifications.*

Mating Connectors

The customer must supply his own mating connector to the Lassen SQ GPS receiver 8-pin (2x4) connector. There are two mating connectors available:

- Surface-Mount Mating Connector

  A recommended surface mount mating connector is Samtec's part number CLP-104-02.

  When a surface-mount mating connector is chosen, the RF connector must be attached to the Lassen SQ GPS module prior to securing the module to the user's PCB. The mounting tabs may be used for securing the Lassen SQ GPS module to the PCB when using the surface-mount mating scheme.

- Cable Strip Mating Connector

  A low profile, cable strip mating connector is the second I/O mating method. A recommended cable strip part is Samtec's part number FFSD-04-?-XX part. The user will need to substitute the following letters and numbers into the part number when ordering this part where the '?' and 'XX' symbols occur: for the '?' symbol substitute the letter S for single end or D for double end; for the 'XX' symbol substitute the overall length in inches, ± 1/8 inch, with a 2 inch minimum. Since the signals are CMOS TTL level signals, Trimble does not recommend cable lengths of longer than six inches.

  If the cable strip I/O connector scheme is used, the connector side of the Lassen SQ module will be facing up and the mounting tabs will be on the top of the module away from PCB. The RF connector is easily accessible, using this interfacing methodology.

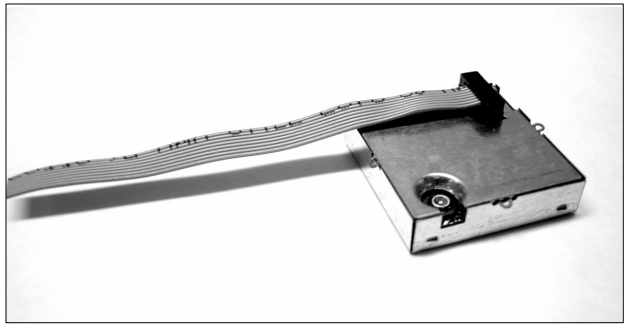


Figure 2.2 Cable Strip Mating Connector

RF Connector

The RF connector mounted on the Lassen SQ module is a Hirose connector, part number H.FL-R-SMT (10) 50 Ohm. The mating RF connector is Hirose H.FL-LP-XXX where XXX depends on the cable type.

Figure 2.3 Lassen SQ GPS Module with Connectors

Possible cable manufactures include the following:

- 1.48 mm diameter (single shield) cable:
  - CO-6F/FH-SB manufactured by Hitachi Cable Ltd.
  - UL1979 manufactured by Junkosha Co., Ltd.
  - 0.8DS-PBE manufactured by Sumitomo Electric Industry Co., Ltd.
- 1.32 mm diameter cable (double shield):
  - A12B0733 manufactured by Junkosha Co., Ltd.
- 1.47 mm diameter cable (single shield):
  - CXN2571 manufactured by W.L. Gore & Associated, Inc.

Trimble offers three antennas for use with the Lassen SQ GPS receiver receiver: The Ultra-Compact Embedded Antenna, which mates directly to the RF connector. The Compact Unpackaged Antenna and the Compact Magnetic-Mount Antenna, which mate through the optional RF transition cable to the module's RF connector. For more information on the antennas, see pag e30.

Digital IO/Power Connector Pinout

The digital IO/Power connector pinout information is listed in Table 2.1.

Table 2.1 J2 I/O Connector Signals

Pin number	Function	Description
1	TXD A	Serial Port A transmit, CMOS/TTL
2	GND	Ground, Power and Signal
3	RXD A	Serial Port A receive, CMOS/TTL
4	PPS	Pulse-Per-Second, CMOS/TTL
5	Reserve	No connect
6	Reserve	No connect
7	Prime Power (VCC)	+3.3 VDC to ± 0.3 VDC
8	Battery Backup Power	+2.5 VDC to + 3.6 VDC

Power Requirements

The Lassen SQ GPS module requires +3.3 VDC ±0.3 VDC at 33 mA, typical excluding the antenna. The on-board capacitance is 10 µF. An important design consideration for power is the module's internal clock frequency at 12.504 MHz ± 3 KHz. Interference spurs on prime power in this narrow frequency band should be kept to less than 1mV.

The receiver does not require any special power up or down sequencing. The receiver power is supplied through pin 7 of the I/O connector. See Table 2.2 for the +3.3 VDC power specifications.

Warning – The Lassen SQ GPS receiver is ready to accept TSIP commands approximately 2.1 seconds after power -up. If a command is sent to the receiver within this 2.1 second window, the receiver will ignore the command. The Lassen SQ GPS receiver will not respond to commands sent within the 2.1 second window and will discard any associated command data.

Battery Back-up

The Lassen SQ GPS receiver provides an input for battery back-up (BBU) power to keep the module's RAM memory alive and to power the real-time clock when the receiver's prime power is turned off. RAM memory is used to store the GPS almanac, ephemeris, and last position. User configuration data, including port parameters and receiver processing options can be stored in non-volatile Flash which does not require back-up power. By using battery back-up, time to first fix is reduced to 20 seconds (typical). Though not required, providing BBU power can reduce time to first fix. A 3.6 volt lithium battery used for back-up power can last up to five years.

Warning – If battery power is not present, the receiver's power can be turned off and then back on to force a system reset and a cold start. The receiver should be off for no less than 3 minutes to ensure that the RAM memory does not retain any old data due to the residual voltage from the power supply. Alternatively, you can enter the cold start command (TSIP Packet 0x1E) to force a system reset and a cold start. Cycle power and issue the cold start TSIP command immediately after switching the power back on.

Note – *2.5V is the minimum allowable battery back-up voltage. When the battery back-up power output drops below 2.5V, the real-time clock may not operate over the specified temperature range. This can also significantly extend the time to first fix.*

Table 2.2 Power Requirements

Signal	Voltage	Current	J2 Pin #
VCC	3.0 to 3.6	33mA	7
Battery Back-up	2.5 to 3.6	19µA (at 3.3 volts, +25°C)	8
Ground	0	--	2

Serial Interface

As an embedded design, the Lassen SQ GPS module provides direct CMOS compatible TTL level serial I/O. The RX and TX signals on the J2 I/O connector are driven directly by the UART on the Lassen SQ GPS receiver. Interfacing these signals directly to a UART in your application circuitry provides direct serial communication without the complication of RS-232 or RS-422 line drivers.

Note – *The serial I/O signals on J2 are TTL level. They are not inverted or driven to RS-232 levels.*

Pulse-Per-Second (PPS)

The Lassen SQ GPS receiver provides a four microsecond wide, CMOS compatible TTL level Pulse-Per-Second (PPS). The PPS is a positive pulse available on pin 4 of the power and I/O connector. The rising edge of the PPS pulse is synchronized with respect to UTC. The timing accuracy is ±95 nanoseconds when valid position fixes are being reported.

The rising edge of the pulse is typically less than 20 nanoseconds. The distributed impedance of the attached signal line and input circuit can affect the pulse shape and rise time. The PPS can drive a load up to 5mA without damaging the module. The falling edge of the pulse should not be used. The PPS is always on (early PPS) and is driven by the Real Time Clock (RTC) until the receiver acquires GPS time from the satellite and is getting fixes. The PPS is output immediately after main power is applied, and continues even if the receiver loses GPS lock. The drift of the PPS, when the Lassen SQ GPS receiver is not tracking satellites, is unspecified and should not be used for synchronization.

Note – *Trimble Navigation has measured better than 50 nanoseconds accuracy on the Lassen SQ GPS receiver PPS signal in static mode. For more information on the use of the Lassen SQ GPS module in timing applications, contact your Trimble sales representative.*

Mounting

The Lassen SQ GPS PCB is encased in a metal enclosure. The enclosure acts as a protective case. There are four mounting solder tabs on the bottom of the enclosure. When the surface-mount mating connector is used, the mounting tabs may be used for securing the Lassen SQ GPS module on the user's PCB. When the cable strip I/O connector scheme is used, the connector side of the Lassen SQ GPS module will be faced up and the mounting tabs will be on the top of the module away from PCB.

The Lassen SQ GPS module can be attached to the integrator platform by many methodologies including solder, glue, double sided adhesive tape, and custom hold down mounts for the module's mounting tabs.

Note – *See Appendix D for mechanical drawings and specifications regarding the spacing of the mounting tabs and the dimensions of the enclosure.*

GPS Antennas

The antenna receives the GPS satellite signals and passes them to the receiver. The GPS signals are spread spectrum signals in the 1575 MHz range and do not penetrate conductive or opaque surfaces. Therefore, the antenna must be located outdoors with a clear view of the sky. The Lassen SQ GPS receiver requires an ***active*** antenna. The received GPS signals are very low power, approximately -130 dBm, at the surface of the earth. Trimble's active antennas include a preamplifier that filters and amplifies the GPS signals before delivery to the receiver.

Trimble offers three antennas for use with the Lassen SQ GPS receiver described below and in Appendix D.

1. The Ultra-Compact Embedded GPS Antenna with an HFL connector, is ideal for portable and mobile applications. This unpackaged antenna is approximately the same size as the module itself, and can be easily integrated into mobile applications. This antenna is supplied with the Starter Kit (see Figure 2.4).
2. A Compact Unpackaged Antenna with an MCX connector, slightly larger than the ultra-compact model (see #1 above), mates to the Hirose connector on the Lassen SQ GPS module with an optional RF transition cable (see Figur e2.5).
3. A Compact Magnetic-Mount GPS Antenna with a 5 m cable and an MCX connector. This antenna provides for a flexible, movable installation. The MCX output connector mates to the Hirose connector on the Lassen SQ GPS module with an optional RF transition cable. This antenna is supplied with the Starter Kit (see Figure 2.6). The MCX connector on the end of the antenna cable mates to the MCX connector in the front of the Starter Kit interface unit.

Warning – When magnetic-mount or permanent-mount GPS antennas are installed on a metal surface for prolonged periods, care must be taken to insulate the antennas in order to prevent galvanic corrosion.

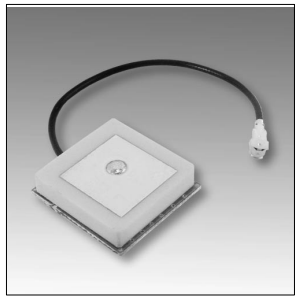


Figure 2.4 Ultra-Compact Embedded GPS Antenna

Figure 2.5 Compact Unpackaged GPS Antenna

Figure 2.6 Compact Magnetic-Mount GPS Antenna

CHAPTER

3

Software Interface

In this chapter:

- Start-up
- Communicating with the Lassen SQ GPS receiver Module
- Port Protocol and Data Output Options
- Custom Port Configuration
- When prompted, select the factory default option.

Start-up

Lassen SQ GPS module is a complete 8-channel parallel tracking GPS receiver designed to operate with the L1 frequency, standard position service, Coarse Acquisition code. When connected to an external GPS antenna, the receiver contains all the circuitry necessary to automatically acquire GPS satellite signals, track up to 8 GPS satellites, and compute location, speed, heading, and time. The receiver will automatically begin to search for and track GPS satellite signals at power-up.

The performance of a GPS receiver at power-on is determined largely by the availability and accuracy of the satellite ephemeris data and the availability of a GPS system almanac.

The first time the receiver is powered-up, it is searching for satellites from a cold start (no almanac). While the receiver will begin to compute position solutions within the first two minutes, the receiver must continuously track satellites for approximately 15 minutes to download a complete almanac. This initialization process should not be interrupted. With a complete almanac and back-up power, the time to first fix can typically be shortened to less than 45 seconds. The receiver will respond to commands almost immediately after power-up (see Warning below).

Note – *See Chapter 4 for further detail on ephemeris data and the GPS almanac.*

Warning – The Lassen SQ GPS receiver is ready to accept TSIP commands approximately 2.1 seconds after power -up. If a command is sent to the receiver within this 2.1 second window, the receiver will ignore the command. The Lassen SQ GPS receiver will not respond to commands sent within the 2.1 second window and will discard any associated command data.

Communicating with the Lassen SQ GPS receiver Module

The Lassen SQ GPS receiver supports two message protocols: TSIP and NMEA. Communication with the module is through a CMOS compatible, TTL level serial port. The port characteristics can be modified to accommodate your application requirements. Port parameters are stored in non-volatile memory (flash) which does not require backup power. Table 3.1. lists the default port characteristics.

Software Tools

The Software Tools provided on the Starter Kit CD-ROM include both user friendly Windows and DOS applications to facilitate communication with the receiver, via the Trimble Standard Interface Protocol (TSIP). This CD also includes sample C source code and reusable routines to aid in developing applications.

Note – *The TSIP and NMEA protocols are discussed beginning on page 37 of this chapter, and in Appendix A, Appendix B, and Appendix C.*

Port Configuration

The Lassen SQ GPS module has a single I/O port.Tabl e3.1 provides the default protocol and port configuration for the receiver, as delivered from the factory. TSIP IN/OUT is the default protocol.

Table 3.1 Default Protocol and Port Configuration

Input		Output	
Protocol	**Default Setup**	**Protocol**	**Default Setup**
TSIP	Baud Rate: 9600 Data Bits: 8 Parity: Odd Stop Bits: 1 No Flow Control	TSIP	Baud Rate: 9600 Data Bits: 8 Parity: Odd Stop Bits: 1 No Flow Control

The Lassen SQ GPS receiver can also be configured to output NMEA messages. The industry standard port characteristics for NMEA are:

- Baud Rate: 4800
- Data Bits: 8
- Parity: None
- Stop Bits:1
- No Flow Control

Any standard serial communications program, such as Windows Hyper-Terminal or PROCOMM, can be used to view the NMEA output messages. TSIP is a binary protocol and outputs raw binary serial data that cannot be read when using Windows Terminal or PROCOMM. To view the output of the TSIP protocol in text format, use the TSIPCHAT or the SQ_Monitor program (see the CD-ROM provided in the Starter Kit).

The serial port driver in the TSIPCHAT Tool Kit matches the Lassen SQ GPS receiver serial port characteristics. The TSIPPRNT program converts binary data logged with the TSIPCHAT program into text that may be printed and displayed. Both of these tools are included in the Software Developer's Toolkit.

Warning – When using the TSIP protocol to change port assignments or settings, confirm that your changes do not affect the ability to communicate with the receiver (e.g., selecting the PC COM port settings that do not match the receiver's, or changing the output protocol to TSIP while not using TSIPCHAT).

Port Protocol and Data Output Options

Protocol Configuration and Interface

The factory default protocol for the Lassen SQ GPS receiver is the Trimble Standard Interface Protocol (TSIP), for both input and output. The serial port setting is 9600 baud 8-odd-1. The receiver protocol can be re-configured using TSIP command packet 0xBC, in conjunction with TSIPCHAT, SQ_Monitor, or a user written serial interface program. See Table 2 for protocol configuration options, and Appendix A for details on the 0xBC command packet.

TSIPCHAT provides the simplest means to communicate with the receiver using a PC (386 or higher) running either the DOS or Windows operating systems. Responses are displayed on the computer monitor in text format.

SQ_Monitor, a Windows-based GUI, provides a versatile graphical interface for monitoring TSIP data. This application allows the user to view complete receiver operations including data output, status and configuration. In this application, the entry of command packets is replaced by traditional point and click pull-down menus.

C source code routines for TSIPCHAT are also provided on the CD contained in the Starter Kit. When used as software design templates, this source code can significantly speed-up code development.

The protocol settings and options are stored in battery-backed Random-Access-Memory (BBRAM). They can also be saved into the non-volatile memory (Flash), if desired, using command 0x8E-26. See to Appendix A for additional information on Flash storage for custom operation.

TSIP Data Output Modes

TSIP is the default protocol for the Lassen SQ GPS receiver. This binary language offers users a wide variety of commands and reports. TSIP enables the Lassen SQ GPS receiver to operate in two data output modes, both available during operation. In Query Mode, packet data is returned in response to input query packets. In Automatic Mode, a selected group of data packets is output continuously at two fixed rates – every second and every five seconds. The format and ensemble of the automatic output packets is configured using packets 0x35, 0x70, and 0x8E-20 (see Appendi xA for packet details). Packet settings are stored in BBRAM. They can also be saved in non-volatile memory (Flash) using command packet 0x8E-26. See Appendix A for additional information on Flash storage for custom operation.

Automatic TSIP Output Options and Defaults

Default 0x35 setting (byte 0, 1, 2, 3 = 2, 2, 0, 0):

- Position and velocity data precision: 4 byte floating point
- Position output option and format (byte 0 setting):
  - Latitude – radian
  - Longitude – radian
  - Altitude – meters (WGS-84)
- No super-packet output (byte 0 setting)
- Velocity output option and format (byte 1 setting):
  - East Velocity – meters/sec.; + for East
  - North Velocity – meters/sec.; + for North
  - Up Velocity – meters/sec.; + for Up

- Time reports option and format (byte 2 setting):
  - GPS (not UTC) time of week – seconds; 4 byte floating point
  - Extended GPS week number – weeks; 2 byte integer (INT16)
  - GPS UTC offset – seconds; 4 byte floating point

Default 0x70 setting (byte 0, 1, 2, 3 = 1, 1, 1, 0):

- Position-Velocity Dynamic Filter enabled
- Position-Velocity static Filter enabled
- Altitude Filter enabled

Default 0x8E-20 setting (byte 1 = 1):

- 0x8F-20 output is included in the super-packet for automatic output IF packet 0x35 selects the super-packet for automatic output options

Automatic TSIP Output Packets (fixed rate)

One second interval:

- 0x4A – (1) GPS position fix; (2) clock bias and time of fix; {20 byte format}
- 0x56 – velocity fix
- 0x6D – (1) list of satellites used for position fixes; (2) PDOP, HDOP, VDOP; (3) fix mode
- 0x82 – DGPS position fix mode

Five second interval:

- 0x41 – (1) GPS time of the week (seconds); (2) extended GSP week number; (3) GPS UTC offset(seconds)
- 0x46 – health of receiver
- 0x4B – (1) Machine/Code ID; (2) Real-time-clock availability status; (3) almanac validity status; (4) having super-packet support status

Packet Output Order

After power up or a software reset (packet 0x1E), seven start-up packets are sent, only once, by the receiver in this order: 45, 46, 4B, 4A, 56, 41, 82

Before position fixes are available, the 1 second and 5 second interval packets are sent in this order, periodically:

- Every one second for 5 seconds: 6D, 82
- Every five seconds 41, 46, 4B

When position fixes are available, the 1 second and 5 second interval packets are sent in this order, periodically:

- Every one second for 4 seconds: 4A, 56, 6D, 82
- Every 5 seconds: 4A, 56, 41, 46, 4B, 6D, 82

NMEA 0183 Protocol and Data Output Options

The National Marine Electronics Association (NMEA) protocol is an industry standard data protocol which was developed for the marine industry. Trimble has chosen to adhere stringently to the NMEA 0183 data specification as published by the NMEA. The Lassen SQ GPS receiver also adheres to the NMEA 0183, Version 3.0 specification.

NMEA data is output in standard ASCII sentence formats. Message identifiers are used to signify what data is contained in each sentence. Data fields are separated by commas within the NMEA sentence. In the Lassen SQ GPS receiver, NMEA is an output only protocol. The NMEA protocol is described in detail in Appendix C.

The receiver is shipped from the factory with the TSIP protocol configured on Port 1. The receiver can be reconfigured using TSIP command packet 0xBC, in conjunction with TSIPCHAT, SQ_Monitor, or a user written serial interface program.

The NMEA output messages selection and message output rate can be set using TSIP command packet 0x7A. The default setting is to output the GGA and VTG messages at a 1 second interval, when the receiver output protocol is configured to NMEA, using packet 0xBC.

If NMEA is to be permanent for the application, the protocol configuration (0xBC) and NMEA message output setting (0x7A) can be stored in the non-volatile memory (on-board flash) using TSIP command 0x8E-26.

Custom Port Configuration

TSIPCHAT can be used to customize the Lassen SQ GPS receiver configuration settings and to save a configuration to non-volatile memory. The most recent port configuration is stored in BBRAM. This eliminates the need to repeat setup each time the receiver power is cycled. However, if the battery-backed power is accidentally lost, the port configuration automatically resets to either what was saved in the non-volatile memory (Flash) or to the factory default.

Tip – To ensure continuous operation, store all port configuration changes in the non-volatile memory.

Following are step-by-step instructions for using TSIPCHAT to customize Lassen SQ GPS receiver port configuration.

Customizing the Configuration

1. Insert the CD in the CD-ROM drive of your computer.
2. Open a DOS window and set the path to the TSIPCHAT location.
3. To run the program, type TSIPCHAT –c1 if attached to PC COM1, or type TSIPCHAT –c2 if attached to PC COM2.
4. Power-up the receiver. Automatic report streams should be scrolling up in the DOS window. Assuming that your receiver is set to the default configuration, the settings will be: 9600 baud, 8-odd-1.

Note – *If data is not being output after receiver power up, use the "^i" command in TSIPCHAT to reset the COM1/COM2 setting in PC (not the receiver).*

Tip – Entering "?" in the TSIPCHAT window displays all the available commands and their corresponding TSIP packets.

---

5. To re-configure the port settings and protocol, type “U” and respond to the input prompts. At the end of this procedure, select the option that resets the PC COM port to match the new settings. Communication should resume almost immediately.

Saving the Configuration

1. Before storing the new configuration in Flash, confirm that the receiver has been configured to the desired settings.

Warning – Record the new serial port settings. If power is lost, this will speed-up recovery. Alternatively, the receiver can always be returned to the default configuration.

---

2. To save the configuration to Flash:
   - Enter “=” to access the command list page for the 0x8E command packet.
   - Enter “s”, to send the 0x8E-26 command packet.
   - Communication is momentarily suspended while the configuration is being stored in Flash.
3. To confirm that the configuration changes have been saved, turn-off the power supply and the battery back-up for a few minutes. Then, power-up the receiver and confirm that the configuration changes have been retained. Alternatively, you can use Packet 1E to command a cold start.

Note – *Command packet 0x8E-26 executes storage of various types of receiver settings in addition to the port and protocol. See Table 3.2 for a complete list of the settings that can be stored in Flash memory.*

Returning to the Factory Settings

At any time, the receiver can be returned to the factory default configuration, using command packet 0x1E.

1. Type "^k" to invoke the 0x1E command.
2. When prompted, select the factory default option.

Table 3.2 TSIPChat Command Settings Stored in Flash Memory Command Packet 0x8E-26

TSIP Command ID	TSIPCHAT Keystroke	Description	TSIP Response ID
0x35	'O'	TSIP input/output formatting – Superpacket output (on/off) – Position format (LLA and/or ECEF) – Precision (double or single) – altitude format (MSL or HAE) – Timetag format (GPS or UTC) – SNR format (AMU or C/N_0 – Automatic pseudorange output	0x55
0x70	'l'	Position filter controls – Position filter on/off – Static filter on/off – Altitude filter on/off	0x70
0x7A	'q'	NMEA message formats and schedule – NMEA output messages – NMEA output interval	0x7B

Table 3.2 TSIPChat Command Settings Stored in Flash Memory Command Packet 0x8E-26 (Continued)

TSIP Command ID	TSIPCHAT Keystroke	Description	TSIP Response ID
0xBB	'p'	GPS configuration parameters – Operating dimension (2D, 3D,...) – DGPS mode – Dynamics mode – Elevation mask – SNR mask – DOP mask – PDOP switch – DGPS correction age	0xBB
0xBC	'U'	Serial port configuration – Protocol: input, output – Baud, data bits, parity, stop bits	0xBC
0x8E-20	'= g'	Fixed point superfix control (default = on)	0x8F-20

Timing Applications

The Lassen SQ GPS receiver is an excellent source for accurate system timing. Two examples of applications requiring accurate time are environmental data acquisition and synchronization of communications networks. The timing functions of the receiver are supported by the TSIP protocol and the PPS signal. See Report Packet 41 in Appendix A for a description of the time function reports for TSIP.

Note – *GPS time differs from UTC (Universal Coordinated Time) by a variable integer number of seconds: UTC = (GPS time) - (GPS UTC Offset)*

As of April 2002, the GPS UTC offset was 13 seconds. The offset increases by 1 second approximately every 18 months. System designers should plan to read the offset value as a part of the timing interface to obtain UTC. The GPS week number is in reference to a base week (Week #0), starting January 6, 1980.

The current GPS UTC offset is contained within the almanac transmitted by the GPS system. The Lassen SQ GPS receiver must have a complete almanac before the offset data is valid.

Extended GPS Week Number

The Lassen SQ GPS receiver outputs the Extended GPS Week Number as the absolute number of weeks since the beginning of GPS time or January 6, 1980. If the true GPS Week Number is desired, ignore the extra MSBs of the Extended GPS Week Number and use only the 10 LSBs (bytes 4 and 5 of Packet 0x41).

CHAPTER

4

Operation and Performance

In this chapter:

- Introduction
- GPS Satellite Message
- Satellite Acquisition and Time to First Fix
- Satellite Mask Settings
- Standard Operating Modes
- Position Accuracy
- Coordinate Systems
- Performance Characteristics
- GPS Timing
- System Architecture

Introduction

This chapter describes the Lassen SQ GPS receiver satellite acquisition and tracking processes, performance characteristics, and system architecture. This discussion assumes that you are familiar with the basic theory of the Global Positioning System. Before proceeding to the detailed discussion of the satellite acquisition and tracking process, please review the GPS satellite message description on the next page.

The Lassen SQ GPS receiver satellite acquisition and tracking algorithms can achieve a position solution without any initialization. The receiver automatically selects and tracks the best combination of satellites to compute position and velocity. As satellites move out of view, the Lassen SQ GPS receiver automatically acquires new satellites and includes them in the solution set as required.

GPS Satellite Message

Every GPS satellite transmits the Coarse/Acquisition (C/A) code and satellite data modulated onto the L1 carrier frequency (1575.42 MHz). The satellite data transmitted by each satellite includes a satellite almanac for the entire GPS system, its own satellite ephemeris and its own clock correction.

The satellite data is transmitted in 30-second frames. Each frame contains the clock correction and ephemeris for that specific satellite, and two pages of the 50-page GPS system almanac. The almanac is repeated every 12.5 minutes. The ephemeris is repeated every 30 seconds.

The system almanac contains information about each of the satellites in the constellation, ionospheric data, and special system messages. The GPS system almanac is updated weekly and is typically valid for months. The ephemeris contains detailed orbital information for a specific satellite. Ephemeris data changes hourly, but is valid for up to four hours. The GPS control segment updates the system almanac weekly and the ephemeris hourly through three ground-based control stations. During normal operation, the Lassen SQ GPS receiver module updates its ephemeris and almanac as needed.

The performance of a GPS receiver at power-on is determined largely by the availability and accuracy of the satellite ephemeris data and the availability of a GPS system almanac.

Satellite Acquisition and Time to First Fix

Cold-Start

The term “cold-start” describes the performance of a GPS receiver at power-on when no navigation data is available. “cold” signifies that the receiver does not have a current almanac, satellite ephemeris, initial position, or time. The cold-start search algorithm applies to a Lassen SQ GPS receiver which has no memory of its previous session (i.e., is powered on without the memory backup circuit connected to a source of DC power). This is the “out of the box” condition of the GPS module as received from the factory.

In a cold-start condition the receiver automatically selects a set of eight satellites and dedicates an individual tracking channel to each satellite, to search the Doppler range frequency for each satellite in the set. If none of the eight selected satellites is acquired after a pre-determined period of time (time-out), the receiver will select a new search set of eight satellites and will repeat the process, until the first satellite is acquired. As satellites are acquired, the receiver automatically collects ephemeris and almanac data. The Lassen SQ GPS receiver uses the knowledge gained from acquiring a specific satellite to eliminate other satellites, those below the horizon, from the search set. This strategy speeds the acquisition of additional satellites required to achieve the first position fix.

The cold-start search sets are established to ensure that at least three satellites are acquired within the first two time-out periods. As soon as three satellites are found, the receiver will compute an initial position fix. The typical time to first fix is less than 2 minutes.

A complete system almanac is not required to achieve a first position fix. However, the availability and accuracy of the satellite ephemeris data and the availability of a GPS almanac can substantially shorten the time to first fix.

Warm Start

In a warm-start condition the receiver has been powered down for at least one hour but has stored a current almanac, an initial position, and time, in memory.

When connected to an external back-up power source (battery back-up), the Lassen SQ GPS receiver retains the almanac, approximate position, and time to aid in satellite acquisition and reduce the time to first fix. When an external back-up battery is not used, the TSIP protocol allows the almanac, an initial position, and time to be uploaded to the receiver via the serial port, to initiate a warm start.

During a warm start, the Lassen SQ GPS receiver identifies the satellites which are expected to be in view, given the system almanac, the initial position and the approximate time. The receiver calculates the elevation and expected Doppler shift for each satellite in this expected set and directs the eight tracking channels in a parallel search for these satellites.

The warm start time to first fix, when the receiver has been powered down for more than 60 minutes (i.e. the ephemeris data is old), is usually less than 45 seconds.

Hot Start

A hot start strategy applies when the Lassen SQ GPS receiver has been powered down for less than 60 minutes, and the almanac, position, ephemeris, and time are valid. The hot start search strategy is similar to a warm start, but since the ephemeris data in memory is considered current and valid, the acquisition time is typically less than 20 seconds.

Garage Search Strategy

During a warm start search, the Lassen SQ GPS receiver knows which satellites to search for, based on the system almanac, the initial position (last known position) and the current time. In some cases, the receiver may not be able to acquire the expected satellite signals (e.g., a vehicle parked in a garage or a vessel in a covered berth). Trimble's patented “garage search” strategy, also known as a split search, is designed for such situations.

If the receiver does not acquire the expected set of satellites within 5 minutes of power-on, some of the eight tracking channels will continue to search for the expected satellites (warm search) while the remaining channels are directed in a cold start search. This strategy minimizes the time to first fix in cases where the stored almanac, position and time are invalid. The stored information is flushed from memory, if the cold start search proves effective and the warm search fails.

System Reset

The Lassen SQ GPS receiver can be reset with software commands or by cycling power. A system reset will cause the receiver to restart and begin the satellite acquisition and tracking process again. There are three types of system resets: soft reset, hard reset, and factory reset. The TSIP protocol supports all three resets using the 0x1E command. Power cycling can be used for either the soft reset or the hard reset.

A soft reset is a system restart. In a soft reset, the system will attempt to acquire satellites using the satellite information and last position data stored in RAM, and the time information supplied by the real-time clock. There are two ways to initiate a soft reset:

- Cycling main power while keeping the memory and the real-time clock alive with back-up power.
- Issuing Command Packet 0x25.

A soft reset is the same as a warm or hot start, if the information contained in memory and supplied by the real-time clock is valid.

A hard reset is a system restart that results in satellite acquisition search using a default search set. Any data contained within RAM memory is discarded and the real-time clock is re-initialized. Even if back-up power is supplied, the information from memory and the real-time clock is not used. There are two ways to initiate a soft reset:

- Issuing the 0x1E command with a value of 0x4B.
- Cycling power without back-up power applied. Using this method, power must be removed for at least 3 minutes to ensure any residual memory storage is erased. If power is cycled rapidly, the 0x1E command with a value of 0x4B must then be issued to the receiver 2.5 seconds after power is restored to ensure a hard reset.

A factory reset is used to restore all the factory default settings into the receiver. Any user settings stored in Flash memory will be erased. Issuing the 0x1E command with a value of 0x46 will initiate a factory reset.

Satellite Mask Settings

Once the Lassen SQ GPS receiver has acquired and locked onto a set of satellites, which pass the mask criteria listed in this section, and has obtained a valid ephemeris for each satellite, it will output regular position, velocity and time reports according to the protocol selected.

The default satellite masks observed by the Lassen SQ GPS receiver are listed in Table 1. These masks serve as the screening criteria for satellites used in fix computations and ensure that position solutions meet a minimum level of accuracy. The Lassen SQ GPS receiver will only output position, course, speed and time when a satellite set can be acquired which meets all of the mask criteria. The satellite masks can be adjusted in GPS receivers accepting the TSIP protocol. (See Appendix A for details on key setup parameters.)

Table 4.1 Satellite Mask Settings

Mask	Setting
Elevation	5°
SNR	3
PDOP	12
PDOP Switch	6

Elevation Mask

Satellites below a 5° elevation are not used in the position solution. Although low elevation satellites can contribute to a lower/better PDOP, the signals from low elevation satellites are poorer quality, since they suffer greater tropospheric and ionospheric distortion than the signals from higher elevation satellites. These signals travel further through the ionospheric and tropospheric layers.

In addition, low elevation satellites can contribute to frequent constellation switches, since the signals from these satellites are more easily obscured by buildings and terrain. Constellation switches can cause noticeable jumps in the position output. Since worldwide GPS satellite coverage is generally excellent, it is not usually necessary to use satellites below a 5° elevation to improve GPS coverage time. In some applications, like urban environments, a higher mask may be warranted to minimize the frequency of constellation switches and the impact of reflected signals.

SNR Mask

Although the Lassen SQ GPS receiver is capable of tracking signals with SNRs as low as 0, the default SNR mask is set to 3 to eliminate poor quality signals from the fix computation and minimize constellation switching. Low SNR values may result from:

- Low Elevation Satellites
- Partially Obscured Signals (e.g. Dense Foliage)
- Multi-Reflected Signals (Multi-Path)

The distortion of signals and the frequent constellation switches associated with low-elevation satellites were discussed above. In mobile applications, the attenuation of signals by foliage is typically a temporary condition. Since the Lassen SQ GPS receiver can maintain lock on signals with SNRs as low as 0, it offers excellent performance when traveling through heavy foliage.

Multi-reflected signals, also known as Multi-path, can degrade the position solution. Multi-path is most commonly found in urban environments with many tall buildings and a preponderance of mirrored glass, which is popular in modern architecture. Multi-reflected signals tend to be weak (low SNR value), since each reflection attenuates the signal. By setting the SNR mask to 3 or higher, the impact of multi-reflected signals is minimized.

DOP Mask

Position Dilution of Precision (DOP) is a measure of the error caused by the geometric relationship of the satellites used in the position solution. Satellite sets which are tightly clustered or aligned in the sky will have a high DOP and will contribute to a lower position accuracy. For most applications, a DOP mask of 12 offers a satisfactory trade-off between accuracy and GPS coverage time. With world-wide GPS coverage now available, the DOP mask can be lowered even further for many applications without sacrificing coverage.

PDOP Switch

The default positioning mode for the Lassen SQ GPS receiver is Automatic. In this mode, the receiver attempts to generate a 3-dimensional (3D) position solution, when four or more satellites meeting the mask criteria are visible. If such a satellite set cannot be found, the receiver will automatically switch to 2-dimensional (2D) mode. The PDOP switch establishes the trade-off between 3D positioning and PDOP. With the PDOP Switch set to 6, the receiver will compute a 2D position with a HDOP below 6 rather than a 3D position with a PDOP greater than 6, even when four or more satellites are visible.

Note – *PDOP Switch is only used in Auto mode.*

Standard Operating Modes

The tracking mode controls the allocation of the receiver's tracking channels and the method used for computing position fixes.

Fix Modes

The Lassen SQ GPS receiver offers three positioning modes: 2D Manual, 3D Manual, and Automatic 2D/3D. Automatic 2D/3D is the default mode for the Lassen SQ GPS receiver. The positioning mode can be modified in receivers accepting TSIP commands. (See Appendix A for more information on the TSIP protocol.)

2D ManualLassen SQ GPS Receiver

In 2D Manual mode, the Lassen SQ GPS receiver will only generate 2-dimensional (2D) position solutions (latitude and longitude only), regardless of the number of visible satellites. If the altitude is not entered, the receiver uses zero as the default altitude. The greater the deviation between the actual and default altitudes, the greater the error in the 2D position. For TSIP applications, enter local altitude in MSL/HAE via TSIP packet 2AH (see Appendix A).

3D Manual

In 3D Manual mode, the Lassen SQ GPS receiver will only generate 3-dimensional (3D) position solutions (latitude, longitude, and altitude). A 3D solution requires at least four visible satellites which pass the mask criteria. If less than four conforming satellites are visible, the Lassen SQ GPS receiver will suspend position data outputs.

2D/3D Automatic

The default operating mode for the Lassen SQ GPS receiver is 2D/3D Automatic. In this mode, the Lassen SQ GPS receiver attempts to generate a 3-dimensional (3D) position solution, if four or more satellites meeting the mask criteria are visible. If only three satellites are visible which meet the mask criteria, the Lassen SQ GPS receiver will automatically switch to 2-dimensional (2D) mode and will use the last calculated altitude, if available, or the default altitude in the position solution. In 2D/3D Automatic mode, the PDOP switch is active.

Position Accuracy

GPS position accuracy is degraded by atmospheric distortion, satellite geometry, satellite clock errors, and receiver clock errors. Effective models for atmospheric distortion of satellite signals have been developed to minimize the impact of tropospheric and ionospheric effects. The impact of satellite clock errors is minimized by incorporating the clock corrections transmitted by each satellite used in the position solution.

Coordinate Systems

Once the Lassen SQ GPS receiver achieves its first fix, it is ready to commence output of position, velocity, and time information. This information is output over serial communication channel in either the TSIP or NMEA protocol, as determined by the settings of the receiver. These protocols are defined in the Appendices. To change from one protocol to another, see Appendix A.

TSIP Coordinate Systems

TSIP has the widest choice of coordinate systems. The output format is chosen by TSIP command 0x35. The output formats include the following:

- LLA position — Latitude, longitude, altitude (LLA) according to the WGS-84 ellipsoid. Altitude can be chosen to be height above ellipsoid (HAE) or height above mean sea level (MSL).
- ENU velocity — ENU velocity is the velocity in East, North, and Up coordinates. These coordinates are easily converted to speed and heading.
- ECEF position and velocity — ECFF position and velocity is Earth-Centered, Earth-Fixed frame is a Cartesian coordinate frame with its center at the earth's center, the z-axis through the North Pole, and the x-axis through longitude 0 degrees, latitude 0 degrees.Velocity is reported relative to the same axes.

There are also two time coordinate systems:

- GPS time — GPS time is determined by an ensemble of atomic clocks operated by the Department of Defense (DOD).
- UTC time — UTC time is the world standard maintained by an ensemble of atomic clocks operated by government organizations around the world. UTC time replaced GMT (Greenwitch Mean Time) as the world standard, in 1986.

GPS time is steered relative to Universal Coordinated Time (UTC). GPS does not recognize leap seconds resulting in a situation where GPS time is currently 13 seconds ahead of UTC time. Time tags for most output messages can be in either UTC time or GPS time, as chosen by TSIP command 0x35.

NMEA 0183

The NMEA 0183 protocol only supports LLA format and UTC time. Velocity is always described as horizontal speed and heading; vertical speed is not output.

Performance Characteristics

Update Rate

The Lassen SQ GPS receiver computes and outputs position solutions once per second, on the second. NMEA outputs can be scheduled at a slower rate using TSIP command 0x7A (see Appendix A).

Dynamic Limits

The dynamic operating limits for the Lassen SQ GPS receiver are listed below. These operating limits assume that the GPS module is correctly embedded and that the overall system is designed to operate under the same dynamic conditions.

Table 4.2 Lassen SQ GPS Receiver Operating Limits

Operation	Limit
Acceleration	4 g (39.2 m/s^2)
Jerk	20 m/s^3
Speed	500 m/s
Altitude	18,000 m

Note – *The Lassen SQ GPS Receiver firmware contains an algorithm that allows either the speed limit or altitude limit to be exceeded, but not both. This allows the receiver to be used in high altitude (research balloon) applications without a special factory configuration.*

Re-Acquisition

Re-acquisition time for a momentary signal blockages is typically under 2 seconds.

When a satellite signal is momentarily interrupted during normal operation, the receiver continues to search for the lost signal at the satellite's last known Doppler frequency. If the signal is available again within 15 seconds, the receiver will normally re-establish track within two seconds. If the lost signal is not re-acquired within 15 seconds, the receiver initiates a broader frequency search. The receiver will continue to search for the satellite until it falls below the elevation mask.

GPS Timing

In many timing applications, such as time/frequency standards, site synchronization systems and event measurement systems, GPS receivers are used to discipline local oscillators.

The GPS constellation consists of 24 orbiting satellites. Each GPS satellite contains a highly-stable atomic (Cesium) clock, which is continuously monitored and corrected by the GPS control segment. Consequently, the GPS constellation can be considered a set of 24 orbiting clocks with worldwide 24-hour coverage.

GPS receivers use the signals from these GPS “clocks” to correct its internal clock, which is not as stable or accurate as the GPS atomic clocks. GPS receivers like the Lassen SQ GPS receiver output a highly accurate timing pulse (PPS) generated by its internal clock, which is constantly corrected using the GPS clocks. This timing pulse is synchronized to UTC within ±95 ns.

In addition to serving as a highly accurate stand-alone time source, GPS receivers are used to synchronize distant clocks in communication or data networks. This synchronization is possible since all GPS satellite clocks are corrected to a common master clock. Therefore, the relative clock error is the same, regardless of which satellite or satellites are used. For timing applications requiring a “common clock”, GPS is the ideal solution.

The position and time errors are related by the speed of light. Therefore, a position error of 100 meters corresponds to a time error of approximately 333 ns. The hardware and software implementation affects the GPS receiver's PPS accuracy level. The receiver's clocking rate determines the PPS steering resolution.

The Lassen SQ GPS receiver clocking rate is 3.126 MHz. This rate corresponds to a steering resolution of ±160 ns.

Serial Time Output

Both the TSIP and NMEA protocols include time messages. See Report Packet 41 in Appendix A for a description of the time reports for each protocol.

Note – *GPS time differs from UTC (Universal Coordinated Time) by a variable, integer number of seconds UTC=(GPS time)-(GPS UTC offset).*

As of April 2002, the GPS UTC offset was 13 seconds. The offset has historically increased by 1 second about every 18 months. System designers should plan to read the offset value as a part of the timing interface to obtain UTC. The GPS week number is in reference to a base week (Week #0), starting January 6, 1980.

Pulse-Per-Second (PPS)

The Lassen SQ GPS receiver provides a four microsecond wide, CMOS compatible TTL level Pulse-Per-Second (PPS). The PPS is a positive pulse available on pin 4 of the Lassen SQ GPS receiver power and I/O connector. The rising edge of the PPS pulse is synchronized with respect to UTC. The timing accuracy is ±95 nanoseconds when valid position fixes are being reported.

The rising edge of the pulse is typically less than 20 nanoseconds. The distributed impedance of the attached signal line and input circuit can affect the pulse shape and rise time. The PPS can drive a load up to 5mA without damaging the module. The falling edge of the pulse should not be used. The PPS is always on (early PPS) and is driven by the Real Time Clock (RTC) until the receiver acquires GPS time from the satellite and is obtaining fixes. The PPS is output immediately after main power is applied, and continues even if the receiver loses GPS lock. The drift of the PPS, when the Lassen SQ GPS receiver is not tracking satellites, is unspecified and should not be used for synchronization.

Note – *Trimble Navigation has measured better than 50 nanoseconds accuracy of the Lassen SQ GPS receiver PPS signal in static mode. For more information on timing applications, contact your Trimble sales representative.*

System Architecture

The Lassen SQ GPS receiver (see Figur e4.3) uses eight processing channels operating on the L1 frequency of 1575.42 MHz and using the coarse acquisition (C/A) code. The module uses custom integrated circuitry designed by Trimble to track the GPS satellite signals. These ICs also contain support circuitry to the navigation processor. An integrated 32-bit microprocessor is used for tracking, computing a position, and performing the I/O operations.

The Lassen SQ GPS receiver receives the amplified GPS satellite signals through the antenna feed line connector and passes them to the RF down converter. A highly stable crystal reference oscillator operating at 12.504 MHz is used by the down converter to produce the signals used by the 8-channel signal processor. The 8-channel signal processor tracks the GPS satellite signals and extracts the carrier code information as well as the navigation data at 50 bits per second.

Operation of the tracking channels is controlled by the navigation processor. The tracking channels are used to track the highest eight satellites above the horizon. The navigation processor will then use the optimum satellite combination to compute a position. The navigation processor also manages the ephemeris and almanac data for all of the satellites, and performs the data I/O.

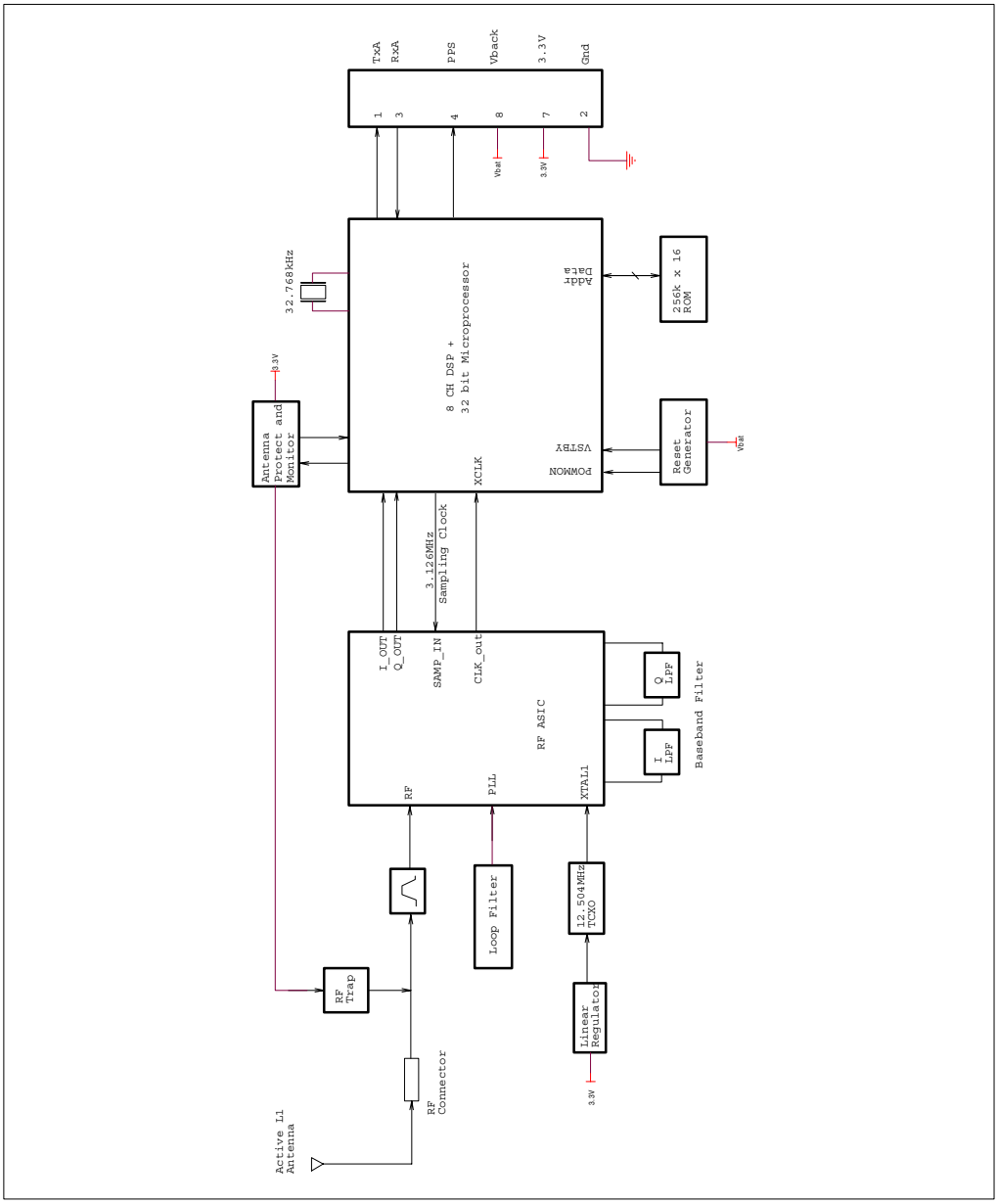


Figure 4.3 Lassen SQ GPS receiver Block Diagram

APPENDIX

A

Trimble Standard Interface Protocol (TSIP)

The Trimble Standard Interface Protocol (TSIP) provides the system designer with over 20 commands that may be used to configure a GPS receiver for optimum performance in a variety of applications. TSIP enables the system designer to customize the configuration of a GPS module to meet the requirements of a specific application.

This appendix provides the information needed to make judicious use of the powerful features TSIP has to offer, to greatly enhance overall system performance, and to reduce the total development time. The reference tables beginning on pag e70 will help you determine which packets apply to your application. For those applications requiring customization see Customizing Receiver Operations, page 71 for a detailed description of the key setup parameters. Application guidelines are provided for each TSIP Command Packet, beginning on page 84.

Interface Scope

The Trimble Standard Interface Protocol is used extensively in Trimble receiver designs. The protocol was originally created for the Trimble Advanced Navigation Sensor (TANS) and is colloquially known as the TANS protocol even though the protocol applies to many other devices.

The Lassen SQ GPS receiver has one serial I/O communications port. This port is a bi-directional control and data port utilizing Trimble Standard Interface Protocol (TSIP). This port may also be used to receive TSIP commands and to output industry standard ASCII-based NMEA messages. The data I/O port characteristics and other options are user programmable and can be stored in non-volatile flash memory.

The TSIP protocol is based on the transmission of packets of information between the user equipment and the unit. Each packet includes an identification code (1 byte, representing 2 hexadecimal digits) that identifies the meaning and format of the data that follows. Each packet begins and ends with control characters.

This document describes in detail the format of the transmitted data, the packet identification codes, and all available information over the output channel to allow the user to choose the data required for his particular application. As will be discussed, the receiver transmits some of the information (position and velocity solutions, etc.) automatically when it is available, while other information is transmitted only on request. Additional packets may be defined for particular products and these will be covered in the specifications for those products as necessary.

The TSIPCHAT utility, part of the GPS Tool Kit, is designed to exercise many of the TSIP packets.

Packet Structure

TSIP packet structure is the same for both commands and reports. The packet format is:

```
<DLE> <id> <data string bytes> <DLE> <ETX>
```

Where:

- <DLE> is the byte 0x10
- <ETX> is the byte 0x03
- <id> is a packet identifier byte, which can have any value excepting <ETX> and <DLE>.

The bytes in the data string can have any value. To prevent confusion with the frame sequences <DLE> <ID> and <DLE> <ETX>, every <DLE> byte in the data string is preceded by an extra <DLE> byte ('stuffing'). These extra <DLE> bytes must be added ('stuffed') before sending a packet and removed after receiving the packet. Notice that a simple <DLE> <ETX> sequence does not necessarily signify the end of the packet, as these can be bytes in the middle of a data string. The end of a packet is <ETX> preceded by an odd number of <DLE> bytes.

Multiple-byte numbers (integer, float, and double) follow the ANSI/IEEE Std. 754 IEEE Standard for binary Floating-Point Arithmetic. They are sent most-significant byte first. This may involve switching the order of the bytes as they are normally stored in Intel based machines. Specifically:

- UINT8 = Byte: An 8 bit unsigned integer.
- UINT16 = Word: A 16 bit unsigned integer.
- INT16 = Integer: A 16 bit integer.
- INT32 = Long: A 32 bit integer.
- UINT32 = ULong: A 32 bit unsigned integer.
- Single — Float, or 4 byte REAL has a precision of 24 significant bits, roughly 6.5 digits.
- Double — 8 byte REAL has a precision of 52 significant bits. It is a little better than 15 digits.

Automatic Output Packets

The Lassen SQ GPS receiver receiver is configured to automatically output the following packets. For minimal system implementations, these output packets provide all of the information required for operation including time, position, velocity, and receiver and satellite status and health. Position and velocity are reported using one or more of the packets listed below, depending on the selected I/O options. While there are other packets automatically output, the following packets provide the information most commonly used. No input packets are required.

Table A.1 Automatic Output Packets

Output Packet ID	Description	Reporting Interval
0x41	GPS time	5 seconds
0x42, 0x83, 0x4A, 0x84, 0x8F-20	position (choose packet with I/O options)	1 second
0x43, 0x56, 0x8F-20	velocity (choose packet with I/O options)	1 second
0x46	health of receiver	5 seconds
0x4B	machinecode/status (includes antenna fault detect)	5 seconds
0x6D	all-in-view satellite selection, DOPs, Fix Mode	1 second
0x82	DGPS position fix mode (only in DGPS mode)	1 second

Customizing Receiver Operations

For information on customizing receiver operations, see the following tables on selecting report data.

Automatic Position and Velocity Reports

The receiver automatically outputs position and velocity reports at set intervals. Automatic report packets are controlled by Packet 35. Setting the control bits as indicated in the table below allows you to control which position and velocity packets are output.

Table A.2 Packet 35: Automatic Position and Velocity Reports Control Setting Bits

Report Packet ID	Description	Packet 0x35, Byte 0				Packet 0x35, Byte 1	
		Bit 0	Bit 1	Bit 4	Bit 5	Bit 0	Bit 1
0x42	single precision XYZ position	1		0			
0x83	double-precision XYZ position	1		1			
0x4A	single-precision LLA position		1 (default)	0			
0x84	double-precision LLA position		1	1			
0x43	velocity fix (XYZ, ECEF)					1	
0x56	velocity fix (ENU)						1(default)
0x8F-20	LLA and ENU				1		

Initialization Packets to Speed Start-up

If you are not supplying the receiver with battery power when main power is off, you can still “warm-start” the receiver by sending the following commands after the receiver has completed its internal initialization and has sent Packet 82.

Table A.3

Input Byte	Description
0x2B	initial position
0x2E	initial time
0x38	almanac (for each SV)
0x38	almanac health
0x38	ionosphere page
0x38	UTC correction

Packets Output at Power-Up

The following table lists the messages output by the receiver at power-up. After completing its self-diagnostics, the receiver automatically outputs a series of packets which indicate the initial operating condition of the receiver. Messages are output as listed in the table below. After Packet 82 is output, the sequence is complete and the receiver is ready to accept commands.

Table A.4 Packet Power-up Output Messages

Output ID	Description	Notes
0x45	software version	--
0x46	receiver health	--
0x4B	machine code/status	--
As chosen, see Table A.3 default: 0 x 4A, 0 x 56	position/Velocity output	As chosen, see Table A.3.
0x41	GPS time	
82	DGPS position fix mode	--

Timing Packets

If you are using the Lassen SQ GPS receiver as a timing system, you may need to implement the following TSIP control commands.

Table A.5 Timing Packet TSIP Control Commands

Input ID	Description	Output ID
0x21	get the current GPS time	0x41
0x38-05	request UTC parameters	0x58-05

Satellite Data Packets

The following packets contain a variety of GPS satellite data.

Table A.6 Satellite Data Packet Data I/O Descriptions

Input ID	Description	Output ID
0x27	request signal levels	0x47
0x38	request/load satellite system data	0x58
0x3C	request tracking status	0x5C

Backwards Compatibility

TSIP packets implemented in the Lassen SQ GPS receiver are backward compatible with those used in Lassen SK II GPS receiver. For information regarding compatibility with other Trimble receivers, contact the Trimble Technical Assistance Center.

Recommended TSIP Packets

Table A.7 Recommended TSIP Packet Data

Function	Description	Input	Output
Protocol and port setup	set/query port configuration	0xBC	0xBC
	set/query NMEA configuration	0x7A	0x7B
	set/query I/O options (autoreport and format options)	0x35	0x55
Navigation	GPS time	0x21	0x41
	position & velocity (superpacket)	0x8E-20 or 0x37 or auto	0x8F-20
	double-precision LLA	0x37/auto	0x84
	double-precision XYZ	0x37/auto	0x83
	ENU velocity	0x37/auto	0x56
	XYZ velocity	0x37/auto	0x43
Satellite and tracking information	query receiver state (health)	0x26	0x46, 0x4B
	query current satellite selection	0x24	0x6D
	query signal levels	0x27	0x47
	query satellite information (azimuth, elevation, etc.)	0x3C	0x5C
Receiver settings	query software version	0x1F	0x45
	query receiver ID & error status	0x26	0x4B, 0x46
	set/query receiver configuration	0xBB	0xBB
	set altitude for 2D mode	0x2A	0x4A
	disable PV/altitude filters	0x70	0x70
	set/query positioning mode (2D v. 3D)	0xBB	0xBB
GPS System	query/load GPS system data	0x38	0x58

Table A.7 (Continued)Recommended TSIP Packet Data

Function	Description	Input	Output
Initialization	full reset (clear battery backup and/or non-volatile settings)	0x1E	
	soft reset	0x25	
	set GPS time	0x2E	0x4E
	set exact LLA	0x32	
	set approx. XYZ	0x23	
	set approx. LLA	0x2B	
	set exact XYZ	0x31	

Command Packets Sent to the Receiver

The table below summarizes the command packets sent to the receiver. The table includes the input Packet ID, a short description of each packet, and the associated response packet. In some cases, the response packets depend on user-selected options. These selections are covered in the packet descriptions beginning on page 84.

Table A.8 User-Selected Command Packet Options

Input ID	Packet Description	Output ID
0x1E	clear battery back-up/reset	See Note 1
0x1F	software version	0x45
0x21	current time	0x41
0x23	initial position (XYZ ECEF)	--
0x24	request receiver position fix mode	0x6D
0x25	soft reset & self-test	See Note 1
0x26	receiver health	0x46, 0x4B
0x27	signal levels	0x47
0x2A	altitude for 2-D mode	0x4A

Table A.8 User-Selected Command Packet Options (Continued)

Input ID	Packet Description	Output ID
0x2B	initial position (Lat, Lon, Alt)	--
0x2D	oscillator offset	0x4D
0x2E	set GPS time	0x4E
0x31	accurate initial position (XYZ Cartesian ECEF)	--
0x32	accurate initial position	--
0x35	I/O options	0x55
0x37	status and values of last position and velocity	0x57
0x38	load or request satellite system data	0x58
0x3C	tracking status	0x5C, see Note 2
0x70	filter configuration	0x70
0x7A	set/request NMEA output configuration	0x7B
0xBB	set receiver configuration	0xBB
0xBC	set port configuration	0xBC
0x8E-20	last fix with extra information (fixed point)	0x8F-20
0x8E-26	Store settings in Flash memory.	0x8F-26

Note 1 – *Output is determined by packet 0 x 35. SeeTabl eA.4 to determine which messages are output at power-up.*

Note 2 – *No response sent if data is not available.*

Report Packets Sent by the Receiver to the User

The table below summarizes the packets output by the receiver. The response packets may depend on user-selected options. These selections are described on page84.

Table A.9 User-Selected Report Packet Options

Output ID	Packet Description	Input ID
0x41	GPS time	0x21, auto
0x42	single-precision XYZ position	0x37, auto
0x43	velocity fix (XYZ ECEF)	0x37, auto
0x45	software version information	0x1F, power-up
0x46	health of Receiver	0x26, auto, power-up
0x47	signal level for all satellites	0x27
0x4A	single-precision LLA position	0x37, auto
0x4B	machine code/status	0x26, auto, power-up
0x4D	oscillator offset	0x2D
0x4E	response to set GPS time	0x2E
0x55	I/O options	0x35
0x56	velocity fix (ENU)	0x37, auto
0x57	information about last computed fix	0x37
0x58	GPS system data/acknowledge	0x38
0x5C	satellite tracking status	0x3C
0x6D	all-in-view satellite selection	0x24, auto
0x82	differential position fix mode	0x62, auto
0x83	double-precision XYZ	auto, 0x37
0x84	double-precision LLA	auto, 0x37
0x8F-20	last fix with extra information (fixed point)	auto, 0x37, 0x8E-20
0xBB	GPS navigation configuration	0xBB
0xBC	Receiver port configuration	0xBC

Key Setup Parameters or Packet BB

Selecting the correct operating parameters has significant impact on receiver performance. Packet 0xBB (set receiver configuration) controls the key setup parameters.

The default operating parameters allow the receiver to perform well in almost any environment. The user can optimize the receiver to a particular application if the vehicle dynamics and expected level of obscuration are understood. If the receiver is then taken out of this environment, the specifically tuned receiver may not operate as well as a receiver with the default options.

The table below lists suggested parameter selections as a function of obscuration and whether accuracy or fix density is important. In this table, NA indicates that the operating parameter is not applicable, DC (don't care) indicates that the user may choose the operating parameter.

Table A.10 Setup Parameters in Packet 0xBB

Parameter	Accuracy	Fixes	Factory Default
Fix mode	Man 3D	AUTO	AUTO
Dynamics code	Land	Land	Land
Elevation mask	10°	5°	5°
Signal mask	6.0	4.0	3.0
DOP mask	6.0	12.0	12.0
DOP switch	NA	8.0	6.0
DGPS correction age	10 Seconds	N/A	30 Seconds

The default values in Table A.10 allow the receiver to operate well under the most varied and demanding conditions. A user may choose to change the default parameters if the receiver is only required to perform in a specific or limited environment. The user should be warned that when the receiver is exposed to operating conditions which are different from the conditions described by the user setup, then the performance may be degraded.

Initially, the user must consider the environment in which the receiver is expected to operate. There is a trade-off between how frequently a position fix is output versus the absolute accuracy of the fix. The user must decide which takes priority and then make the appropriate selections. This becomes increasingly important when frequent satellite blockages are expected, as in downtown “urban canyon” environments and heavily foliated areas.

Following is a description of the key fields in Packet 0xBB.

Set Fix Mode

Packet 0xBB is used to choose the appropriate position fix mode for your application: 2-D, 3-D or AUTO. The default mode is AUTO 2-D/3-D, where the receiver first attempts to obtain a 3-D solution with a PDOP below the DOP switch. If this is not possible, then the receiver attempts to obtain a 2-D solution with a DOP less than the DOP mask. This mode supplies fairly continuous position fixes even when there is frequent obscuration. This mode is preferable for most land or air applications, where altitude changes are occurring and there is occasional obscuration.

The highest accuracy fix mode is 3-D manual, where altitude is always calculated along with the latitude, longitude, and time. However, this requires four satellites with a PDOP below the DOP mask set in Packet BB in order to obtain a position. Normally, this will provide the most accurate solution. Thus, if only 3-D solutions are desired, then the user should request 3-D manual mode. Depending on how the PDOP mask is set, this may be restrictive when the receiver is subjected to frequent obscuration, or when the geometry is poor due to an incomplete constellation.

Alternatively, if the user only wants a 2-D solution, then 2-D manual should be requested. In this case, the receiver uses either the last altitude obtained in a 3-D fix, or the altitude supplied by the user. However, any error in the assumed altitude will affect the accuracy of the latitude and longitude solution.

When using the 2-D mode, expect fixes with accuracies which are at best as accurate as the supplied altitude. If a marine user enters sea-level as the altitude, then small errors in the horizontal solution will occur when the sea state is rough or there are high tidal variations. However, these errors may be smaller than the altitude errors induced by SA, so 2-D may be preferable for a marine user who does not want to observe “unusual” altitudes.

Dynamics Code

The feature default is LAND mode, where the receiver assumes a moderate dynamic environment. In this case, the satellite search and re-acquisition routines are optimized for vehicle type environments. In SEA mode, the search and re-acquisition routines assume a low acceleration environment and reverts to user entered altitude in 2-D auto. In AIR mode, the search and re-acquisition routines are optimized for high acceleration conditions.

Elevation Mask

This is the minimum elevation angle for satellites to be used in a solution output by the receiver. Satellites which are near the horizon are typically more difficult to track due to signal attenuation, and are also generally less accurate due to higher variability in the ionospheric and tropospheric corruption of the signal. When there are no obstructions, the receiver can generally track a satellite down to near the horizon. However, when this mask is set too low, the receiver may experience frequent constellation switching due to low elevation satellites being obscured.

Frequent constellation switching is undesirable because position jumps may be experienced when SA is present and DGPS is not available to remove these effects. The benefit of a low elevation mask is that more satellites are available for use in a solution and a better PDOP may be yielded. The current mask is set to five degrees and provides a reasonable trade-off of the benefits and drawbacks. High accuracy users may prefer a mask angle around ten degrees, where the ionosphere and troposphere begin to be more predictable

Signal Level Mask

This mask defines the minimum signal strength for a satellite used in a solution. There is some internal hysteresis on this threshold which allows brief excursions below the threshold if lock is maintained and the signal was previously above the mask. The factory default mask has been set to 3 (AMU). High accuracy users may use a slightly higher mask of 6.0-8.0, since weaker measurements may be slightly noisier and are often caused by reflected signals which provide erroneous ranges.

One should also resist the temptation to set the elevation and SNR masks too low. The satellite geometry is sometimes improved considerably by selecting low elevation satellites. They are, however, subject to significant signal degradation by the greater ionospheric and tropospheric attenuation that occurs. They are also subject to more obscuration by the passing scenery when the receiver is in a moving vehicle. The code phase data from those satellites is therefore more difficult to decode and therefore has more noise.

Note – *A level of hysteresis in the signal level mask is allowed in the core operating software. The hysteresis allows the receiver to continue using satellite signals which fall slightly below the mask and prevents the receiver from incorporating a new signal until the signal level slightly exceeds the mask. This feature minimizes constellation changes caused by temporary fluctuations in signal levels.*

DOP Mask and Switch

The DOP mask is the maximum DOP limit for any 2-D or 3-D position solution will be made. The DOP switch is the level at which the receiver stops attempting a 3-D solution, and tries for a 2-D solution when in automatic 2-D, 3-D mode. The switch level has no effect in either manual mode. Raising the DOP mask will generally increase the fix density during obscuration, but the fixes with the higher DOP will be less accurate (especially with SA present). Lowering the mask will improve the average accuracy at the risk of lowering the fix density.

Packet Descriptions

Command Packet 0x1E - Clear Battery Backup, then Reset

This packet commands the GPS receiver to clear all battery back-up data and to perform a software reset. This packet contains one data byte.

Table A.11 Command Packet 0x1E Format

Byte	Item	Type	Value	Definition
0	Reset mode	Unit 8	0x4B 0x46	Cold start: Erase BBRAM and restart Factory reset: Erase BBRAM and Flash and restart

Warning – All almanac, ephemeris, current position, mode, and communication port setup information is lost when executing the “Factory Reset” command. In normal use this packet should not be sent.

Warning – It is very helpful to keep a fresh copy of the current almanac, which is stored in the file GPSALM.DAT collected by the TSIPCHAT command “!”. This allows near-instantaneous recuperation by the receiver in case of power loss or clearing of battery-backed memory by using the TSIPCHAT command “@” to load it back into the receiver memory.

Command Packet 0x1F - Request Software Versions

This packet requests information about the version of software running in the Navigation and Signal Processors. This packet contains no data. The GPS receiver returns Packet 0x45.

Command Packet 0x21 - Request Current Time

This packet requests current GPS time. This packet contains no data. The GPS receiver returns Packet 0x41.

Command Packet 0x23 - Initial Position (XYZ Cartesian ECEF)

This packet provides the GPS receiver with an approximate initial position in XYZ coordinates. This packet is useful if the user has moved more than about 1,000 miles since the previous fix. (Note that the GPS receiver can initialize itself without any data from the user; this packet merely reduces the time required for initialization.) **This packet is ignored if the receiver is already calculating positions.** The data format is shown below.

Note – *To initialize using the Latitude-Longitude-Altitude representation, use Command Packet 0x2B.*

Table A.12 Command Packet 0x23 Data Format

Byte	Item	Type	Units
0-3	X	Single	Meters
4-7	Y	Single	Meters
8-11	Z	Single	Meters

Command Packet 0x24 - Request GPS Receiver Position Fix Mode

This packet requests current position fix mode of the GPS receiver. This packet contains no data. The GPS receiver returns Packet 0x6D.

Command Packet 0x25 - Initiate Soft Reset & Self Test

This packet commands the GPS receiver to perform a software reset. This is equivalent to cycling the power. The GPS receiver performs a self-test as part of the reset operation. This packet contains no data. Following completion of the reset, the receiver will output the start-up messages (see Table A.4). The GPS receiver sends Packet 0x45 only on power-up and reset (or on request); thus if Packet 0x45 appears unrequested, then either the GPS receiver power was cycled or the GPS receiver was reset.

Command Packet 0x26 - Request Health

This packet requests health and status information from the GPS receiver. This packet contains no data. The GPS receiver returns Packet 0x46 and 0x4B.

Command Packet 0x27 - Request Signal Levels

This packet requests signal levels for all satellites currently being tracked. This packet contains no data. The GPS receiver returns Packet 0x47.

Command Packet 0x2A - Altitude for 2-D Mode

Reference Altitude is the altitude used for manual 2-D positions if the altitude flag is set. Altitude is in units of HAE WGS-84 or MSL depending on the selected I/O options for the position (see page 92).

The Altitude Flag determines whether or not the Reference Altitude will be used. If set, it will be used. If cleared, altitude hold (last 3-D altitude) is used.

Note – *With no data byte, this packet requests the current values of these altitude parameters. In this case, the GPS receiver returns Packet 4A.*

This packet sets or requests the altitude parameters used for the Manual 2-D mode: Reference Altitude and Altitude Flag. Packet 0x4A (9 byte format) is returned.

Table A.13 Packet 0x2A Set Reference Altitude Description

Byte	Item	Type	Definition
0-3	Altitude	Single	Reference altitude for 2-D

Table A.14 Packet 0x2A Clear Reference Altitude Description

Byte	Item	Type	Value	Definition
0	Altitude Flag	UINT8	0 x FF	Clear Altitude flag

Note – *With no data bytes, this packet requests the current values of these altitude parameters. In this case, the GPS receiver returns Packet 4A (9 byte format).*

Command Packet 0x2B - Initial Position (Latitude, Longitude, Altitude)

This packet provides the GPS receiver with an approximate initial position in latitude and longitude coordinates (WGS-84). This packet is useful if the user has moved more than about 1,000 miles since the previous fix. (Note that the GPS receiver can initialize itself without any data from the user; this packet merely reduces the time required for initialization.) **This packet is ignored if the receiver is already calculating positions**. The data format is shown in the table below.

Table A.15 Command Packet 0x2B Data Format

Byte	Item	Type	Units
0-3	Latitude	Single	Radians, north
4-7	Longitude	Single	Radians, east
8-11	Altitude	Single	Meters

Note – *To initialize with ECEF position, use Command Packet 0x23.*

Command Packet 0x2D - Request Oscillator Offset

This packet requests the calculated offset of the GPS receiver master oscillator. This packet contains no data. The GPS receiver returns Packet 0x4D. This packet is used mainly for service. The permissible oscillator offset varies with the particular GPS receiver unit.

Command Packet 0x2E - Set GPS Time

This packet provides the approximate GPS time of week and the week number to the GPS receiver. The GPS receiver returns Packet 0x4E. The data format is shown below. The GPS week number reference is Week # 0 starting January 6, 1980. The seconds count begins at the midnight which begins each Sunday morning. This packet is usually not required when the battery back-up voltage is applied as the internal clock keeps time to sufficient accuracy. **This packet is ignored if the receiver has already calculated the time from tracking a GPS satellite**.

Note – *See report Packet 41 for information on the Extended GPS week number.*

Table A.16 Command Packet 0x2E Data Formats

Byte	Item	Type	Units
0-3	GPS time of week	Single	Seconds
4-5	Extended GPS week number	INT16	Weeks

Command Packet 0x31 - Accurate Initial Position (XYZ Cartesian ECEF)

This packet is identical in content to Packet 0x23. This packet provides an initial position to the GPS receiver in XYZ coordinates. However, the GPS receiver assumes the position provided in this packet to be accurate. This packet is used for satellite acquisition aiding in systems where another source of position is available. For acquisition aiding, the position provided by the user to the GPS receiver in this packet should be accurate to a few kilometers. For high-accuracy time transfer, position should be accurate to a few meters.

Table A.17 Command Packet 0x31 Data Format

Byte	Item	Type	Units
0-3	X-axis	Single	Meters
4-7	Y-axis	Single	Meters
8-11	Z-axis	Single	Meters

Command Packet 0x32 - Accurate Initial Position, (Latitude, Longitude, Altitude)

This packet is identical in content to Packet 0x2B. This packet provides the GPS receiver with an accurate initial position in latitude, longitude, and altitude coordinates. However, the GPS receiver assumes the position provided in this packet to be accurate. This packet is used for satellite acquisition aiding in systems where another source of position is available. For acquisition aiding, the position provided by the user to the GPS receiver in this packet should be accurate to a few kilometers. For high-accuracy time transfer, position should be accurate to a few meters.

Table A.18 Command Packet 0x32 Data Format

Byte	Item	Type	Units
0-3	Latitude	Single	Radians, North
4-7	Longitude	Single	Radians, East
8-11	Altitude	Single	Meters

Command Packet 0x35 - Set/Request I/O Options

This packet requests the current I/O options and allows the I/O options to be set as desired. To request the option settings without making any changes, send the packet with no data bytes. To change the option settings, include four data bytes with the values. The I/O options, their default settings, and the byte values for all possible configurations are shown below.

The Set/Request I/O options are stored in battery-backed memory. To store them in non-volatile RAM (Flash) use the 0x8E-26 command. The GPS receiver returns Packet 0x55. See Table A.4 for information on saving the settings to non-volatile memory. These abbreviations apply to the following table: ALT (Altitude), ECEF (Earth-centered, Earth-fixed), XYZ (Cartesian coordinates), LLA (latitude, longitude, altitude), HAE (height above ellipsoid), WGS-84 (Earth model (ellipsoid)), MSL geoid (mean sea level), and UTC (coordinated universal time).

Table A.19 Command Packets 0x35 and 0x55 Data Descriptions

Byte	Bit	Item	Type	Value	Definition	Associated Packets
Position						
0	0 (LSB)	XYZ ECEF	Bit	0 1	XYZ ECEF output off XYZ ECEF output on	0x42 (single) 0x83 (double)
	1	LLA Output	Bit	0 1	LLA output off LLA output on	0x4A (single) 0x84 (double)
	2	LLA ALT Output	Bit	0 1	HAE (Note 1) MSL geoid	0x4A / 0x84 0x8F-17 0x8F-18
	3	ALT input	Bit	0 1	HAE (Note 1) MSL geoid.	0x2A
	4	Precision-of-position output	Bit	0 1	Send single-precision packet. Send double-precision packet.	0x42, 0x4A 0x83, 0x84
	5	Super Packet Output	Bit	0 1	Output no Super Packets. Output all enabled Super Packets.	0x8F-20 (Note 2)
	6-7	reserved				
Velocity						
1	0	XYZ ECEF	Bit	0 1	XYZ ECEF output off XYZ ECEF output on	0x43
	1	ENU output	Bit	0 1	ENU output off ENU output on	0x56
	2-7	reserved				
Timing						
2	0	Time Type	Bit	0 1	GPS time UTC	0x42, 0x43, 0x4A, 0x83, 0x84, 0x56
	1-7	reserved				

Table A.19 Command Packets 0x35 and 0x55 Data Descriptions (Continued)

Byte	Bit	Item	Type	Value	Definition	Associated Packets
Auxiliary/Pseudo Range Measurements						
3	0	Raw measuring	Bit	0 1	Raw measurements off Raw measurements on	0x5A
	1	Raw / Filtered	Bit	0 1	Raw PR's in 5A Filtered PR's in 5A	0x5A (Note 3)
	2	reserved				
	3	Output dB Hz instead of AMU	Bit	0 1	Output dB Hz Output AMU's	0x5A, 0x5C, 0x47
	4-7	reserved				

Note 1 – *In the current version of the Lassen SQ GPS receiver, the output HAE altitude is always in the WGS-84 datum. The input HAE altitude is always in the WGS-84 datum.*

Note 2 – *Packet 8E must be used to specify which superpacket is to be output.*

Note 3 – *Automatic output of 0x5A messages is supported in the Lassen SQ GPS receiver for backwards compatibility with older TSIP applications.*

Command Packet 0x37 - Request Status and Values of Last Position and Velocity

This packet requests information regarding the last position fix and is only used when the receiver is not automatically outputting positions. The GPS receiver returns Report Packet 0x57 followed by the position/velocity packets specified in Command Packet 0x35.

Command Packet 0x38 - Request/Load Satellite System Data

This packet requests current satellite data (almanac, ephemeris, etc.) or permits loading initialization data from an external source (for example, by extracting initialization data from an operating GPS receiver unit via a data logger or computer and then using that data to initialize a second GPS receiver unit). The GPS receiver returns Packet 0x58. (Note that the GPS receiver can initialize itself without any data from the user; it merely requires more time.)

To request data without loading data, use only bytes 0 through 2; to load data, use all bytes. Before loading data, observe the caution notice below. The data formats are located in Report Packet 0x58.

Table A.20 Command Packet 0x38 Data Formats

Byte	Item	Type	Value	Definition
0	Operation	UINT8	1 2	Request data from Lassen SQ GPS receiver; Load data into Lassen SQ GPS receiver
1	Type of data	UINT8	2 3 4 5 6	Almanac Health page, T_oa, WN_oa Ionosphere UTC Ephemeris
2	Sat PRN#	UINT8	0 1 - 32	Data that is not satellite - ID specific Satellite PRN number
3	Length (n)	UINT8		Number of bytes of data to be loaded
4 to n+3	Data	UINT8		Satellite data

Warning – Proper structure of satellite data is critical to Lassen SQ GPS receiver operation. Requesting data is not hazardous; Loading data improperly is hazardous. Use this packet only with extreme caution. The data should not be modified in any way. It should only be retrieved and stored for later download. The ephemeris data cannot be loaded into the receiver.

Command Packet 0x3C - Request Current Satellite Tracking Status

This packet requests the current satellite tracking status. The GPS receiver returns Packet 0x5C if data is available.

Table A.21 Command Packet 0x3C Data Format

Byte	Item	Type	Value	Definition
0	Satellite #	UINT8	0 1 - 32	All satellites in the current tracking set desired satellite

Report Packet 0x41 - GPS Time

This packet provides the current GPS time of week and the week number. The GPS receiver sends this packet in response to Packet 0x21 and during an automatic packets update cycle. Update cycles occur approximately every 5 seconds. The data format is shown below.

Table A.22 Report Packet 0x41 Data Formats

Byte	Item	Type	Units
0-3	GPS time of week	Single	seconds
4-5	Extended GPS week number	INT16	weeks
6-9	GPS UTC offset	Single	seconds

Note – *UTC time lags behind GPS time by an integer number of seconds; UTC = (GPS time) - (GPS UTC offset).*

Warning – GPS week number runs from 0 to 1023 and then cycles back to week #0. week # 0 began January 6, 1980. The first cycle back to week #0 was on August 22, 1999. The extended GPS week number however, does not cycle back to 0. For example: the week # for August 22, 1999 = 1024; the Week # for April 1, 2002 = 1160.

The seconds count begins with “0” each Sunday morning at midnight GPS time. A negative indicated time-of-week indicates that time is not yet known; in that case, the packet is sent only on request. The following table shows the relationship between the information in Packet 0x41, and the Packet 0x46 status code.

Table A.23 Packets 0x41 and 0x46 Status Code Relationships

Approximate Time Accuracy	Time Source	Sign (TOW)	Packet 46 Status Code
none	no time at all	-	0x01
unknown	approximate time from real-time clock or Packet 2E	+	0x01
20-50 msec + clock drift	time from satellite	+	0x02 - 0x0C
full accuracy	time from GPS solution	+	0x00

Note – *Before using the GPS time from Packet 0x41, verify that the Packet 0x46 status code is 00 (“Doing position fixes”). This will ensure the most accurate GPS time.*

Report Packet 0x42 - Single-Precision Position Fix, XYZ ECEF

This packet provides current GPS position fix in XYZ ECEF coordinates. If the I/O “position” option is set to XYZ ECEF (byte 0: bit 0, Packet 0x35)and the I/O Precision-of-Position Output (byte 0: bit 4, Packet 0x35) is set to single-precision, then the GPS receiver sends this packet each time a fix is computed. The data format is shown below.

Table A.24 Report Packet 0x42 Data Formats

Byte	Item	Type	Units
0-3	X	Single	meters
4-7	Y	Single	meters
8-11	Z	Single	meters
12-15	Time-of-fix	Single	seconds

The time-of-fix is in GPS time or UTC as selected by the I/O “timing” option. Packet 83 provides a double-precision version of this information.

Report Packet 0x43 - Velocity Fix, XYZ ECEF

This packet provides current GPS velocity fix in XYZ ECEF coordinates. If the I/O velocity option is set to XYZ ECEF (byte 1, bit 0, Packet 0x35), then the GPS receiver sends this packet each time a fix is computed. The data format is shown below.

Table A.25 Report Packet 0x43 Data Formats

Byte	Item	Type	Units
0-3	X velocity	Single	meters/second
4-7	Y velocity	Single	meters/second
8-11	Z velocity	Single	meters/second
12-15	bias rate	Single	meters/second
16-19	time-of-fix	Single	seconds

The time-of-fix is in GPS time or UTC as selected by the I/O “timing” option (byte 2, bit 0, Packet 0x35).

Report Packet 0x45 - Software Version Information

This packet provides information about the version of software in the Navigation and Signal Processors. The GPS receiver sends this packet after power-on and in response to Packet 0x1F.

Table A.26 Report Packet 0x45 Data Formats

Byte	Item	Type
0	Major version number	UINT8
1	Minor version number	UINT8
2	Month	UINT8
3	Day	UINT8
4	Year number minus 1900	UINT8
5	Major revision number	UINT8
6	Minor revision number	UINT8
7	Month	UINT8
8	Day	UINT8
9	Year number minus 2000	UINT8

The first 5 bytes refer to the Navigation Processor and the second 5 bytes refer to the Signal Processor.

Report Packet 0x46 - Health of Receiver

This packet provides information about the satellite tracking status and the operational health of the receiver. The receiver sends this packet after power-on or software-initiated resets, in response to Packet 0x26 and, every five seconds. Packet 0x4B is always sent along with this packet.

Note – *If receiver status changes between five second outputs, no notification is given until the next cycle.*

Table A.27 Report Packet 0x46 Data Formats

Byte	Bit	Item	Type	Value	Definition
0		Status code	UINT8	0x00 0x01 0x02 0x03 0x08 0x09 0x0A 0x0B 0x0C	Doing position fixes Don't have GPS time yet Need initialization (0=normal, 1=shutdown due to RF initialization timeout) PDOP is too high No usable satellites Only 1 usable satellite Only 2 usable satellites Only 3 usable satellites The chosen satellite is unusable
1	0	Battery backup	Bit	0 1	OK BBRAM was not available at start-up
1	4	Antenna feedline fault		0 1	OK short

The error codes in Byte 1 of Packet 0x46 are encoded into individual bits within the byte. The bit positions are their meanings are shown below.

Note – *After status is detected, its bit remains set until the receiver is reset.*

Report Packet 0x47 - Signal Levels for all Satellites

This packet provides received signal levels for all satellites currently being tracked or on which tracking is being attempted (i.e., above the elevation mask and healthy according to the almanac). The receiver sends this packet only in response to Packet 0x27. The data format is shown below.

Table A.28 Report Packet 0x47 Data Formats

Byte	Item	Type
0	Count	UINT8
1	Satellite number 1	UINT8
2- 5	Signal level 1	Single
6	Satellite number 2	UINT8
7-10	Signal level 2	Single
(etc.)	(etc.)	(etc.)

Up to 8 satellite number/signal level pairs may be sent, indicated by the count field. Signal level is normally positive. If it is zero then that satellite has not yet been acquired. If it is negative then that satellite is not currently in lock. The absolute value of signal level field is the last known signal level of that satellite.

Note – *The signal level provided in this packet is a linear measurement of the signal strength after correlation or de-spreading. Units, either AMU or dBHz, are controlled by Packet 0x35.*

Report Packet 0x4A - 20 Byte Format

This packet provides current GPS position fix in LLA (latitude, longitude, and altitude) coordinates. If the I/O Position option is set to LLA and the I/O Precision-of-Position Output is set to single-precision (all controlled by Packet 35), then the receiver sends this packet each time a fix is computed. Command Packet 35 controls position output (XYZ or LLA) and (single or double) output precision. The data format is shown in below.

Table A.29 Report Packet 0x4A Data Formats

Byte	Item	Type	Units
0-3	Latitude	Single	radians; + for north, - for south
4-7	Longitude	Single	radians; + for east, - for west
8-11	Altitude	Single	meters (HAE or MSL)
2-15	Clock Bias	Single	meters
6-19	Time-of-Fix	Single	seconds (GPS or UTC)

The default altitude conversion is WGS-84. Altitude is referred to the datum ellipsoid or the MSL Geoid, depending on which I/O “LLA altitude” option is selected. The time-of-fix is in GPS time or UTC, depending on which I/O “timing” option is selected. This packet also is sent at start-up with a negative time-of-fix to report the current known position. Packet 0x84 provides a double-precision version of this information

Warning – When converting from radians to degrees, significant and readily visible errors will be introduced by use of an insufficiently precise approximation for the constant PI). The value of the constant PI as specified in ICD-GPS-200 is 3.1415926535898.

Report Packet 0x4A - 9 Byte Format

Report Packet 0x4A is also sent in response to the setting or requesting of the Reference Altitude Parameters using Command Packet 0x2A. These parameters can be used in the Manual 2-D mode.

Reference Altitude

The altitude used for manual 2-D positions if the altitude flag is set. Altitude is in units of HAE WGS-84 or MSL depending on the selected I/O options set for positioning with Command Packet 35.

Altitude Flag

A flag that determines whether or not the Reference Altitude will be used. If set, it will be used. If cleared, altitude hold (last 3-D altitude) will be used. The data format is shown in the following table.

Table A.30 Reference Altitude

Byte	Item	Type	Units
0-3	Reference Altitude	Single	Meters
4-7	Reserved	Single	
8	Altitude flag	UINT8	

Report Packet 0x4B - Machine/Code ID and Additional Status

The receiver transmits this packet in response to packets 0x25 and 0x26 and following a change in state. In conjunction with Packet 0x46, "health of receiver," this packet identifies the receiver and may present status messages. The machine ID can be used by equipment communicating with the receiver to determine the type of receiver to which the equipment is connected. Then the interpretation and use of packets can be adjusted accordingly.

Table A.31 Report Packet 0x4B Data Formats

Byte	Item	Type/	Definition
0	Machine ID	UINT8	Receiver dependent
1	Status 1	UINT8	See Table A.31
2	Status 2	UINT8	Bit 0 = Super packets supported

The status codes are encoded into individual bits within the bytes. The bit positions and their meanings are listed in the table below.

Table A.32 Report Packet 0x4B Bit Positions and Descriptions

Status 1 Bit Positions	Meaning if bit value = 1
0 (LSB)	Not used
1	Real-time Clock was not available at power-up.
2	Not used
3	The almanac stored in the receiver is not complete and current.
4-7	Not used

Report Packet 0x4D - Oscillator Offset

This packet provides the current value of the receiver master oscillator offset in Hertz at carrier. This packet contains one single precision number. The receiver sends this packet in response to Packet 0x2D. The permissible offset varies with the receiver unit.

Report Packet 0x4E - Response to Set GPS Time

Indicates whether the receiver accepted the time given in a Set GPS time packet. the receiver sends this packet in response to Packet 0x2E. This packet contains one byte.

Table A.33 Report Packet 0x4E Data Formats

Value	Meaning
ASCII “Y”	The receiver accepts the time entered via Packet 2E. The receiver has not yet received the time from a satellite.
ASCII “N”	The receiver does not accept the time entered via Packet 2E. The receiver has received the time from a satellite and uses that time. The receiver disregards the time in Packet 0x 2E.

Report Packet 0x55 - I/O Options

These abbreviations apply to the following table: ALT (Altitude), ECEF (Earth-centered, Earth-fixed), XYZ (Cartesian coordinates), LLA (latitude, longitude, altitude), HAE (height above ellipsoid), WGS-84 (Earth model (ellipsoid)), MSL geoid (Earth (mean sea level) mode), and UTC (coordinated universal time).

Table A.34 Command Packets 0x55 and 0x35 Data Descriptions

Byte	Bit	Item	Type	Value	Definition
Position					
0	0	XYZ ECEF	Bit	0 1	XYZ ECEF output off XYZ ECEF output on
0	1	LLA Output	Bit	0 1	LLA output off LLA output on
0	2	LLA ALT Output	Bit	0 1	HAE (WGS-84 datum) MSL geoid
0	3	ALT input	Bit	0 1	HAE (WGS-84 datum). MSL geoid
0	4	Precision-of-position output	Bit	0 1	Send single-precision packet. Send double-precision packet.
0	5	Super Packet Output	Bit	0 1	Output no Super Packets. Output all enabled Super Packets.
0	6-7	reserved			
Velocity					
1	0	XYZ ECEF	Bit	0 1	XYZ ECEF output off XYZ ECEF output on
1	1	ENU output	Bit	0 1	ENU output off ENU output on
1	2-7	reserved			

Table A.34 Command Packets 0x55 and 0x35 Data Descriptions (Continued)

Byte	Bit	Item	Type	Value	Definition
Timing					
2	0	Time Type	Bit	0 1	GPS time UTC
2	1-7	reserved			
Auxiliary / Range Measurements					
3	0	Raw measuring	Bit	0 1	Raw measurements off Raw measurements on
3	1	Raw / Filtered	Bit	0 1	Raw PR's in 0x5A Filtered PR's in 0x5A
3	2	reserved			
3	3	Output dB Hz instead of AMU	Bit	0 1	Output dB Hz Output AMU's
3	4-7	reserved			

Note 1 – *See the associated superpacket output, described later in this appendix. Packet 8E must be used to specify which superpacket is to be output.*

Note 2 – *Automatic output of 0x5A raw measurement messages is supported in the Lassen SQ GPS receiver for backwards compatibility with older TSIP applications.*

Report Packet 0x56 - Velocity Fix, East-North-Up (ENU)

If East-North-Up (ENU) coordinates have been selected for the I/O "velocity" option (see Packet 0x35), the receiver sends this packet under the following conditions:

- Each time that a fix is computed
- In response to Packet 0x37 (last known fix)

The data format is shown below.

Table A.35 Report Packet 0x56 Data Formats

Byte	Item	Type	Units
0-3	East Velocity	Single	m/s; + for east, - for west
4-7	North Velocity	Single	m/s; + for north, - for south
8-11	Up Velocity	Single	m/s; + for up, - for down
12-15	Clock Bias Rate	Single	m/s
16-19	Time-of-Fix	Single	seconds (GPS or UTC)

The time-of-fix is in GPS or UTC time as selected by the I/O "timing" option.

Report Packet 0x57 - Information About Last Computed Fix

This packet provides information concerning the time and origin of the previous position fix. The receiver sends this packet, among others, in response to Packet 0x37. The data format is shown below.

Table A.36 Report Packet 0x57 Data Formats

Byte	Item	Type	Units	Byte 0 Value/Velocity
0	Source of information	UINT8	--	00 temporary no fix 01 good current fix
1	Mfg. diagnostic	UINT8	--	
2-5	Time of last fix	Single	seconds, GPS time	
6-7	Week of last fix	INT16	weeks, GPS time	

Report Packet 0x58 - Satellite System Data/Acknowledge from Receiver

This packet provides GPS data (almanac, ephemeris, etc.). The receiver sends this packet in response to Packet 0x38 (acknowledges the loading of data). The data format is shown below.

Table A.37 Report Packet 0x58 Data Formats

Byte	Item	Type	Value	Definition
0	Operation	UINT8	1 2	Request data from receiver; Load data into receiver
1	Type of data	UINT8	2 3 4 5 6	Almanac Health page, T_oa, WN_oa Ionosphere UTC Ephemeris
2	Sat PRN#	UINT8	0 1 - 32	Data that is not satellite - ID specific satellite PRN number
3	Length (n)	UINT8		Number of bytes of data to be loaded
4 to n+3	Data			

Note – *If data is not available, byte 3 is set to 0 and "no" data is sent.*

The binary almanac, health page, and UTC data streams are similar to Report Packets 0x40, 0x49, and 0x4F respectively, but these reports are preferred. To get ionosphere or ephemeris, this report must be used.

Note – *Ephemeris cannot be loaded into Lassen SQ GPS receiver.*

Table A.38 Report Packet 0x58 Almanac Data

Byte	Item	Type	Definition / ICD-GPS-200
4	t_oa_raw	UINT8	Sec 20.3.3.5.1.2
5	SV_HEALTH	UINT8	Sec 20.3.3.5.1.2
6-9	e	Single	Sec 20.3.3.5.1.2
10-13	t_oa	Single	Sec 20.3.3.5.1.2

Table A.38 Report Packet 0x58 Almanac Data

14-17	i_o	Single	Sec 20.3.3.5.1.2
18-21	OMEGADOT	Single	Sec 20.3.3.5.1.2
22-25	sqrt_A	Single	Sec 20.3.3.5.1.2
26-29	OMEGA_0	Single	Sec 20.3.3.5.1.2
30-33	omega	Single	Sec 20.3.3.5.1.2
34-37	M_0	Single	Sec 20.3.3.5.1.2
38-41	a_f0	Single	Sec 20.3.3.5.1.2
42-45	a_f1	Single	Sec 20.3.3.5.1.2
46-49	Axis	Single	Sec 20.3.3.5.1.2
50-53	n	Single	Sec 20.3.3.5.1.2
54-57	OMEGA_n	Single	Sec 20.3.3.5.1.2
58-61	ODOT_n	Single	Sec 20.3.3.5.1.2
62-65	t_zc	Single	Sec 20.3.3.5.1.2. see Note 2.
66-67	weeknum	INT16	Sec 20.3.3.5.1.2
68-69	wn_oa	INT16	Sec 20.3.3.5.1.2

Note 1 – *All angles are in radians.*

Note 2 – *If data is not available, t_zc is set to -1.0.*

Table A.39 Report Packet 0x58 Almanac Health Data

Byte	Item	Type	Definition/ ICD-GPS-200
4	week # for health	UINT8	Sec 20.3.3.5.1.3
5-36	SV_health	UINT8	Sec 20.3.3.5.1.3
37	t_oa for health	UINT8	Sec 20.3.3.5.1.3
38	current t_oa	UINT8	units = seconds/2048
39-40	current week #	INT16	

Table A.40

Byte	Item	Type	Definition / IDC-GPS-200
4-11	---	---	not used
12-15	alpha_0	Single	Sec 20.3.3.5.1.9
16-19	alpha_1	Single	Sec 20.3.3.5.1.9
20-23	alpha_2	Single	Sec 20.3.3.5.1.9
24-27	alpha_3	Single	Sec 20.3.3.5.1.9
28-31	beta_0	Single	Sec 20.3.3.5.1.9
32-35	beta_1	Single	Sec 20.3.3.5.1.9
36-39	beta_2	Single	Sec 20.3.3.5.1.9
40-43	beta_3	Single	Sec 20.3.3.5.1.9

Table A.41

Byte	Item	Type	Definition / IDC-GPS-200
4-16	---	---	not used
17-24	A_0	Double	Sec 20.3.3.5.1.8
25-28	A_1	Single	Sec 20.3.3.5.1.8
29-30	delta_t_LS	Integer	Sec 20.3.3.5.1.8
31-34	t_ot	Single	Sec 20.3.3.5.1.8
35-36	WN t	Integer	Sec 20.3.3.5.1.8
37-38	WN_LSF	Integer	Sec 20.3.3.5.1.8
39-40	DN	Integer	Sec 20.3.3.5.1.8
41-42	delta_t_LSF	Integer	Sec 20.3.3.5.1.8

Table A.42

Byte	Item	Type	Definition / IDC -GPS-200
4	sv_number	UINT8	SV PRN number
5-8	t_ephem	Single	time of collection
9-10	weeknum	INT16	Sec 20.3.3.3, Table 20-I

Table A.42 (Continued)

Byte	Item	Type	Definition / IDC -GPS-200
11	codeL2	UINT8	Sec 20.3.3.3, Table 20-I
12	L2Pdata	UINT8	Sec 20.3.3.3, Table 20-I
13	SVacc_raw	UINT8	Sec 20.3.3.3, Table 20-I
14	SV_health	UINT8	Sec 20.3.3.3, Table 20-I
15-16	IODC	INT16	Sec 20.3.3.3, Table 20-I
17-20	T_GD	Single	Sec 20.3.3.3, Table 20-I
21-24	t_oc	Single	Sec 20.3.3.3, Table 20-I
25-28	a_f2	Single	Sec 20.3.3.3, Table 20-I
29-32	a_f1	Single	Sec 20.3.3.3, Table 20-I
33-36	a_f0	Single	Sec 20.3.3.3, Table 20-I
37-40	SVacc	Single	Sec 20.3.3.3, Table 20-I
41	IODE	UINT8	Sec 20.3.3.4
42	fit_interval	UINT8	Sec 20.3.3.4
43-46	C_rs	Single	Sec 20.3.3.4
47-50	delta_n	Single	Sec 20.3.3.4
51-58	M_0	Double	Sec 20.3.3.4
59-62	C_uc	Single	Sec 20.3.3.4, radians
63-70	e	Double	Sec 20.3.3.4
71-74	C_us	Single	Sec 20.3.3.4, radians
75-82	sqrt_A	Double	Sec 20.3.3.4
83-86	t_oe	Single	Sec 20.3.3.4
87-90	C_ic	Single	Sec 20.3.3.4, radians
91-98	OMEGA_0	Double	Sec 20.3.3.4
99-102	C_is	Single	Sec 20.3.3.4, radians
103-110	i_0	Double	Sec 20.3.3.4
111-114	C_rc	Single	Sec 20.3.3.4

Table A.42 (Continued)

Byte	Item	Type	Definition / IDC -GPS-200
115-122	omega	Double	Sec 20.3.3.4
123-126	OMEGADOT	Single	Sec 20.3.3.4
127-130	IDOT	Single	Sec 20.3.3.4
131-138	Axis	Double	= $(sqrt\_A)^2$
139-146	n	Double	derived from delta_n
147-154	r1me2	Double	= $sqrt(1.0-e^2)$
155-162	OMEGA_n	Double	derived from OMEGA_0, OMEGADOT
163-170	ODOT_n	Double	derived from OMEGADOT

Note – *All angles are in radians.*

Report Packet 0x5C - Satellite Tracking Status

This packet provides tracking status data for a specified satellite. Some of the information is very implementation-dependent and is provided mainly for diagnostic purposes. The receiver sends this packet in response to Packet 0x3C. The data format is shown below.

Table A.43 Report Packet 0x5C Data Format s

Byte	Bit	Item	Type	Value	Definition
0		Satellite PRN number	UINT8	number 1 - 32	
1	0-2	reserved	Bits	reserved	
1	3-5	Channel	Bits	0-7	
1	6-7	reserved	Bits	reserved	
2		Acquisition flag	UINT8	0 1 2	Never acquired Acquired Re-opened search

Table A.43 Report Packet 0x5C Data Formats (Continued)

Byte	Bit	Item	Type	Value	Definition
3		Ephemeris flag	UINT8	0 1	Flag not set Good ephemeris for this satellite (<4 hours old, good health)
4-7		Signal level	Single	Same as in Packet 0x47	
8-11		GPS time of last measurement	Single	<0 >0	No measurements have been taken. Center of the last measurement taken from this satellite.
12-15		Elevation	Singles	radians	Approximate elevation of this satellite above the horizon. Updated about every 15 sec.s. Used for searching and computing measurement correction factors.
16-19		Azimuth	Single	radians	Approximate azimuth from true north to this satellite. Updated typically about every 3 to 5 minutes. Used for computing measurement correction factors.
20-23		reserved	UINT8	0	

Report Packet 0x6D - All-In-View Satellite Selection

This packet provides a list of satellites used for position fixes by the GPS receiver. The packet also provides the PDOP, HDOP, and VDOP of that set and provides the current mode (automatic or manual, 3-D or 2-D). This packet has variable length equal to 16+nSVs where "nSVs" is the number of satellites used in the solution.

The GPS receiver sends this packet in response to Packet 0x24. The data format is shown below.

Table A.44 Report Packet 0x6D Data Formats

Byte	Bit	Item	Type	Value	Definition
0	0-2	Dimension	UINT8	3 4	2D 3D
0	3			0 1	Auto Manual
0	4-7			-	nSVs
1-4		PDOP	Single		PDOP
5-8		HDOP	Single		HDOP
9-12		VDOP	Single		VDOP
13-16		TDOP	Single		TDOP
(16+nSVvs)		SV PRN	UINT8		

Note – *The Lassen SQ GPS receiver sends this packet automatically after a position fix or every second if no position fix occurs.*

Command Packet 0x70 - Filter Control

Trimble OEM receivers have a number of filters. Command 0x70 provides control for these filters. It returns Report 0x70. There are three filters associated with 0x70:

- Position-Velocity (PV) Filter
- Static Filter
- Altitude Filter

The Position-Velocity (PV) Filter is the main filter and is used to "soften" the effect of constellation switches on position fixes. The filter has virtually no effect on velocity output and there is no lag due to vehicle dynamics. There may be a small increase in accuracy however.

A feature of the PV filter is the "Static Filter" which engages when the receiver is moving very slowly. This feature improves accuracy in the urban environment. The static filter should be turned off for the following applications:

- Slow-moving environments such as walking or drifting with the current
- When rooftop testing of receivers for moving applications

The altitude filter is a simple averaging filter with a time constant of a few seconds. It should be left on in marine and land applications.

To query for the current settings, use Command Packet 0x70 with no databytes. To input new settings, Command Packet 0x70 is sent with four data bytes.

Table A.45 Command Packet 70 Data Formats

Byte	Item	Type	Value	Definition
0	Position Velocity Filter	UINT8	0 1	Off On
1	Static Filter	UINT8	0 1	Off On
2	Altitude Filter	UINT8	0 1	Off On
3	Reserved	UINT8	reserved	

Report Packet 0x70

This report is sent as a response to Command Packet 0x70 as either a query or a set. It contains four bytes, as shown in Table A.46.

Command Packet 0x7A

The NMEA message determines whether or not a given NMEA message will be output. If the bit for a message is set, the message will be sent every “interval” second. Use the values shown below to determine the NMEA interval and message mask. While fixes are being generated, the output order is: ZDA, GGA, GLL, VTG, GSA, GSV, RMC.

Table A.46 Command Packet 0x7A and Report Packet 0x7B Data Formats

Byte	Bit	Item	Type	Value	Definition
0		Subcode	UINT8	0	
1		Interval	UINT8	1-255	Fix interval in seconds
2		Reserved	UINT8	0	
3		Reserved	UINT8	0	
4	0	RMC	Bit	0 1	Off On
4	1-7	Reserved	Bit	0	
5	0	GGA	Bit	0 1	Off On
5	1	GLL	Bit	0 1	Off On
5	2	VTG	Bit	0 1	Off On
5	3	GSV	Bit	0 1	Off On
5	4	GSA	Bit	0 1	Off On
5	5	ZDA	Bit	0 1	Off On
5	6-7	Reserved	Bit	0	

Report Packet 0x7B

This packet provides the NMEA settings and interval.

Report Packet 0x82 - Differential Position Fix Mode

This packet provides the differential position fix mode of the receiver. This packet contains only one data byte to specify the mode. The packet is sent in response to Packet 0x62 and whenever a satellite selection is made and the mode is Auto GPS / DGPS (modes 2 and 3). The receiver switches automatically between modes 2 and 3 based on the availability of differential corrections for a constellation which meets all other masks. If such a constellation is not available, then the receiver stays in its current automatic mode (2 or 3), and does not do position solutions.

Valid modes are:

Mode 0	Differential off (Manual GPS) — The receiver does position solutions without differential corrections, even if the differential corrections are available.
Mode 1	Differential on (Manual DGPS) — The receiver only does position solutions if valid differential correction data are available.
Mode 2	Differential currently off (Auto DGPS) — The receiver is not receiving differential correction data for all satellites in constellation which meets all other masks, and is doing non-differential position solutions.
Mode 3	Differential currently on (Auto DGPS) — The receiver is receiving differential correction data for all satellites in a constellation which meets all other masks, and is doing differential position solutions.

Note – *The Lassen SQ GPS receiver sends this packet automatically after every position fix except when in Mode 0.*

Report Packet 0x83 - Double-Precision XYZ Position Fix and Bias Information

This packet provides current GPS position fix in XYZ ECEF coordinates. If the I/O Position option is set to XYZ ECEF and the I/O Precision of Position option is set to Double (see Packet 0x35), the receiver sends this packet each time a fix is computed. The data format is shown below.

Table A.47 Report Packet 0x83 Data Formats

Byte	Item	Type	Units
0-7	X	Double	meters
8-15	Y	Double	meters
16-23	Z	Double	meters
24-31	clock bias	Double	meters
32-35	time-of-fix	Single	seconds

The time-of-fix is in GPS time or UTC, as selected by the I/O “timing” option.

Packet 42 provides a single-precision version of this information.

Report Packet 0x84 - Double-Precision LLA Position Fix and Bias Information

This packet provides current GPS position fix in LLA coordinates. If the I/O Position option is set to LLA and the Precision of Position option is set to Double (see Packet 0x35), the receiver sends this packet each time a fix is computed. The data format is shown below.

Table A.48 Report Packet 0x84 Data Formats

Byte	Item	Type	Units
0-7	latitude	Double	radians; + for north, - for south
8-15	longitude	Double	radians; + for east, - for west
16-23	altitude	Double	meters
24-31	clock bias	Double	meters
32-35	time-of-fix	Single	seconds

The time-of-fix is in GPS time or UTC, as selected by the I/O "timing" option.

Warning – When converting from radians to degrees, significant and readily visible errors will be introduced by use of an insufficiently precise approximation for the constant p (PI). The value of the constant PI as specified in ICD-GPS-200 is 3.1415926535898.

Packets 0x8E and 0x8F - Superpacket

See page 128 for information on Packets 0x8E and 0x8F.

Command Packet 0xBB - Navigation Configuration

In query mode, Packet 0xBB is sent with a single data byte and returns Report Packet 0xBB.

Note – *This Command Packet replaces Packets 0x2C, 0x62, 0x75, and 0x77.*

Table A.49 Command Packet 0xBB Query Mode Data Format

Byte #	Item	Type	Value	Definition	Default
0	Subcode	UINT8	0x00	Query mode	

TSIP Packet 0xBB is used to set GPS Processing options. The table below lists the individual fields within the 0xBB Packet. See Table A.4 for information on saving the settings to non-volatile memory.

Table A.50 Command and Report Packet 0xBB Field Descriptions

Byte #	Item	Type	Value	Definition	Default
0	Subcode	UINT8	0x00	Query mode	0x03
1	Operating Dimension	UINT8	0 3 4	Automatic (2D/3D) Horizontal (2D) Full Position (3D)	Automatic
2	DGPS Mode	UINT8	0 1 2 or 3	DGPS off DGPS only DGPS auto	DGPS auto
3	Dynamics Code	UINT8	1 2 3 4	Land Sea Air Stationary	Land
4	reserved				
5-8	Elevation Mask	Single	0.0 - 1.57 (radian)	Lowest satellite elevation for fixes	0.0873 (5^{O})
9-12	AMU Mask	Single	0-25 (AMU)	Minimum signal level for fixes	2.0
13-16	DOP Mask	Single	0.2-100	Maximum DOP for fixes	12.0

Table A.50 Command and Report Packet 0xBB Field Descriptions (Continued)

Byte #	Item	Type	Value	Definition	Default
17-20	DOP Switch	Single	0.2-100	Selects 2D/3D mode	5.0
21	DGPS Age Limit	UINT8	2-90 (seconds)	Maximum time to use a DGPS correction (seconds)	30
22-39	reserved				

Command Packet 0xBC - Protocol Configuration

TSIP Packet 0xBC is used to query the port characteristics. In query mode, Packet 0xBC is sent with a single data byte and returns Report Packet 0xBC. (See Table A.4 for information on saving the settings to non-volatile memory.)

TSIP Packet 0xBC is used to set the communication parameters on Port 1. The table below lists the individual fields within the Packet 0xBC and provides query field descriptions.

The BC command settings are retained in battery-backed RAM.

Table A.51 Command Packet 0xBC Port Characteristics

Byte	Bit	Item	Type	Value	Definition
0		Port to Set	UINT8	0 1 0xFF	Port 1 Port 2 Current port
1		Input Baud Rate	UINT8	2 3 4 5 6 7 8 9	reserved reserved reserved reserved 4800 baud 9600 baud 19200 baud 38400 baud
2		Output Baud Rate	UINT8	As above	As above (Note 1)

Table A.51 Command Packet 0xBC Port Characteristics (Continued)

Byte	Bit	Item	Type	Value	Definition
3		# Data Bits	UINT8	2 3	7 bits 8 bits
4		Parity	UINT8	0 1 2	None Odd Even
5		# Stop Bits	UINT8	0 1	1 bit 2 bits
6		Flow Control	UINT8	0	0 = none
7	0	reserved	Bit	0	
	1	TSIP input	Bit	0 1	off on
	2	reserved	Bit	0	
	3	reserved	Bit	0	
	4-7	reserved	Bit	0	
8	0	reserved	Bit	0	
	1	TSIP output	Bit	0 1	off on
	2	NMEA output	Bit	0 1	off on
	3-7	reserved	UINT8	0	
9		reserved	UINT8	0	

Note 1 – *The Lassen SQ GPS receiver requires that the input and output baud rates be identical.*

Warning – TSIP input or output must have 8 databits (byte 3).

TSIP Superpackets

Several packets have been added to the core TSIP protocol to provide additional capability for OEM receivers. In OEM Packets 0x8E and their 0x8F responses, the first data byte is a sub-code which indicates the superpacket type. For example, in Packet 0x8E-15, 15 is the sub-code that indicates the superpacket type. Therefore the ID code for OEM packets is 2 bytes long followed by the data.

Command Packet 0x8E-20 - Request Last Fix with Extra Information

This packet requests Packet 0x8F-20 or marks it for automatic output. If only the first byte (20) is sent, an 0x8F-20 report containing the last available fix will be sent immediately. If two bytes are sent, the packet is marked/unmarked for auto report according to the value of the second byte as shown in below. 0x37 can also be used for requesting 0x8F-20 if the 0x8F-20 is scheduled for auto output.

Table A.52 Command Packet 0x8E-20 Field Descriptions

Byte	Item	Type	Definition
0	Sub-packet id	UINT8	0x20
1	Mark for Auto-report (See Packet 35 byte 0 bit 5)	UINT8	0 = do not auto-report 1 = mark for auto-report

Note – *Auto-report requires that superpacket output is enabled. Refer to Command Packet 35.*

Report Packet 0x8F-20 - Last Fix with Extra Information (binary fixed point)

This packet provides complete information about the current position velocity fix in a compact, fixed-length 56-byte packet. The fields are fixed-point with precision matched to the receiver accuracy. It can be used for automatic position/velocity reports. The latest fix can also be requested by 0x8E-20 or 0x37 commands.The data format is shown below.

Table A.53 Report Packet 0x8F-20 Data Formats

Byte	Bit	Item	Type	Value	Definition
0		Sub-packet id	UINT8		Id for this sub-packet (always 0x20)
1		KeyByte	UINT8		Reserved for Trimble DGPS Post-processing.
2-3		east velocity	INT16		0.005 m/s or 0.020 m/s See Note 1.
4-5		north velocity	INT16		0.005 m/s or 0.020 m/s See Note 1.
6-7		up velocity	INT16		0.005 m/s or 0.020 m/s See Note 1.
8-11		Time Of Week	UINT32		GPS Time in milliseconds
12-15		Latitude	INT32	-2^{30} to 2^{30}	WGS-84 latitude, 2^{-31} semicircle (-90° - 90°)
16-19		Longitude	UINT32	0 to 2^{32}	WGS-84 latitude, 2^{-31} semicircle (0° - 360°)
20-23		Altitude	UINT32		Altitude above WGS-84 ellipsoid, mm.
24	0	Velocity Scaling		0 1	0.005 m/s^2 0.020 m/s^2
	1-7	reserved			
25		reserved			
26		Datum			Datum index + 1 0=unknown

Table A.53 Report Packet 0x8F-20 Data Formats (Continued)

Byte	Bit	Item	Type	Value	Definition
27	0	Fix Available	Bit	0 1	Yes No
	1	DGPS Corrected	Bit	0 1	No Yes
	2	Fix Dimension	Bit	0 1	3D 2D
	3	Alt Hold	Bit	0 1	Last 3D Altitude User-entered altitude
	4	Filtered	Bit	0 1	Unfiltered Filtered
	5-7	reserved			
28		NumSVs	UINT8		Number of satellites used for fix. Will be zero if no fix was available.
29		UTC Offset	UINT8		Number of leap seconds between UTC time and GPS time.
30-31		Week	INT16		GPS time of fix, weeks.
32	0-5	PRN 1	UINT8	1-32	PRN of first satellite
	6-7	reserved			
33		IODE 1	UINT8		IODE of first satellite
34	0-5	PRN 2	UINT8	1-32	PRN of second satellite
	6-7	reserved			
35		IODE 2	UINT8		IODE of second satellite
36	0-5	PRN 3	UINT8	1-32	PRN of third satellite
	6-7	reserved			
37		IODE 3	UINT8		IODE of third satellite
38	0-5	PRN 4	UINT8	1-32	PRN of fourth satellite
	6-7	reserved			

Table A.53 Report Packet 0x8F-20 Data Formats (Continued)

Byte	Bit	Item	Type	Value	Definition
39		IODE 4	UINT8		IODE of fourth satellite
40	0-5	PRN 5	UINT8	1-32	PRN of fifth satellite
	6-7	reserved			
41		IODE 5	UINT8		IODE of fifth satellite
42	0-5	PRN 6	UINT8	1-32	PRN of sixth satellite
	6-7	reserved			
43		IODE 6	UINT8		IODE of sixth satellite
44	0-5	PRN 7	UINT8	1-32	PRN of seventh satellite
	6-7	reserved			
45		IODE 7	UINT8		IODE of seventh satellite
46	0-5	PRN 8	UINT8	1-32	PRN of eighth satellite
	6-7	reserved			
47		IODE 8	UINT8		IODE of eighth satellite
48-55		Ionospheric Parameters			

Note – *Velocity scale controlled by byte 24, bit 1. Overflow = 0x8000.*

Command Packet 0x8E-26 - Non-Volatile Memory Storage

The 0x8E-26 command is issued with no data to cause the current settings to be saved to non-volatile memory. The 0x8F-26 report is generated after the values have been saved. (See Chapter 3, Software Interface, for information on the settings that can be saved to non-volatile memory.)

Table A.54 Command Packet 0x8E-26 Definitions

Byte #	Item	Type	Value	Definition
0	Subcode	UINT8	0x26	Save Settings

Report Packet 0x8F-26 - Non-Volatile Memory Status

This report will be issued after an 0x8E-26 command.

Table A.55 Report Packet 0x8F-26 Field Descriptions

Byte/	Item	Type	Value	Definition
0	Subcode	UINT8	0x26	Save Settings
1-4	reserved			

APPENDIX

B

TSIP Tool kit User's Guide

The GPS Tool Kit program disk includes several TSIP interface programs designed to help developer's evaluate and integrate the GPS module and create GPS applications. These programs run on a PC Win95, Win98, Win2000, or WinNT platform. They are intended as a base upon which to build application specific software, so the source code in ANSI C is included for many of these programs. The GPS Tool Kit program disk includes the following programs:

32 bit Windows Applications

SQ_Monitor: is a 32 bit windows application which interfaces with a TSIP-speaking receiver through a serial port. The program accepts TSIP reports and displays them in a window with fields for position, velocity, time, receiver status and satellite track status. It allows the user to exercise some basic TSIP commands. SQ_Monitor can also log TSIP report in binary format for later translation.

DOS Programs and C Source Code

TSIPCHAT.EXE: is a 16 bit DOS-based application which interfaces with the Lassen SQ GPS receiver through a serial port. It allows the user to exercise TSIP commands and may be used to view NMEA output. TSIPCHAT can log a TSIP report in binary format. Source code is provided.

TSIPPRNT.EXE: is a 16 bit DOS-based application which interprets a binary TSIP data stream, such as logged by TSIPCHAT or SQ_Monitor, and prints it to a file. Source code is provided.

For further information on the above programs, see the README file on the Toolkit disk.

A 32-bit Windows Application, TSIP Reader, is available from the FTP web site, ftp://ftp.trimble.com/pub/set/embedded/bin, that interprets binary TSIP data streams.

SQ_Monitor

SQ_Monitor requires Win95, Win98, Win2000, or WinNT. Once the program is started, it immediately prompts for the serial port connected to the GPS receiver.

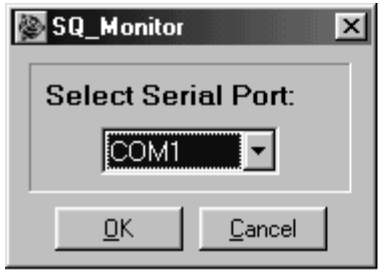


Figure B.1 SQ_Monitor - Serial Port Selection

The main window is displayed once the GPS receiver is communicating with the application. Fields with question marks "?" indicate that information is not yet available.

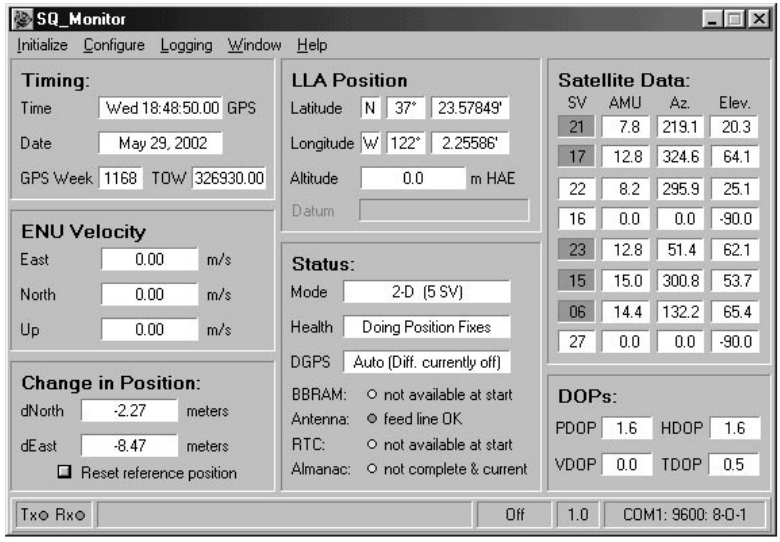


Figure B.2 SQ_Monitor - Main Window

Delta Position

SQ_Monitor displays delta position which is the difference between the current position and the initial position. The initial position can be reset at anytime by selecting [Reset Initial Position], (see).

File Storage

SQ_Monitor provides for file storage of the raw binary TSIP stream directly from the serial port. To turn data collection on or off, use the record pull-down menu. The recorded binary data stream can be translated into an ASCII file with the program TSIPRNT or TSIPReader.

APPENDIX

C

NMEA 0183

This appendix provides a brief overview of the NMEA 0183 protocol, and describes both the standard and optional messages offered by the Lassen SQ GPS receiver.

NMEA 0183 is a simple, yet comprehensive ASCII protocol which defines both the communication interface and the data format. The NMEA 0183 protocol was originally established to allow marine navigation equipment to share information. Since it is a well established industry standard, NMEA 0183 has also gained popularity for use in applications other than marine electronics. The latest release of NMEA 0183 is Version 3.0 (July 1, 2000). Trimble Navigation supports both version 2.1 and version 3.0. The primary change in release 3.0 is the addition of the mode indicators in the GLL, RMC, and VTG messages.

For those applications requiring output only from the GPS receiver, NMEA 0183 is a popular choice since, in many cases, an NMEA 0183 software application code already exists. The Lassen SQ GPS receiver is available with firmware that supports a subset of the NMEA 0183 messages: GGA, GLL, GSA, GSV, RMC, VTC, and ZDA. For a nominal fee, Trimble can offer custom firmware with a different selection of messages to meet your application requirements.

For a complete copy of the NMEA 0183 standard, contact:

NMEA National Office
PO Box 3435
New Bern, NC 28564-3435
U.S.A.
Telephone: +1-919-638-2626
Fax: +1-919-638-4885

The NMEA 0183 Communication Interface

NMEA 0183 allows a single source (talker) to transmit serial data over a single twisted wire pair to one or more receivers (listeners). The table below lists the standard characteristics of the NMEA 0183 data transmissions.

Table C.1 NMEA 0183 Standard Characteristics

Signal Characteristic	NMEA Standard
Baud Rate	4800
Data Bits	8
Parity	None (Disabled)
Stop Bits	1

NMEA 0183 Message Format

The NMEA 0183 protocol covers a broad array of navigation data. This broad array of information is separated into discrete messages which convey a specific set of information. The entire protocol encompasses over 50 messages, but only a sub-set of these messages apply to a GPS receiver like the Lassen SQ GPS receiver. The NMEA message structure is described below.

```
$IDMSG,D1,D2,D3,D4,.......,Dn*CS[CR][LF]
```

"$"	The "$" signifies the start of a message.
ID	The talker identification is a two letter mnemonic which describes the source of the navigation information. The GP identification signifies a GPS source.
MSG	The message identification is a three letter mnemonic which describes the message content and the number and order of the data fields.
","	Commas serve as delimiters for the data fields.
Dn	Each message contains multiple data fields (Dn) which are delimited by commas.
"*"	The asterisk serves as a checksum delimiter.
CS	The checksum field contains two ASCII characters which indicate the hexadecimal value of the checksum.
[CR][LF]	The carriage return [CR] and line feed [LF] combination terminate the message.

NMEA 0183 messages vary in length, but each message is limited to 79 characters or less. This length limitation excludes the "$" and the [CR][LF]. The data field block, including delimiters, is limited to 74 characters or less.

Field Definitions

Many of the NMEA date fields are of variable length, and the user should always use the comma delineators to parse the NMEA message date field. Table C.2 specifies the definitions of all field types in the NMEA messages supported by Trimble.

Table C.2 Field Type Summary

Type	Symbol	Definition
Status	A	Single character field: A=Yes, data valid, warning flag clear V=No, data invalid, warning flag set
Special Format Fields		
Latitude	llll.lll	Fixed/variable length field: Degreesminutes.decimal-2 fixed digits of degrees, 2 fixed digits of minutes and a variable number of digits for decimal-fraction of minutes. Leading zeros always included for degrees and minutes to maintain fixed length. The decimal point and associated decimal-fraction are optional if full resolution is not required.
Longitude	yyyyy.yyy	Fixed/Variable length field: Degreesminutes.decimal-3 fixed digits of degrees, 2 fixed digits of minutes and a variable number of digits for decimal-fraction of minutes. Leading zeros always included for degrees and minutes to maintain fixed length. The decimal point and associated decimal-fraction are optional if full resolution is not required.
Time	hhmmss.ss	Fixed/Variable length field: hoursminutesseconds.decimal-2 fixed digits of minutes, 2 fixed digits of seconds and a variable number of digits for decimal-fraction of seconds. Leading zeros always included for hours, minutes, and seconds to maintain fixed length. The decimal point and associated decimal-fraction are optional if full resolution is not required.

Table C.2 Field Type Summary (Continued)

Type	Symbol	Definition
Defined		Some fields are specified to contain pre-defined constants, most often alpha characters. Such a field is indicated in this standard by the presence of one or more valid characters. Excluded from the list of allowable characters are the following that are used to indicated field types within this standard: “A”, “a”, “c”, “hh”, “hhmmss.ss”, “llll.ll”, “x”, “yyyyy.yy”
Numeric Value Fields		
Variable	x.x	Variable length integer or floating numeric field. Optional leading and trailing zeros. The decimal point and associated decimal-fraction are optional if full resolution is not required (example: 73.10=73.1=073.1=73).
Fixed HEX	hh	Fixed length HEX numbers only, MSB on the left
Information Fields		
Fixed Alpha	aa	Fixed length field of upper-case or lower-case alpha characters
Fixed number	xx	Fixed length field of numeric characters

Note – *Spaces are only be used in variable text fields.*

Note 2 – *Units of measure fields are appropriate characters from the Symbol column (see Table C.2), unless a specified unit of measure is indicated.*

Note 3 – *Fixed length field definitions show the actual number of characters. For example, a field defined to have a fixed length of 5 HEX characters is represented as hhhhh between delimiters in a sentence definition.*

NMEA 0183 Message Options

The Lassen SQ GPS receiver can output any or all of the messages listed in Table C.3. In its default configuration (as shipped from the factory), the Lassen SQ GPS receiver outputs two messages: GGA and VTG. These messages are output at a 1 second interval with the “GP” talker ID and checksums. These messages are output at all times during operation, with or without a fix. If a different set of messages ha been selected (using Packet 0x7A), and this setting has been stored in Flash memory (using Packet 0x8E-26), the default messages are permanently replaced until the receiver is returned to the factory default settings.

Note – *The user can configure a custom mix of the messages listed in Table C.3. See Chapter 3, and TSIP command packets 0xBC, 0x7A, and 8E-26 in Appendix A for details on configuring NMEA output.*

Warning – If too many messages are specified for output, you may need to increase the unit’s baud rate.

Table C.3 Lassen SQ GPS Receiver NMEA Messages

	Message	Description
Default Output	GGA	GPS fix data
	GLL	Geographic position - Latitude/Longitude
	GSA	GPS DOP and active satellites
	GSV	GPS satellites in view
	RMC	Recommended minimum specific GPS/Transit data
Default Output	VTG	Track made good and ground speed
	ZDA	Time & Date

The format for each message in Table C.3 is described in more detail in the next section.

NMEA 0183 Message Formats

GGA - GPS Fix Data

The GGA message includes time, position and fix related data for the GPS receiver.

```
$GPGGA,hhmmss.ss,llll.lll,a,nnnnn.nnn,b,t,uu,
v.v,w.w,M,x.x,M,y.y,zzzz*hh <CR><LF>
```

Table C.4 GGA - GPS Fix Data Message Parameters

Field #	Description
1	UTC of Position
2,3	Latitude, N (North) or S (South)
4,5	Longitude, E (East) or W (West)
6	GPS Quality Indicator: 0 = No GPS, 1 = GPS, 2 = DGPS
7	Number of Satellites in Use
8	Horizontal Dilution of Precision (HDOP)
9, 10	Antenna Altitude in Meters, M = Meters
11, 12	Geoidal Separation in Meters, M=Meters. Geoidal separation is the difference between the WGS-84 earth ellipsoid and mean-sea-level.
13	Age of Differential GPS Data. Time in seconds since the lastType 1 or 9 Update
14	Differential Reference Station ID (0000 to 1023)
hh	Checksum

GLL - Geographic Position - Latitude/Longitude

The GLL message contains the latitude and longitude of the present vessel position, the time of the position fix and the status.

```
$GPGLL,llll.lll,a,yyyyy.yyy,a,hhmmss.ss,A,i*hh<CR>
<LF>
```

Table C.5 GLL - Geographic Position - Latitude / Longitude Message Parameters

Field #	Description
1,2	Latitude, N (North) or S (South)
3,4	Longitude, E (East) or W (West)
5	UTC of position
6	Status: A = Valid, V= Invalid
7	Mode Indicator A=Autonomous Mode D=Differential Mode E=Estimated (dead reckoning) Mode M=Manual Input Mode S=Simulated Mode N-Data Not Valid
hh	Checksum

GSA - GPS DOP and Active Satellites

The GSA messages indicates the GPS receiver's operating mode and lists the satellites used for navigation and the DOP values of the position solution.

```
$GPGSA,a,x,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,xx,
xx,x.x,x.x,x.x*hh<CR><LF>
```

Table C.6 GSA - GPS DOP and Active Satellites Message Parameters

Field #	Description
1	Mode: M = Manual, A = Automatic. In manual mode, the receiver is forced to operate in either 2D or 3D mode. In automatic mode, the receiver is allowed to switch between 2D and 3D modes subject to the PDOP and satellite masks.
2	Current Mode: 1 = fix not available, 2 = 2D, 3 = 3D
3 to 14	PRN numbers of the satellites used in the position solution. When less than 12 satellites are used, the unused fields are null
15	Position dilution of precision (PDOP)
16	Horizontal dilution of precision (HDOP)
17	Vertical dilution of precision (VDOP)
hh	Checksum

GSV - GPS Satellites in View

The GSV message identifies the GPS satellites in view, including their PRN number, elevation, azimuth and SNR value. Each message contains data for four satellites. Second and third messages are sent when more than 4 satellites are in view. Fields #1 and #2 indicate the total number of messages being sent and the number of each message respectively.

```
$GPGSV,x,x,xx,xx,xx,xxx,xx,xx,xx,xxx,xx,xx,xx,
xxx,xx,xx,xx,xxx,xx*hh<CR><LF>
```

Table C.7 GSV - GPS Satellites in View Message Parameters

Field #	Description
1	Total number of GSV messages
2	Message number: 1 to 3
3	Total number of satellites in view
4	Satellite PRN number
5	Satellite elevation in degrees (90° Maximum)
6	Satellite azimuth in degrees true (000 to 359)
7	Satellite SNR (C/No), null when not tracking
8,9,10,11	PRN, elevation, azimuth and SNR for second satellite
12,13,14,15	PRN, elevation, azimuth and SNR for third satellite
16,17,18,19	PRN, elevation, azimuth and SNR for fourth satellite
hh	Checksum

RMC - Recommended Minimum Specific GPS/Transit Data

The RMC message contains the time, date, position, course, and speed data provided by the GPS navigation receiver. A checksum is mandatory for this message and the transmission interval may not exceed 2 seconds. All data fields must be provided unless the data is temporarily unavailable. Null fields may be used when data is temporarily unavailable.

```
$GPRMC,hhmmss.ss,A,llll.ll,a,yyyyy.yy,a,
x.x,x.x,xxxxxx,x.x,a,i*hh<CR><LF>
```

Table C.8 RMC - Recommended Minimum Specific GPS / Transit Data Message Parameters

Field #	Description
1	UTC of Position Fix.
2	Status: A = Valid, V = navigation receiver warning
3,4	Latitude, N (North) or S (South).
5,6	Longitude, E (East) or W (West).
7	Speed over the ground (SOG) in knots
8	Track made good in degrees true.
9	Date: dd/mm/yy
10,11	Magnetic variation in degrees, E = East / W= West
12	Position System Mode Indicator; A=Autonomous, D=Differential, E=Estimated (Dead Reckoning), M=Manual Input, S=Simulation Mode, N=Data Not Valid
hh	Checksum (Mandatory for RMC)

VTG - Track Made Good and Ground Speed

The VTG message conveys the actual track made good (COG) and the speed relative to the ground (SOG).

```
$GPVTG,x.x,T,x.x,M,x.x,N,x.x,K,i*hh<CR><LF>
```

Table C.9 VTG - Track Made Good and Ground Speed Message Parameters

Field #	Description
1	Track made good in degrees true.
2	Track made good in degrees magnetic.
3,4	Speed over the ground (SOG) in knots.
5,6	Speed over the ground (SOG) in kilometer per hour.
7	Mode Indicator: A=Autonomous Mode, D=Differential Mode, E=Estimated (dead reckoning) Mode, M=Manual Input Mode, S=Simulated Mode, N-Data Not Valid
hh	Checksum

ZDA - Time & Date

The ZDA message contains UTC, the day, the month, the year and the local time zone.

```
$GPZDA,hhmmss.ss,xx,xx,xxxx,,*hh<CR><LF>
```

Table C.10 ZDA - Time & Date Message Parameters

Field #	Description
1	UTC
2	Day (01 to 31)
3	Month (01 to 12)
4	Year
5	unused
6	unused
hh	Checksum

Note – *Fields #5 and #6 are null fields in the Lassen SQ GPS receiver output. A GPS receiver cannot independently identify the local time zone offsets.*

Warning – If UTC offset is not available, time output will be in GPS time until the UTC offset value is collected from the GPS satellites. When the offset becomes available, the time will jump to UTC time.

Note – *GPS time can be used as a timetag for the 1PPS. The ZDA message comes out 100-500 msec after the PPS.*

Exception Behavior

When no position fix is available, some of the data fields in the NMEA messages will be blank. A blank field has no characters between the commas. There are three general cases when no fix is available: at power-up without BBRAM (cold start); at power-up with BBRAM (warm start); and when the GPS signal is temporarily blocked. These three cases have different NMEA output behavior in the Lassen SQ GPS receiver. This section describes the behavior for the current product. The specification for this behavior may change in future products.

Power-up with No BBRAM

In this case, no previous fix is available in battery-backed memory. If the output message list and output rate has been customized (using TSIP command packet 0x7A) and stored in Flash memory, then at power-up the receiver will output the messages according to the customized setting. Otherwise, GGA messages are output every second. Before fixes are available, the message fields will be empty.

Power-up with BBRAM

In this case, a previous fix is available in battery-backed memory at power-up. If the output message list and output rate has been customized (using TSIP command packet 0x7A) and stored in Flash memory, then at power-up the receiver will output the messages according to the customized setting. Otherwise, GGA messages are output every second. Before fixes are available, the message fields will be empty except for the Time field, assuming the back-up battery power is present so that time can be tracked continuously by the RTC (Real Time Clock).

Interruption of GPS Signal

If the GPS signal is interrupted temporarily, the NMEA will continue to be output according to the user-specified message list and output rate. Position and velocity fields will be blank until the next fix, but most other fields will be filled.

APPENDIX

D

Specifications and Mechanical Drawings

The Lassen SQ GPS receiver is designed for embedded industrial computing or control, mobile computing or data collection, precision timing, and vehicle tracking applications. This appendix includes the system specifications and mechanical drawings for the Lassen SQ GPS receiver module and the available GPS antennas.

Lassen SQ GPS Receiver Specifications

Performance

General	L1 frequency (1575.42 MHz), C/A code (Standard Positioning Service), 8-channel, continuous tracking receiver, 32 correlators
Update Rate	TSIP @ 1 Hz; NMEA @ 1 Hz
Accuracy	Horizontal: <6 meters (50%), <9 meters (90%) Altitude: <11 meters (50%), <18 meters (90%) Velocity: 0.06 m/sec. PPS: ±95 nanoseconds
Acquisition	Reacquisition: <2 sec. (90%) Hot Start: <14 sec. (50%), <18 sec. (90%) Warm Start: <38 sec. (50%), <45 sec. (90%) Cold Start: <90 sec. (50%), <170 sec. (90%)
Dynamics	Acceleration: 4g (39.2 m/sec2) Motional jerk: 20 m/sec3

Interface

Connectors	I/O: 8-pin (2x4) male header, micro terminal strip ASP 69553.01 RF: Low-profile coaxial connector H.FL-R-SMT (10), 50 Ohm
Serial Port	1 serial port (transmit/receive)
PPS	3.3 V CMOS-compatible, TTL-level pulse Once per second with the rising edge of the pulse synchronized with UTC
Protocols	TSIP @ 9600 baud, 8 bits NMEA 0183 v3.0, selectable baud rate, 8 bits

NMEA	GGA, VTG, GLL, ZDA, GSA, GSV and RMC Messages selectable by TSIP command; selection stored in flash memory

Electrical

Prime Power	+3.0 VDC to +3.6 VDC (3.3 V typ.)
Consumption	GPS board only: 100 mW@3.3V w/ embedded antenna: 133.3 mW@3.3V
Backup Power	+2.5 VDC to +3.6 VDC
Ripple Noise	Max 60 mV, peak-to-peak from 1 Hz to 1 MHz

Environmental

Operating Temp.	-40°C to +85°C
Storage Temp.	-55°C to +105°C
Vibration	0.008 g2/Hz, 5 Hz to 20 Hz 0.05 g2/Hz, 20 Hz to 100 Hz -3Db/OCTAVE, 100 Hz to 900 Hz
Humidity	5% to 95% R.H. non-condensing @ +60°C

Physical

Enclosure	Metal enclosure with solder mounting tabs
Outside Dim.	26 mm W x 26 mm L x 6 mm H (1.02" x 1.02" x 0.24")
Weight	Approximately 5.7 grams (0.2 ounces) including the shield

Accessories

Ultra-Compact Embedded Antenna

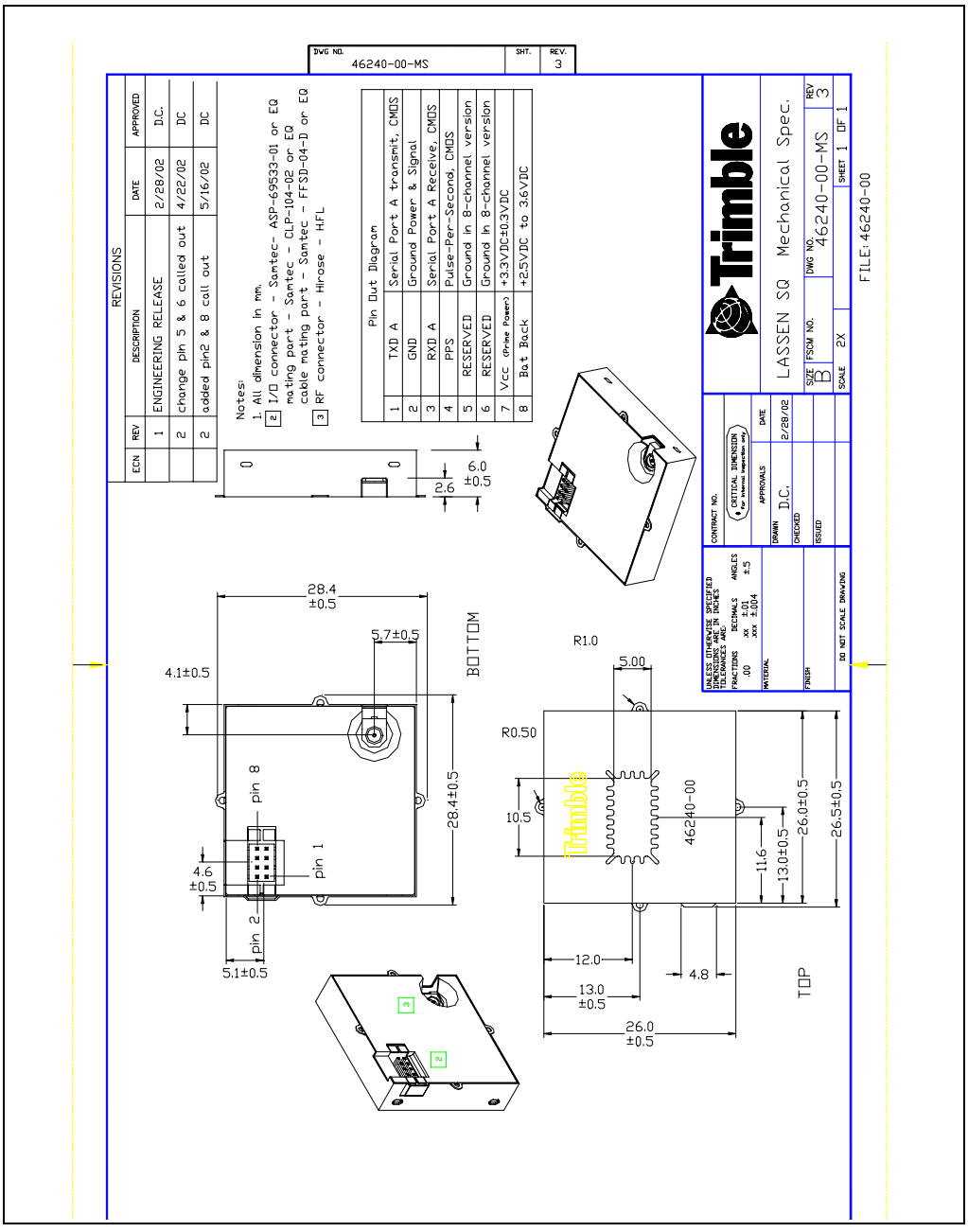
3.3V active miniature unpackaged antenna. Cable length: 8cm, Dim: 22 mm W x 21 mm L x 8 mm H (0.866” x 0.827” x 0.315”), Connector: HFL; mates directly to on-module RF connector.

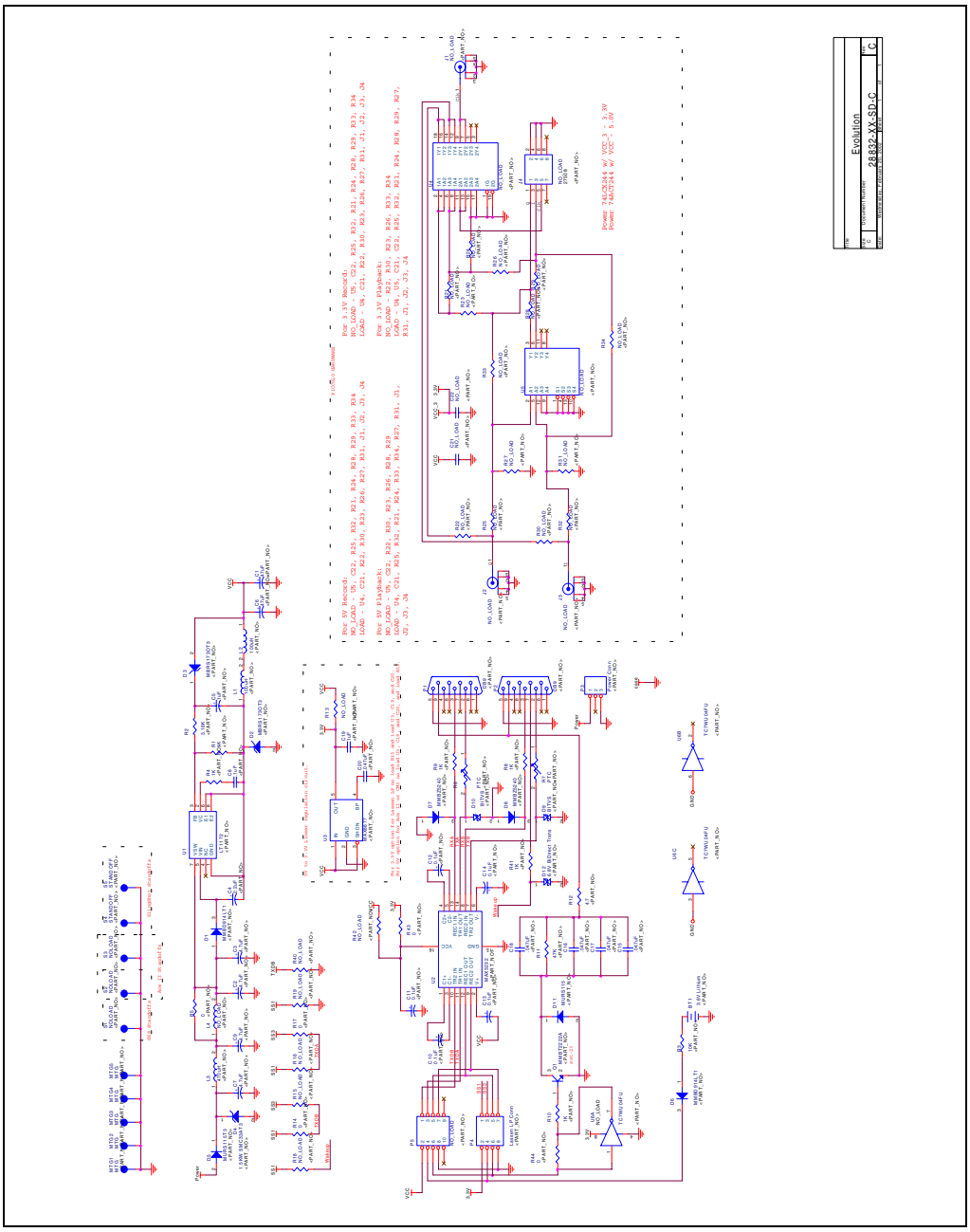
Compact Unpackaged Antenna

3 V active micropatch unpackaged antenna. Cable length: 11cm, Dim: 34.6 mm W x 29 mm L x 9 mm H (1.362” x 1.141” x 0.354”), Connector: MCX; mates with optional RF transition cable to the on-module RF connector.

Compact Magnetic-Mount Antenna

3 V active micropatch antenna with magnetic mount. Cable length: 5m, Dim: 42mm W x 50.5mm L x 13.8mm H (1.65” x 1.99” x 0.55”), Connector: MCX; mates through the optional RF transition cable to the on-module RF connector.


DWG NO. 46240-00-MS
SHT.
REV. 3
REVISIONS
ECN | REV | DESCRIPTION | DATE | APPROVED
1 | ENGINEERING RELEASE | 2/28/02 | D.C.
2 | change pin 5 & 6 called out | 4/22/02 | DC
2 | added pin2 & 8 call out | 5/16/02 | DC
Notes:
1. All dimension in mm.
2 I/O connector - Santec- ASP-69533-01 or EQ
mating part - Santec - CLP-104-02 or EQ
cable mating part - Santec - FFSD-04-D or EQ
3 RF connector - Hirose - HFL
Pin Out Diagram
1 TXD A Serial Port A transmit, CMOS
2 GND Ground Power & Signal
3 RXD A Serial Port A Receive, CMOS
4 PPS Pulse-Per-Second, CMOS
5 RESERVED Ground in 8-channel version
6 RESERVED Ground in 8-channel version
7 Vcc (Prime Power) +3.3VDC±0.3VDC
8 Bat Back +2.5VDC to 3.6VDC
6.0 ±0.5
2.6
28.4 ±0.5
5.7±0.5
4.1±0.5
BOTTOM
pin 8
pin 1
pin 2
4.6 ±0.5
5.1±0.5
28.4±0.5
R1.0
5.00
R0.50
10.5
Trimble
46240-00
11.6
13.0±0.5
26.0±0.5
26.5±0.5
12.0
13.0 ±0.5
26.0 ±0.5
4.8
TOP
Trimble
LASSEN SQ Mechanical Spec.
SIZE B
FSCM NO.
DWG NO. 46240-00-MS
REV 3
SCALE 2X
SHEET 1 OF 1
FILE: 46240-00
CONTRACT NO.
CRITICAL DIMENSION
APPROVALS
DATE
DRAWN D.C. 2/28/02
CHECKED
ISSUED
UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES ARE:
FRACTIONS .00 DECIMALS .XX ±.01 .XXX ±.004 ANGLES ±.5
MATERIAL
FINISH
DO NOT SCALE DRAWING



Ultra Compact Embedded Antenna

Antenna

Frequency Range	1,575.42 ± 1.023MHz
Gain	90°: 3.0dBi min.; 20°: -4.0dBi min. (mounted on the 60mm X 60mm square ground plane)
Polarization	RHCP
Axial Ratio	90°: 4.0dB max.; 10°: 6.0dB max. (mounted on the 60mm X 60mm square ground plane)

LNA

Frequency range	1,575.42 ±1.023MHz
Gain	24 3 dB (+25°C ± 5°C) 24 ± 4 dB (-40°C to 85°C)
Noise	1.5dB max. (+25°C ± 5°C) 2.0dB max. (+85°C)

Out of band rejection

fo=1,575.42MHz	
fo±20MHz	7dB min.
fo±30MHz	12dB min.
fo±50MHz	20dB min.
fo±100MHz	30dB min.

Output Impedance	50Ω
Output VSWR	2.0MAX.

Overall Specifications

Frequency Range	1,575.42±1.023MHz
Gain	27 ± 3dBi (+25°C ± 5°C) 27 ± 4dBi (-40°C TO 85°C) (mounted on the 60mm x 60mm square ground plane)
Output Impedance	50Ω
VSWR	2.0 max.
MTBF	5.13E+6Hr.

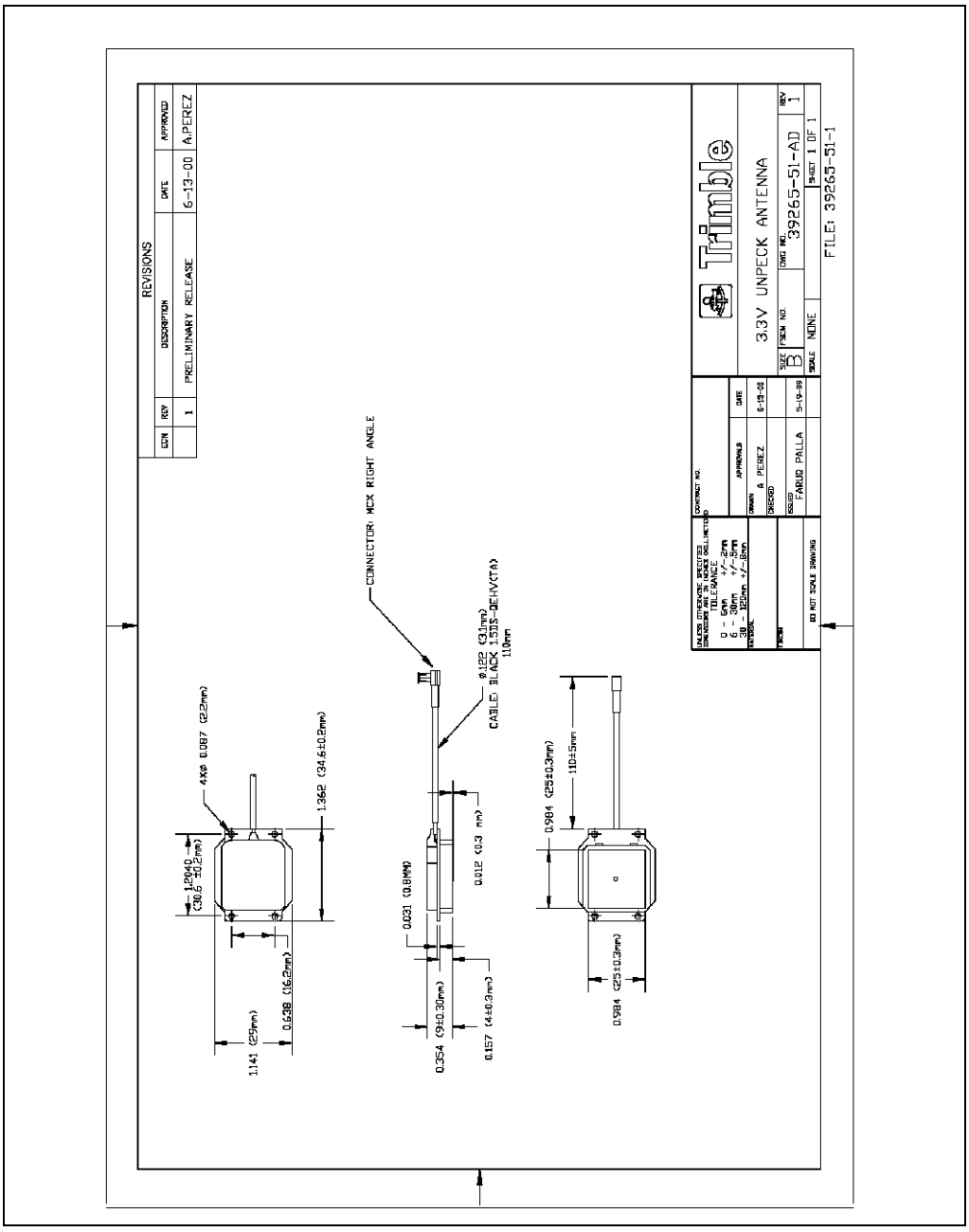
Storage Conditions

Store in room condition as listed below.

Temperature	-20°C to 45°C, humidity 80% max.
Data	Gain (at 3.0V ± 0.3V f=1575MHz) and power consumption at room temperature

General

Operating Temp.	-40°C to + 85°C
Storage Temp.	-40°C to + 100°C
Relative Humidity	20% to 95%
Input voltage	3.0V ± 0.3V
Power Cons.	5-13mA
Output connect.	HFL
Cable	0.8D 85mm: BLACK
Weight	20g TYP


REVISIONS
ECN
REV
DESCRIPTION
DATE
APPROVED
1
PRELIMINARY RELEASE
6-13-00
A.PEREZ
CONNECTOR: MCX RIGHT ANGLE
CABLE: BLACK 1.13IS-QEHVCTA)
110mm
4X ø 0.087 (2.2mm)
1.362 (34.6±0.8mm)
1.141 (29mm)
0.638 (16.2mm)
0.354 (9±0.30mm)
0.157 (4±0.3mm)
0.984 (25±0.3mm)
0.984 (25±0.3mm)
3.3V UNPECK ANTENNA
Trimble
39265-51-AD
REV 1
SHEET 1 OF 1
FILE: 39265-51-1
DRAWN A PEREZ 6-19-00
CHECKED
ISSUED FARUD PALLA 5-19-99
SIZE B
SCALE NONE
DO NOT SCALE DRAWING

Compact Magnetic Mount Antenna

Antenna

Frequency Range	1,575.42+/-1.023MHz
Gain	90°: 3.0dBi min.; 20°: -4.0dBi min. (mounted on the 65mm X 65mm square ground plane)
Polarization	RHCP
Axial Ratio	90°: 4.0dB max.; 10°: 6.0dB max. (mounted on the 65mm X 65mm square ground plane)

LNA

Frequency range	1,575.42 ±1.023MHz
Gain	28 ±3 dB (-40°C to 85°C)
Noise	1.5dB max. (+25°C ± 5°C) 2.2dB max. (+85°C)

Out of band rejection

fo=1,575.42MHz	
fo±20MHz	7dB min.
fo±30MHz	12dB min.
fo+/-50MHz	20dB min.
fo±100MHz	30dB min.

Output Impedance	50Ω
Output VSWR	2.0max.

Overall Specifications

Frequency Range	1,575.42 ± 1.023MHz
Gain	27 ± 3dBi (+25°C ± 5°C) 27 ± 4dBi (-40°C to 85°C) (mounted on the 65mm x 65mm square ground plane)
Output Impedance	50Ω
VSWR	2.0MAX.
ESD	Antenna surface ± 15KV Connector pin ± 8KV
MTBF	5.13E+6Hr.

Storage Conditions

Temperature	-20°C to 45°C, humidity 80% max.

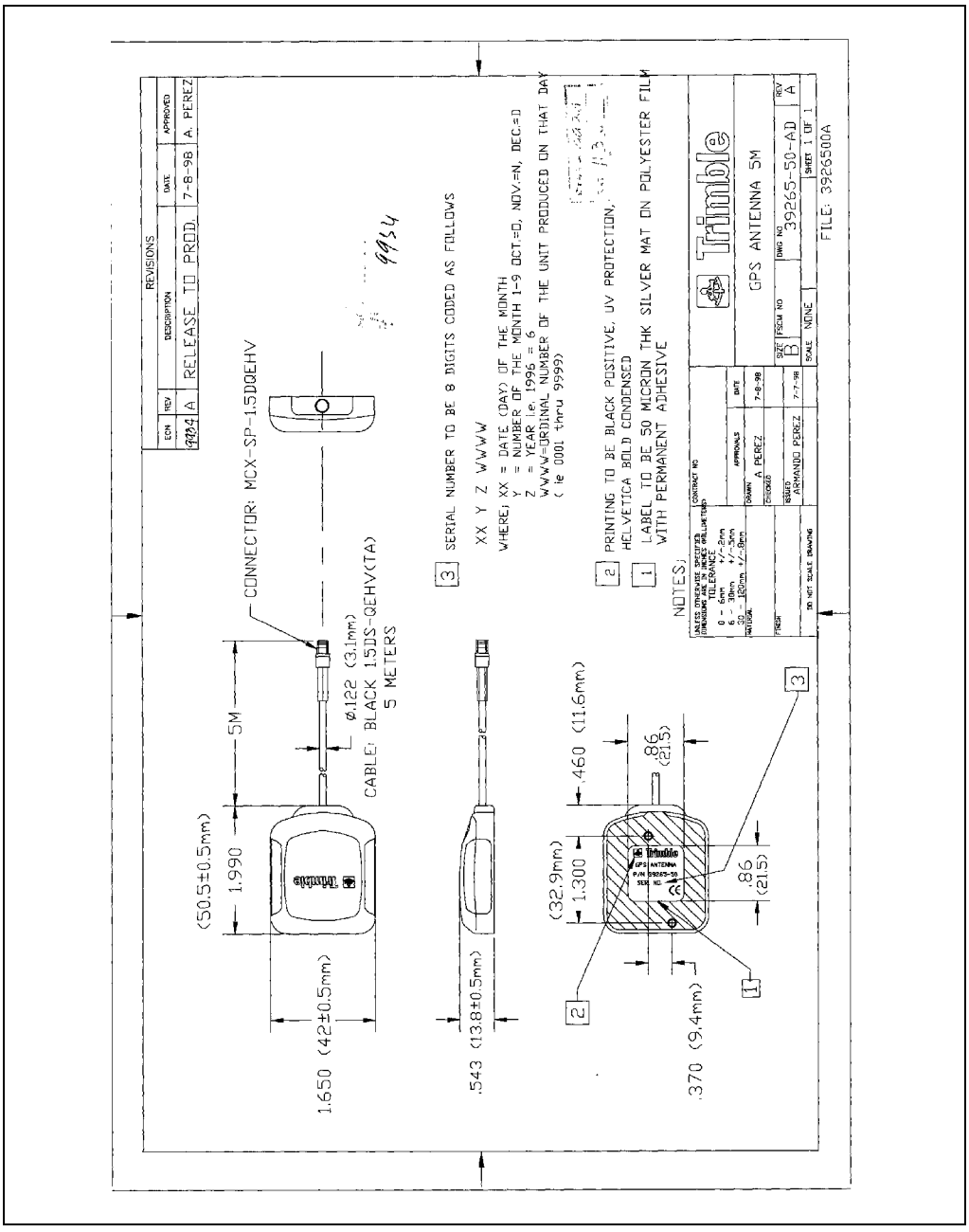
consumption at room temperature

General

Operating Temp.	-40°C to + 85°C
Storage Temp.	-40°C to + 100°C
Relative Humidity	20% to 95%
Input voltage	3.0V ± 0.3V
Power Cons.	15mA max (room temp)
Output connect.	MCX (plug)
Cable	1.5DS-QEHV (Shikoku Cable Co.) black: 5m
Weight	20g TYP

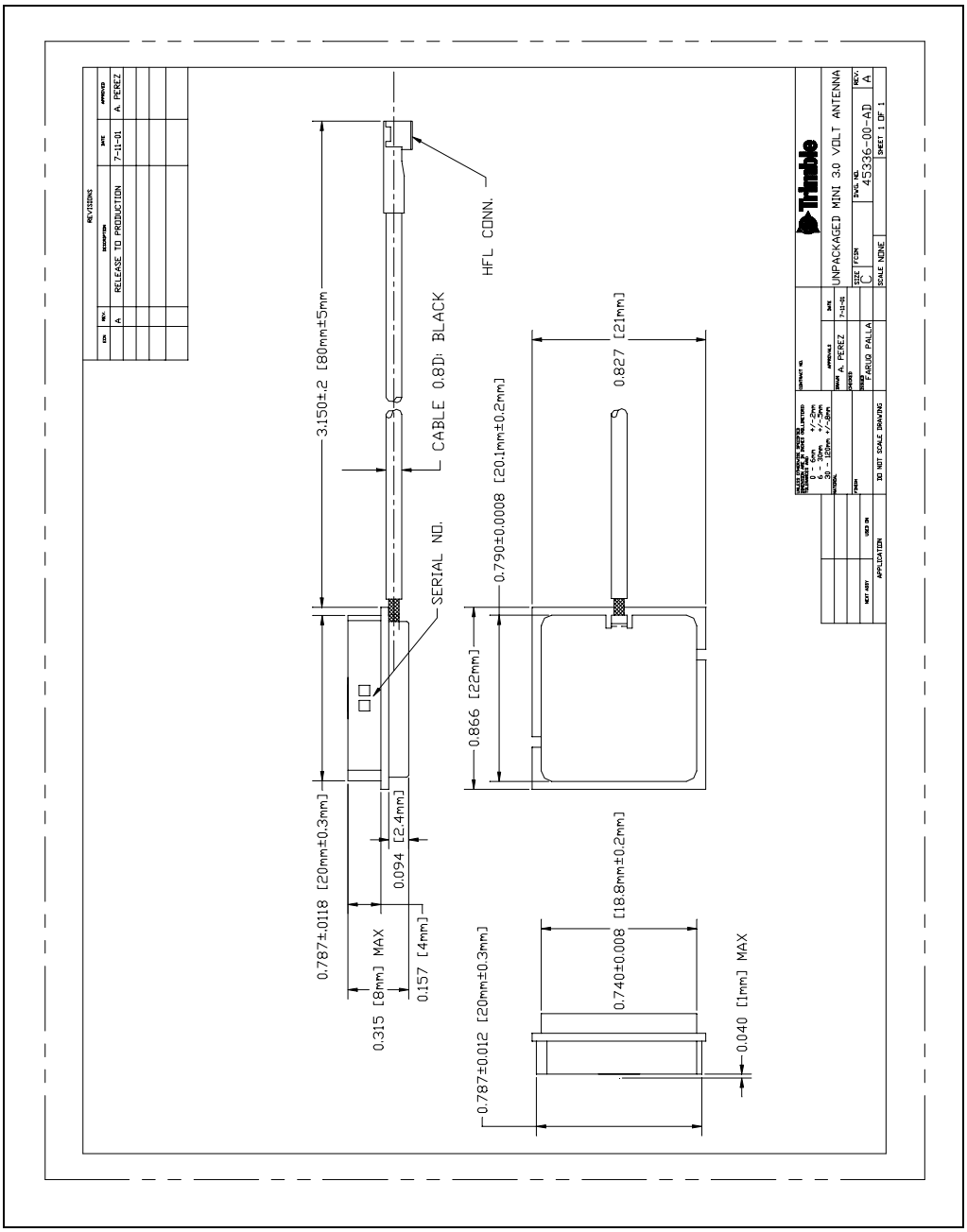
Mechanical

Mounting	Magnetic mount
Force of magnet	3.0kgf min.
Water proof	JISD0203 S2)
Shock	50G vertical axis 30G all axis
Vibration	10~200Hz log sweep 3.0G (sweep time: 15 min.) 3 axis
Withstand	min. speed of 180Km/h
Cable pulling force	5kgf min. Visible or electrical damage must not appear when applying up to 5kgf pulling force between cable and antenna as well as between cable and connector.
Bending test	After bending test 90° right and left x 1000 cycles, no permanent damage found
Anti-corrosion	Based on JIS Z 2371, spray 5%; salt water 35°C; should not rust after 96 hours
Weight	115g ± 15g


REVISIONS
ECN
REV
DESCRIPTION
DATE
APPROVED
9914
A
RELEASE TO PROD.
7-8-98
A. PEREZ
CONNECTOR: MCX-SP-1.5D0EHV
5M
Ø.122 (3.1mm)
CABLE: BLACK 1.5DS-QEHV(TA)
5 METERS
3 SERIAL NUMBER TO BE 8 DIGITS CODED AS FOLLOWS
XX Y Z WWWW
WHERE: XX = DATE (DAY) OF THE MONTH
Y = NUMBER OF THE MONTH 1-9 OCT.=O, NOV.=N, DEC.=D
Z = YEAR ie. 1996 = 6
WWWW=ORDINAL NUMBER OF THE UNIT PRODUCED ON THAT DAY
(ie 0001 thru 9999)
2 PRINTING TO BE BLACK POSITIVE, UV PROTECTION,
HELVETICA BOLD CONDENSED
1 LABEL TO BE 50 MICRON THK SILVER MAT ON POLYESTER FILM
WITH PERMANENT ADHESIVE
NOTES:
(50.5±0.5mm)
1.990
1.650 (42±0.5mm)
.543 (13.8±0.5mm)
.460 (11.6mm)
.86 (21.5)
(32.9mm)
1.300
.370 (9.4mm)
Trimble
GPS ANTENNA 5M
39265-50-AD
REV A
FILE: 39265OOA
SHEET 1 OF 1
SCALE NONE
DO NOT SCALE DRAWING

Compact Unpackaged Antenna

The specifications for this antenna are identical to that of the Compact Magnetic Mount Antenna described earlier in this Appendix (see page 162). A drawing is provided on the following page.


REVISIONS
REV.
DESCRIPTION
DATE
APPROVED
A
RELEASE TO PRODUCTION
7-11-01
A. PEREZ
3.150±.2 [80mm±5mm
HFL CONN.
CABLE 0.8D: BLACK
SERIAL NO.
0.790±0.0008 [20.1mm±0.2mm]
0.827 [21mm]
0.866 [22mm]
0.787±.0118 [20mm±0.3mm]
0.094 [2.4mm]
0.315 [8mm] MAX
0.157 [4mm]
0.787±0.012 [20mm±0.3mm]
0.740±0.008 [18.8mm±0.2mm]
0.040 [1mm] MAX
Trimble
UNPACKAGED MINI 3.0 VOLT ANTENNA
DWG NO.
45336-00-AD
REV.
A
SCALE NONE
SHEET 1 OF 1
DO NOT SCALE DRAWING
APPLICATION

Glossary

This section defines technical terms and abbreviations used in this manual. It includes terms from the field of GPS technology.

2-D GPS mode A procedure of determining a 2-D position using signals received from the best (or only) three available GPS satellites. Altitude is assumed to be known and constant. A 2-D position solution will only be determined if signals from three or more satellites are available.

2 dRMS Twice the distance root mean squared. The error distance within which 95% of the position solutions will fall.

3-D Three Dimensional. A 3-D position is defined as latitude, longitude, and altitude.

2-D Two Dimensional. A 2-D position is defined as latitude, longitude, and altitude.

3-D GPS mode A procedure of determining a 3-D position using signals received from the best (or only) four available GPS satellites. A 3-D position solution will only be determined if signals from four or more satellites are available.

almanac A reduced-precision subset of the ephemeris parameters. Used by the receiver to compute the elevation angle, azimuth angle, and estimated Doppler of the satellites. Each satellite broadcasts the almanac for all the satellites in the system.

ASCII American Standard Code for Information Interchange. A standard set of 128 characters, symbols and control codes used for computer communications. ASCII characters require 7 bits of data to send, but are often sent 8 bits at a time with the extra bit being a zero.

asynchronous communication A method of sending data in which the bits can be sent at random times. Data transmission is not synchronized to a clock. With asynchronous transmission, each character is transmitted one at a time with a “start” bit at the beginning and one or more “stop” bits at the end. Any amount of time can elapse before the next character is sent. \

auto GPS mode A procedure of automatically determining either a 2-D or 3-D position using signals received from GPS satellites. The solution automatically transitions between 2-D and 3-D depending on the number of satellites available, the PDOP of the available satellites, and the defined PDOP switch value. (See PDOP and PDOP constellation switch).

azimuth angle The angle of the line-of-site vector, projected on the horizontal plane, measured clockwise from true North.

bandwidth The range of frequencies occupied by a signal. Also, the information carrying capability of a communication channel or line.

baud A measure of the speed of data transmission. Baud and bit rate are the same for direct equipment interconnections (e.g., via RS-232). Baud and bit rate are not the same for modulated data links, whether wire or radio.

bit Binary digit. The smallest unit of information into which digital data can be subdivided and which a computer can hold. Each bit has only two values (e.g., on/off, one/zero, true/false).

bit rate The rate at which bits are transmitted over a communication path. Normally expressed in bits per second (bps).

byte A set of contiguous bits that make up a discrete item of information. A byte usually consists of a series of 8 bits, and represents one character.

C/A code The Coarse/Acquisition code. This is the civilian code made available by the Department of Defense. It is subject to selective availability (SA). Users can reduce the effects of SA by using differential GPS.

carrier The radio signal on which information is carried. The carrier can be sensed to determine the presence of a signal.

channel Either a single frequency or a pair of radio frequencies used as a communication path.

chip The length of time to transmit either a zero or a one in a binary pulse code.

chip rate Number of chips per second (e.g., C/A code = 1.023 MHz).

configuration A set of conditions or parameters that define the structure of an item. A configuration defines the GPS processing and characteristics of the RS-232 interface ports. The term configuration can also define the hardware components that comprise a subsystem or system.

data bits The bits in a byte of data which carry the actual information.

datum Refers to a mathematical model of the earth. Many local datums model the earth for a small region: e.g., Tokyo datum, Alaska, NAD-27 (North America). Others, WGS-84, for example, model the whole earth.

DCE Data Communications Equipment. The equipment that provides the functions required to establish, maintain, and terminate a communication connection. Any equipment that connects to DTE using an RS-232 or CCITTV.24 standard interface.

default setting A preset or initial value that is assumed to be the preferred or appropriate selection for most situations. The Lassen SK II GPS sensor is shipped with factory default configuration settings; the settings were determined by Trimble Navigation.

DGPS see *differential GPS*

DGPS reference station A device that tracks all GPS satellites in view, periodically performs inter-channel calibrations, and calculates and transmits differential corrections.

differential capable A term used to describe a GPS receiver that is capable of receiving and applying differential GPS corrections.

differential GPS A procedure of correcting GPS solutions to achieve improved position accuracy. Differential GPS provides 2 to 5 meter position accuracy. Differential accuracy is obtained by applying corrections determined by the stationary Differential GPS Reference Station to the GPS data collected by the RPU unit on-board the vehicle.

differential processing GPS measurements can be differenced between receivers, satellites, and epochs. Although many combinations are possible, the present convention for differential processing of GPS phase measurements is to take differences between receivers (single difference), then between satellites (double difference), then between measurement epochs (triple difference).

differential relative positioning Determination of relative coordinates of two or more receivers which are simultaneously tracking the same satellites. Static differential GPS involves determining baseline vectors between pairs of receivers. Also see *differential GPS*

dilution of precision A description of the purely geometrical contribution to the uncertainty in a position fix, given by the expression DOP = SQRT TRACE (AA) where AA is the design matrix for the instantaneous position solution (dependent on satellite-receiver geometry). The DOP factor depends on the parameters of the position-fix solution. Standard terms for the GPS application are:

GDOP: Geometric (three position coordinates plus clock offset in the solution)

PDOP: Position (three coordinates)

HDOP: Horizontal (two horizontal coordinates)

VDOP: Vertical (height only)

TDOP: Time (clock offset only)

DOP see *dilution of precision*.

Doppler aiding The use of Doppler carrier-phase measurements to smooth code-phase position measurements.

Doppler shift The apparent change in frequency of a received signal due to the rate of change of the range between the transmitter and receiver.

earth-centered earth-fixed Cartesian coordinate system where the X direction is the intersection of the prime meridian (Greenwich) with the equator. The vectors rotate with the earth. Z is the direction of the spin axis.

elevation angle The angle between the line of sight vector and the horizontal plane.

elevation mask angle A measure of the minimum elevation angle, above the horizon, above which a GPS satellite must be located before the signals from the satellite will be used to compute a GPS location solution. Satellites below the elevation angle are considered unusable. The elevation mask angle is used to prevent the GPS receiver from computing position solutions using satellites which are likely to be obscured by buildings or mountains.

ellipsoid In geodesy, unless otherwise specified, a mathematical figure formed by revolving an ellipse about its minor axis. It is often used interchangeably with spheroid. Two quantities define an ellipsoid; these are usually given as the length of the semimajor axis, a, and the flattening, f = (a - b)/a, where b is the length of the semiminor axis.

ephemeris A set of parameters that describe the satellite orbit very accurately. It is used by the receiver to compute the position of the satellite. This information is broadcast by the satellites.

epoch Measurement interval or data frequency, as in making observations every 15 seconds. Loading data using 30-second epochs means loading every other measurement.

firmware A set of software computer/processor instructions that are permanently or semi-permanently resident in read-only memory.

frequency The number of vibrations per second of an audio or radio signal. Measured in hertz (Hz), kilohertz (kHz), or megahertz (MHz).

GPS frequencies are: L1 = 1575.42 MHz

L2 = 1227.60 MHz

GDOP Geometric Dilution of Precision. GDOP describes how much an uncertainty in pseudo-range and time affects the uncertainty in a position solution. GDOP depends on where the satellites are relative to the GPS receiver and on GPS clock offsets.

geodetic datum A mathematical model designed to best fit part or all of the geoid. It is defined by an ellipsoid and the relationship between the ellipsoid and a point on the topographic surface established as the origin of datum. This relationship can be defined by six quantities, generally (but not necessarily) the geodetic latitude, longitude, and the height of the origin, the two components of the deflection of the vertical at the origin, and the geodetic azimuth of a line from the origin to some other point. The GPS uses WGS-84.

geoid The actual physical shape of the earth which is hard to describe mathematically because of the local surface irregularities and sea-land variations. In geodetic terms it is the particular equipotential surface which coincides with mean sea level, and which may be imagined to extend through the continents. This surface is everywhere perpendicular to the force of gravity.

GPD GPS with differential corrections applied.

GPS	Global Positioning System. A constellation of 24 radio navigation (not communication) satellites which transmit signals used (by GPS receivers) to determine precise location (position, velocity, and time) solutions. GPS signals are available world-wide, 24 hours a day, in all weather conditions. This system also includes 5 monitor ground stations, 1 master control ground station, and 3 upload ground stations.
GPS antenna	An antenna designed to receive GPS radio navigation signals.
GPS processor	An electronic device that interprets the GPS radio navigation signals (received by a GPS antenna) and determines a location solution. The GPS processor may also be able to apply (and determine) differential GPS corrections.
GPS receiver	The combination of a GPS antenna and a GPS processor.
GPS time	The length of the second is fixed and is determined by primary atomic frequency standards. Leap-seconds are not used, as they are in UTC. Therefore, GPS time and UTC differ by a variable whole number of seconds.
HDOP	Horizontal Dilution of Precision.
HOW	Handover word. The word in the GPS message that contains time synchronization information for the transfer from C/A to P-code.
interface cable (serial)	The interface cable allows data to flow between the Lassen SK II GPS and the communication equipment. One end of the cable has a 9-pin female RS-232 connector and the other end of this cable has a 9-pin male RS-232 connectors.
interference	Refers to the unwanted occurrences on communication channels that are a result of natural or man-made noises and signals, not properly a part of the signals being transmitted or received.
integrated Doppler	A measurement of Doppler shift frequency or phase over time.

IODE	Issue Of Data, Ephemeris. Part of the navigation data. It is the issue number of the ephemeris information. A new ephemeris is available usually on the hour. Especially important for Differential GPS operation that the IODE change is tracked at both the reference station and mobile stations.
jamming	Interference (in either transmitting or receiving signals) caused by other radio signals at exactly or approximately the same frequency
Kalman filter	A numerical method used to track a time-varying signal in the presence of noise. If the signal can be characterized by some number of parameters that vary slowly with time, then Kalman filtering can be used to tell how incoming raw measurements should be processed to best estimate those parameters as a function of time.
masks	See *satellite masks*.
maximum PDOP	A measure of the maximum Position Dilution of Precision (PDOP) that is acceptable in order for the GPS processor to determine a location solution (see PDOP).
NAVSTAR	The name given to the GPS satellites, built by Rockwell International, which is an acronym formed from NAVigation System with Time And Ranging.
NMEA	National Marine Electronics Association. An association that defines marine electronic interface standards for the purpose of serving the public interest.
NMEA 0183 message	NMEA 0183 is a standard for interfacing marine electronics navigational devices. The standard specifies the message format used to communicate with marine devices/components.
packet	An “envelope” for data, which contains addresses and error checking information as well as the data itself.
parity	A scheme for detecting certain errors in data transmission. Parity defines the condition (i.e., even or odd) of the number of items in a set (e.g., bits in a byte).

PDOP	Position Dilution of Precision. PDOP is a unitless figure of merit that describes how an uncertainty in pseudo-range affects position solutions.
PDOP constellation switch	A value, based on PDOP, that defines when the GPS receiver/processor should switch between 2-D and 3-D GPS modes. The PDOP constellation switch is only active when the GPS mode of operation is set to Auto.
PRN	Pseudo-random noise. Each GPS satellite generates its own distinctive PRN code, which is modulated onto each carrier. The PRN code serves as identification of the satellite, as a timing signal, and as a subcarrier for the navigation data.
protocol	A formal set of rules that describe a method of communication. The protocol governs the format and control of inputs and outputs.
pseudo-range	A measure of the range from the GPS antenna to a GPS satellite. Pseudo-range is obtained by multiplying the speed of light by the apparent transit time of the signal from the GPS satellite. Pseudo-range differs from actual range because the satellite and user clocks are offset from GPS time and because of propagation delays and other errors.
RAM	Random-Access Memory.
random-access memory	Memory in which information can be referred to in an arbitrary or random order. The contents of RAM are lost when the System Unit is turned off.
range	A term used to refer to the distance radio signals can travel before they must be received or repeated due to loss of signal strength, the curvature of the earth and the noise introduced because of moisture in the air surrounding the earth's surface.
range rate	The rate of change of range between the satellite and receiver. The range to a satellite changes due to satellite and observer motions. Range rate is determined by measuring the Doppler shift of the satellite beacon carrier.

read-only memory Memory whose contents can be read, but not changed. Information is placed into ROM only once. The contents of ROM are not erased when the system unit's power is turned off.

real time clock An electronic clock, usually battery powered, that keeps current time. Used by a GPS receiver during a warm or hot start to determine where to search for GPS satellite signals.

relative positioning The process of determining the vector distance between two points and the coordinates of one spot relative to another. This technique yields GPS positions with greater precision than a single point positioning mode can.

rise/set time Refers to the period during which a satellite is visible; i.e., has an elevation angle that is above the elevation mask. A satellite is said to “rise” when its elevation angle exceeds the mask and “set” when the elevation drops below the mask.

ROM Read-Only Memory.

RS-232 A communication standard for digital data. Specifies a number of signal and control lines. RS-232 is often associated with a 25 pin connector called a DB-25.

RTCM Radio Technical Commission for Maritime Services. Commission that recommends standards for differential GPS services. “RTCM Recommended Standards For Differential GPS Service,” prepared by RTCM Special Committee No. 104 (RTCM SC-104), defines a communication protocol for sending GPS differential corrections from a differential reference station to remote GPS receivers.

satellite masks As satellites approach the horizon, their signals can become weak and distorted, preventing the receiver from gathering accurate data. Satellite masks enable you to establish criteria for using satellite data in a position solution. There are three types of satellite masks: Elevation, SNR, and PDOP.

SA	Selective Availability. This is the name of the policy and the implementation scheme by which unauthorized users of GPS will have their accuracy limited to 100 meters 2D RMS horizontal and 156 meters 2D RMS vertical.
SEP	Spherical Error Probability. The radius of a sphere such that 50% of the position estimates will fall within the surface of the sphere.
serial communication	A system of sending bits of data on a single channel one after the other, rather than simultaneously.
serial port	A port in which each bit of information is brought in/out on a single channel. Serial ports are designed for devices that receive data one bit at a time.
signal to noise level	GPS signals with SNRs that do not meet the mask criteria are considered unusable.
signal to noise ratio	A measure of the relative power levels of a communication signal and noise on a data line. SNR is expressed in decibels (dB).
SNR	Signal to Noise Ratio.
spread spectrum	The received GPS signal is a wide bandwidth, low-power signal (-160dBW). This property results from modulating the L-band signal with a PRN code in order to spread the signal energy over a bandwidth which is much greater than the signal information bandwidth. This is done to provide the ability to receive all satellites unambiguously and to provide some resistance to noise and multipath.
SPS	Standard Positioning Service. Refers to the GPS as available to the authorized user.
start bit	In asynchronous transmission, the start bit is appended to the beginning of a character so that the bit sync and character sync can occur at the receiver equipment.
stop bit	In asynchronous transmission, the stop bit is appended to the end of each character. It sets the receiving hardware to a condition where it looks for the start bit of a new character.

SV Space Vehicle (GPS satellite).

synchronous communication A method of sending digital data in which the bits come at fixed, rather than random, times and are synchronized to a clock.

TAIP Trimble ASCII Interface Protocol. Designed originally for vehicle tracking applications, TAIP uses printable uppercase ASCII characters in 16 message types for easy integration with mobile data modems, terminals, and personal computers. The TAIP protocol is defined in full in Appendix C.

TANS Trimble Advanced Navigation Sensor. Also refers to a Trimble-specified interface protocol for digital packet communication to/from the GPS receiver. Data output includes time-tagged position and velocity, satellite status, dilution of precision factors and diagnostics of GPS receiver operational status.

Also see *TSIP*

TNL 4000RL Trimble Navigation, Ltd. Reference Locator (4000RL). Product name for the Differential GPS Reference Station.

TSIP Trimble Standard Interface Protocol. A binary/hex packet bi-directional protocol, also known as the TANS protocol. Used by a large number of Trimble sensors. TSIP is the subset of TANS which is recognized by all Trimble sensors except the 4000 series. The TSIP protocol is defined in full in Appendix A.

URA Satellite user range accuracy. The URA is sent by the satellite and is computed by the GPS operators. It is a statistical indicatory of the contribution of the apparent clock and ephemeris prediction accuracies to the ranging accuracies obtainable with a specific satellite based on historical data.

UTC Universal Time Coordinated. Uniform atomic time system/standard that is maintained by the US Naval Observatory. UTC defines the local solar mean time at the Greenwich Meridian.

UTC offset The difference between local time and UTC (Example: UTC - EST = 5 hours).

TMS320x281x, 280x DSP Serial Communication Interface (SCI) Reference Guide

Literature Number: SPRU051B
November 2004

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Contents

Figures

Tables

This page is intentionally left blank.

Chapter 1

Overview

The serial communications interface (SCI) is a two−wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. The SCI receiver and transmitter each have a 16-level deep FIFO for reducing servicing overhead, and each has its own separate enable and interrupt bits. Both can be operated independently for half-duplex communication, or simultaneously for full-duplex communication.

To ensure data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The bit rate is programmable to different speeds through a 16-bit baud-select register.

This reference guide is applicable for the SCI found on both the TMS320x280x and TMS3201x281x family of processors. This includes all Flash-based, ROM-based and RAM-based devices within these families.

Topic **Page**

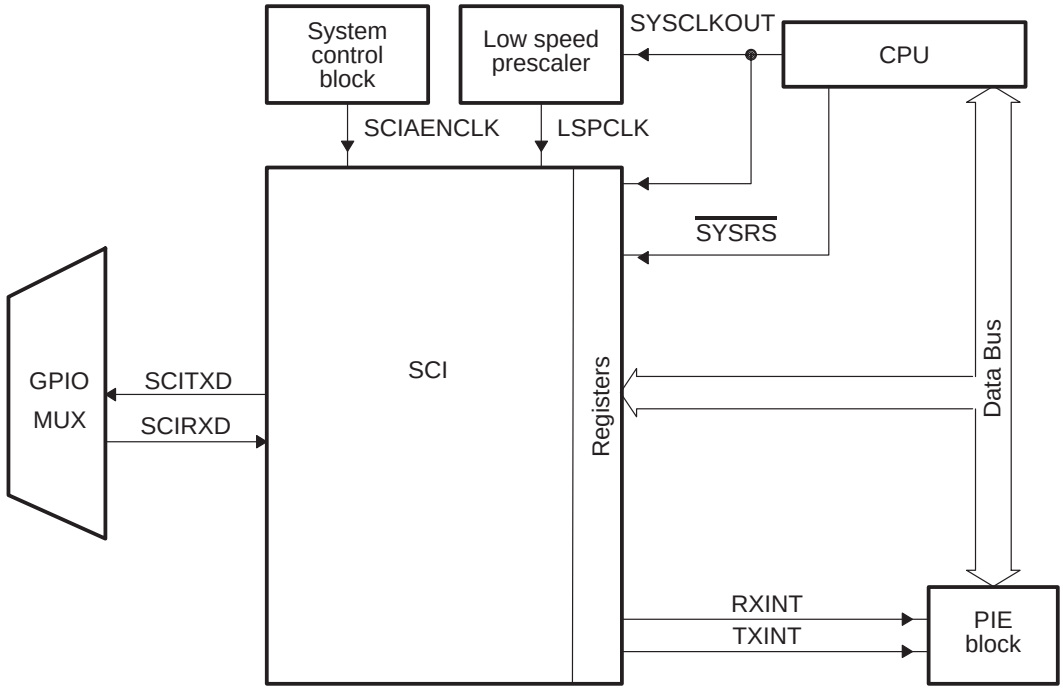
Note: 28x Enhanced Features

The 28x SCI features several enhancements compared to the 240xA SCI. See section 1.2.10 for a description of these features.

1.1 Enhanced SCI Module Overview

The SCI interfaces are shown in Figure 1−1.

Figure 1−1. SCI CPU Interface



Features of the SCI module include:

- Two external pins:
  - SCITXD: SCI transmit-output pin
  - SCIRXD: SCI receive-input pin

  Both pins can be used as GPIO if not used for SCI.
- Baud rate programmable to 64K different rates
- Data-word format
  - One start bit
  - Data-word length programmable from one to eight bits
  - Optional even/odd/no parity bit
  - One or two stop bits

- ❑ Four error-detection flags: parity, overrun, framing, and break detection
- ❑ Two wake-up multiprocessor modes: idle-line and address bit
- ❑ Half- or full-duplex operation
- ❑ Double-buffered receive and transmit functions
- ❑ Transmitter and receiver operations can be accomplished through interrupt-driven or polled algorithms with status flags.
- ❑ Separate enable bits for transmitter and receiver interrupts (except BRKDT)
- ❑ NRZ (non-return-to-zero) format
- ❑ 13 SCI module control registers located in the control register frame beginning at address 7050h

  All registers in this module are 8-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7−0), and the upper byte (15−8) is read as zeros. Writing to the upper byte has no effect.

Enhanced features:

- ❑ Auto-baud-detect hardware logic
- ❑ 16-level transmit/receive FIFO

Figure 1−2 shows the SCI module block diagram.

The SCI port operation is configured and controlled by the registers listed in Table 1−1 and Table 1−2.

Figure 1−2. Serial Communications Interface (SCI) Module Block Diagram

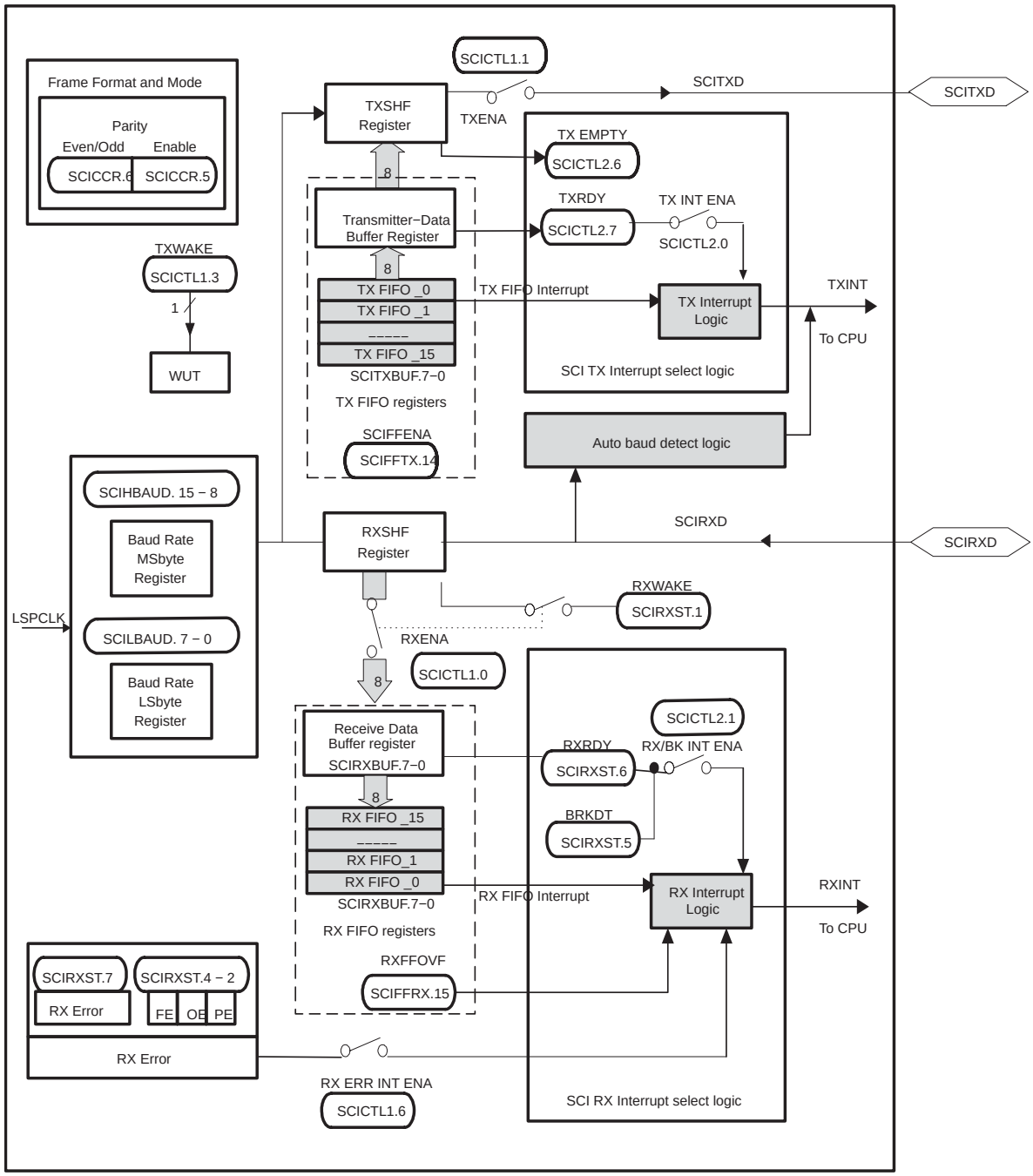


Table 1−1. SCI-A Registers

Name	Address Range	Size (x16)	Description
SCICCR	0x0000−7050	1	SCI-A Communications Control Register
SCICTL1	0x0000−7051	1	SCI-A Control Register 1
SCIHBAUD	0x0000−7052	1	SCI-A Baud Register, High Bits
SCILBAUD	0x0000−7053	1	SCI-A Baud Register, Low Bits
SCICTL2	0x0000−7054	1	SCI-A Control Register 2
SCIRXST	0x0000−7055	1	SCI-A Receive Status Register
SCIRXEMU	0x0000−7056	1	SCI-A Receive Emulation Data Buffer Register
SCIRXBUF	0x0000−7057	1	SCI-A Receive Data Buffer Register
SCITXBUF	0x0000−7059	1	SCI-A Transmit Data Buffer Register
SCIFFTX	0x0000−705A	1	SCI-A FIFO Transmit Register
SCIFFRX	0x0000−705B	1	SCI-A FIFO Receive Register
SCIFFCT	0x0000−705C	1	SCI-A FIFO Control Register
SCIPRI	0x0000−705F	1	SCI-A Priority Control Register

Table 1−2. SCI-B Registers

Name	Address Range	Size (x16)	Description
SCICCR	0x0000−7750	1	SCI-B Communications Control Register
SCICTL1	0x0000−7751	1	SCI-B Control Register 1
SCIHBAUD	0x0000−7752	1	SCI-B Baud Register, High Bits
SCILBAUD	0x0000−7753	1	SCI-B Baud Register, Low Bits
SCICTL2	0x0000−7754	1	SCI-B Control Register 2
SCIRXST	0x0000−7755	1	SCI-B Receive Status Register
SCIRXEMU	0x0000−7756	1	SCI-B Receive Emulation Data Buffer Register
SCIRXBUF	0x0000−7757	1	SCI-B Receive Data Buffer Register
SCITXBUF	0x0000−7759	1	SCI-B Transmit Data Buffer Register
SCIFFTX	0x0000−775A	1	SCI-B FIFO Transmit Register
SCIFFRX	0x0000−775B	1	SCI-B FIFO Receive Register
SCIFFCT	0x0000−775C	1	SCI-B FIFO Control Register
SCIPRI	0x0000−775F	1	SCI-B Priority Control Register

Notes: 1) The registers are mapped to peripheral frame 2. This frame allows only 16-bit accesses. Using 32-bit accesses will produce undefined results.

2) SCIB is an optional peripheral. In some devices this may not be present. See the device-specific data sheet for peripheral availability.

1.2 Architecture

The major elements used in full-duplex operation are shown in Figure 1−2 and include:

- A transmitter (TX) and its major registers (upper half of Figure 1−2)
  - SCITXBUF — transmitter data buffer register. Contains data (loaded by the CPU) to be transmitted
  - TXSHF register — transmitter shift register. Accepts data from register SCITXBUF and shifts data onto the SCITXD pin, one bit at a time
- A receiver (RX) and its major registers (lower half of Figure 1−2)
  - RXSHF register — receiver shift register. Shifts data in from SCIRXD pin, one bit at a time
  - SCIRXBUF — receiver data buffer register. Contains data to be read by the CPU. Data from a remote processor is loaded into register RXSHF and then into registers SCIRXBUF and SCIRXEMU
- A programmable baud generator
- Data-memory-mapped control and status registers

The SCI receiver and transmitter can operate either independently or simultaneously.

1.2.1 SCI Module Signal Summary

Signal Name	Description
External signals	
SCIRXD	SCI Asynchronous Serial Port receive data
SCITXD	SCI Asynchronous Serial Port transmit data
Control	
Baud clock	LSPCLK Prescaled clock
Interrupt signals	
TXINT	Transmit interrupt
RXINT	Receive Interrupt

1.2.2 Multiprocessor and Asynchronous Communication Modes

The SCI has two multiprocessor protocols, the *idle-line* multiprocessor mode (see section 1.2.5 on page 1-11) and the *address-bit* multiprocessor mode (see section 1.2.6 on page 1-13). These protocols allow efficient data transfer between multiple processors.

The SCI offers the universal asynchronous receiver/transmitter (UART) communications mode for interfacing with many popular peripherals. The asynchronous mode (see section 1.2.7 on page 1-15) requires two lines to interface with many standard devices such as terminals and printers that use RS-232-C formats. Data transmission characteristics include:

- One start bit
- One to eight data bits
- An even/odd parity bit or no parity bit
- One or two stop bits

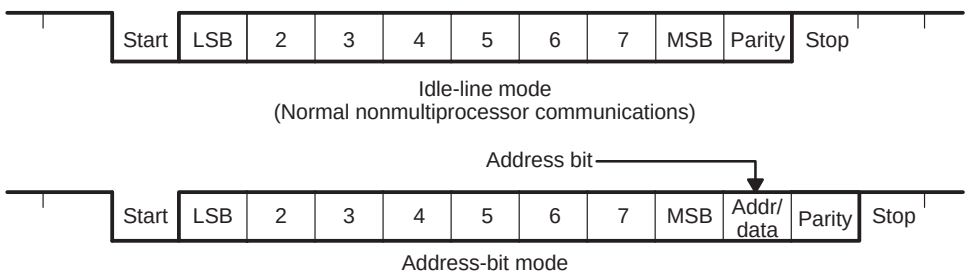
1.2.3 SCI Programmable Data Format

SCI data, both receive and transmit, is in NRZ (non-return-to-zero) format. The NRZ data format, shown in Figure 1−3, consists of:

- One start bit
- One to eight data bits
- An even/odd parity bit (optional)
- One or two stop bits
- An extra bit to distinguish addresses from data (address-bit mode only)

The basic unit of data is called a character and is one to eight bits in length. Each character of data is formatted with a start bit, one or two stop bits, and optional parity and address bits. A character of data with its formatting information is called a frame and is shown in Figure 1−3.

Figure 1−3. Typical SCI Data Frame Formats



To program the data format, use the SCICCR register. The bits used to program the data format are shown in Table 1−3.

Table 1−3. Programming the Data Format Using SCICCR

Bit(s)	Bit Name	Designation	Functions
2−0	SCI CHAR2−0	SCICCR.2:0	Select the character (data) length (one to eight bits).
5	PARITY ENABLE	SCICCR.5	Enables the parity function if set to 1, or disables the parity function if cleared to 0.
6	EVEN/ODD PARITY	SCICCR.6	If parity is enabled, selects odd parity if cleared to 0 or even parity if set to 1.
7	STOP BITS	SCICCR.7	Determines the number of stop bits transmitted—one stop bit if cleared to 0 or two stop bits if set to 1.

1.2.4 SCI Multiprocessor Communication

The multiprocessor communication format allows one processor to efficiently send blocks of data to other processors on the same serial link. On one serial line, there should be only one transfer at a time. In other words, there can be only one talker on a serial line at a time.

Address Byte

The *first byte* of a block of information that the talker sends contains an *address byte* that is read by all listeners. Only listeners with the correct address can be interrupted by the data bytes that follow the address byte. The listeners with an incorrect address remain uninterrupted until the next address byte.

Sleep Bit

All processors on the serial link set the SCI SLEEP bit (bit 2 of SCICTL1) to 1 so that they are interrupted only when the address byte is detected. When a processor reads a block address that corresponds to the CPU device address as set by your application software, your program must clear the SLEEP bit to enable the SCI to generate an interrupt on receipt of each data byte.

Although the receiver still operates when the SLEEP bit is 1, it does not set RXRDY, RXINT, or any of the receiver error status bits to 1 unless the address byte is detected and the address bit in the received frame is a 1 (applicable to address-bit mode). The SCI does not alter the SLEEP bit; your software must alter the SLEEP bit.

1.2.4.1 Recognizing the Address Byte

A processor recognizes an address byte differently, depending on the multiprocessor mode used. For example:

- The *idle-line mode* (section 1.2.5 on page 1-11) leaves a quiet space before the address byte. This mode does not have an extra address/data bit and is more efficient than the address-bit mode for handling blocks that contain more than ten bytes of data. The idle-line mode should be used for typical non-multiprocessor SCI communication.

- The *address-bit mode* (section 1.2.6 on page 1-13) adds an extra bit (that is, an address bit) into every byte to distinguish addresses from data. This mode is more efficient in handling many small blocks of data because, unlike the idle mode, it does not have to wait between blocks of data. However, at a high transmit speed, the program is not fast enough to avoid a 10-bit idle in the transmission stream.

1.2.4.2 Controlling the SCI TX and RX Features

The multiprocessor mode is software selectable via the ADDR/IDLE MODE bit (SCICCR, bit 3). Both modes use the TXWAKE flag bit (SCICTL1, bit 3), RXWAKE flag bit (SCIRXST, bit1), and the SLEEP flag bit (SCICTL1, bit 2) to control the SCI transmitter and receiver features of these modes.

1.2.4.3 Receipt Sequence

In both multiprocessor modes, the receive sequence is:

1) At the receipt of an address block, the SCI port wakes up and requests an interrupt (bit number 1 RX/BK INT ENA-of SCICTL2 must be enabled to request an interrupt). It reads the first frame of the block, which contains the destination address.

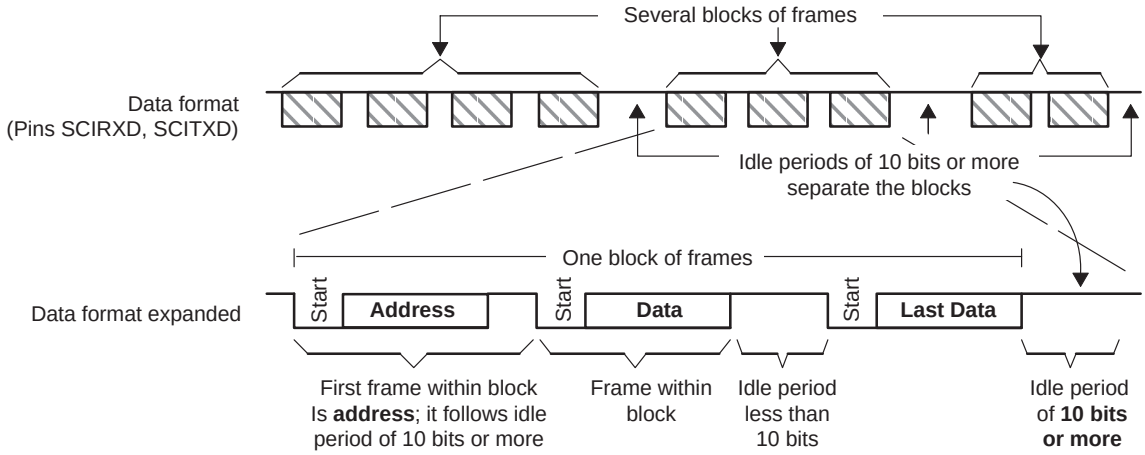
2) A software routine is entered through the interrupt and checks the incoming address. This address byte is checked against its device address byte stored in memory.

3) If the check shows that the block is addressed to the device CPU, the CPU clears the SLEEP bit and reads the rest of the block; if not, the software routine exits with the SLEEP bit still set and does not receive interrupts until the next block start.

1.2.5 Idle-Line Multiprocessor Mode

In the idle-line multiprocessor protocol (ADDR/IDLE MODE bit=0), blocks are separated by having a longer idle time between the blocks than between frames in the blocks. An idle time of ten or more high-level bits after a frame indicates the start of a new block. The time of a single bit is calculated directly from the baud value (bits per second). The idle-line multiprocessor communication format is shown in Figure 1−4 (ADDR/IDLE MODE bit is bit 3 of SCICCR).

Figure 1−4. Idle-Line Multiprocessor Communication Format



1.2.5.1 Idle-Line Mode Steps

The steps followed by the idle-line mode:

1) SCI wakes up after receipt of the block-start signal.

2) The processor recognizes the next SCI interrupt.

3) The interrupt service routine compares the received address (sent by a remote transmitter) to its own.

4) If the CPU *is being addressed*, the service routine clears the SLEEP bit and receives the rest of the data block.

5) If the CPU *is not being addressed*, the SLEEP bit remains set. This lets the CPU continue to execute its main program without being interrupted by the SCI port until the next detection of a block start.

1.2.5.2 Block Start Signal

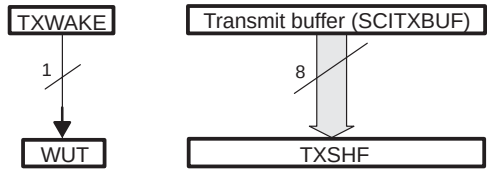
There are two ways to send a block-start signal:

- Method 1: Deliberately leave an idle time of ten bits or more by delaying the time between the transmission of the last frame of data in the previous block and the transmission of the address frame of the new block.
- Method 2: The SCI port first sets the TXWAKE bit (SCICTL1, bit 3) to 1 before writing to the SCITXBUF register. This sends an idle time of exactly 11 bits. In this method, the serial communications line is not idle any longer than necessary. (A don't care byte has to be written to SCITXBUF after setting TXWAKE, and before sending the address, so as to transmit the idle time.)

1.2.5.3 Wake-UP Temporary (WUT) Flag

Associated with the TXWAKE bit is the wake-up temporary (WUT) flag. WUT is an internal flag, double-buffered with TXWAKE. When TXSHF is loaded from SCITXBUF, WUT is loaded from TXWAKE, and the TXWAKE bit is cleared to 0. This arrangement is shown in Figure 1−5.

Figure 1−5. Double-Buffered WUT and TXSHF



Note: WUT = wake-up temporary

Sending a Block Start Signal

To send out a block-start signal of exactly one frame time during a sequence of block transmissions:

1) Write a 1 to the TXWAKE bit.
2) Write a data word (content not important: a *don't care*) to the SCITXBUF register (transmit data buffer) to send a block-start signal. (The first data word written is suppressed while the block-start signal is sent out and ignored after that.) When the TXSHF (transmit shift register) is free again, SCITXBUF contents are shifted to TXSHF, the TXWAKE value is shifted to WUT, and then TXWAKE is cleared.

Because TXWAKE was set to a 1, the start, data, and parity bits are replaced by an idle period of 11 bits transmitted following the last stop bit of the previous frame.

3) Write a new address value to SCITXBUF.

A *don't-care* data word must first be written to register SCITXBUF so that the TXWAKE bit value can be shifted to WUT. After the don't-care data word is shifted to the TXSHF register, the SCITXBUF (and TXWAKE if necessary) can be written to again because TXSHF and WUT are both double-buffered.

1.2.5.4 Receiver Operation

The receiver operates regardless of the SLEEP bit. However, the receiver neither sets RXRDY nor the error status bits, nor does it request a receive interrupt *until an address frame is detected*.

1.2.6 Address-Bit Multiprocessor Mode

In the address-bit protocol (ADDR/IDLE MODE bit=1), frames have an extra bit called an address bit that immediately follows the last data bit. The address bit is set to 1 in the first frame of the block and to 0 in all other frames. The idle period timing is irrelevant (see Figure 1−6, ADDR/IDLE MODE bit in SCICCR, bit 3).

1.2.6.1 Sending an Address

The TXWAKE bit value is placed in the address bit. During transmission, when the SCITXBUF register and TXWAKE are loaded into the TXSHF register and WUT respectively, TXWAKE is reset to 0 and WUT becomes the value of the address bit of the current frame. Thus, to send an address:

1) Set the TXWAKE bit to 1 and write the appropriate address value to the SCITXBUF register.

   When this address value is transferred to the TXSHF register and shifted out, its address bit is sent as a 1. This flags the other processors on the serial link to read the address.

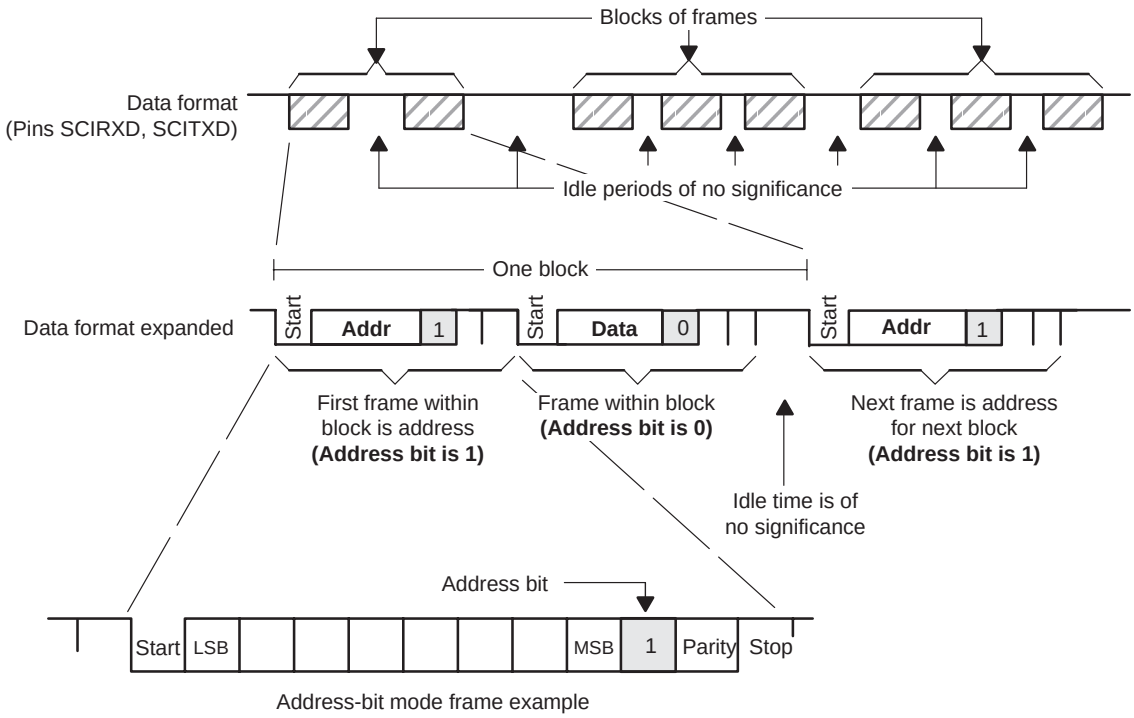
2) Write to SCITXBUF and TXWAKE after TXSHF and WUT are loaded. (Can be written to immediately since both TXSHF and WUT are both double-buffered.

3) Leave the TXWAKE bit set to 0 to transmit non-address frames in the block.

Note: The Address-bit format is for transfers of 11 bytes or less

As a general rule, the address-bit format is typically used for data frames of 11 bytes or less. This format adds one bit value (1 for an address frame, 0 for a data frame) to all data bytes transmitted. The idle-line format is typically used for data frames of 12 bytes or more.

Figure 1−6. Address-Bit Multiprocessor Communication Format



1.2.7 SCI Communication Format

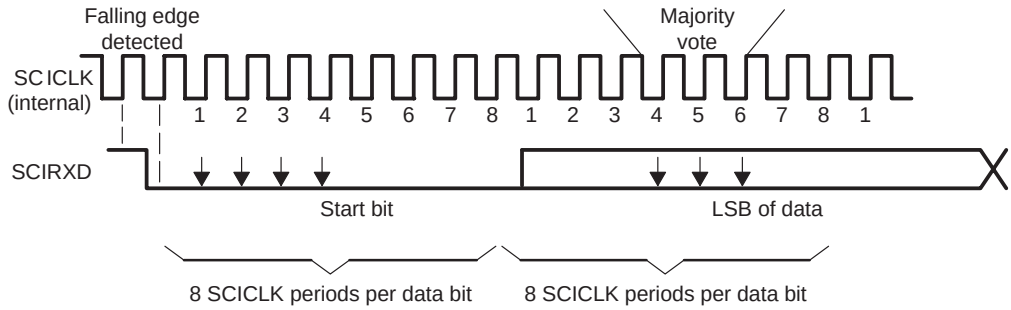
The SCI asynchronous communication format uses either single line (one way) or two line (two way) communications. In this mode, the frame consists of a start bit, one to eight data bits, an optional even/odd parity bit, and one or two stop bits (shown in Figure 1−7). There are *eight SCICLK periods* per data bit.

The receiver begins operation on receipt of a valid start bit. A valid start bit is identified by four consecutive internal SCICLK periods of zero bits as shown in Figure 1−7. If any bit is not zero, then the processor starts over and begins looking for another start bit.

For the bits following the start bit, the processor determines the bit value by making three samples in the middle of the bits. These samples occur on the fourth, fifth, and sixth SCICLK periods, and bit-value determination is on a majority (two out of three) basis. Figure 1−7 illustrates the asynchronous communication format for this with a start bit showing how edges are found and where a majority vote is taken.

Since the receiver synchronizes itself to frames, the external transmitting and receiving devices do not have to use a synchronized serial clock. The clock can be generated locally.

Figure 1−7. SCI Asynchronous Communications Format

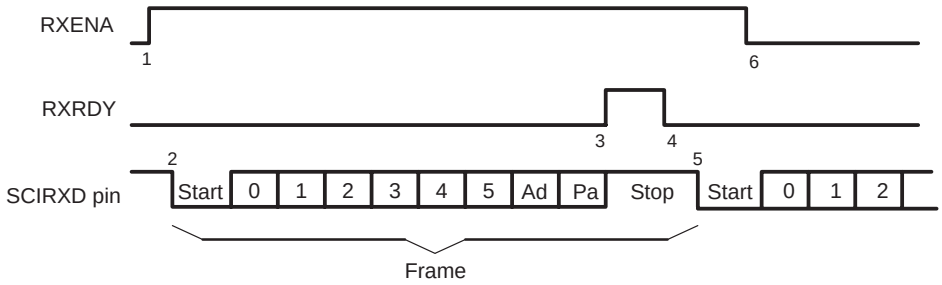


1.2.7.1 Receiver Signals in Communication Modes

Figure 1−8 illustrates an example of receiver signal timing that assumes the following conditions:

- ❑ Address-bit wake-up mode (address bit does not appear in idle-line mode)
- ❑ Six bits per character

Figure 1−8. SCI RX Signals in Communication Modes



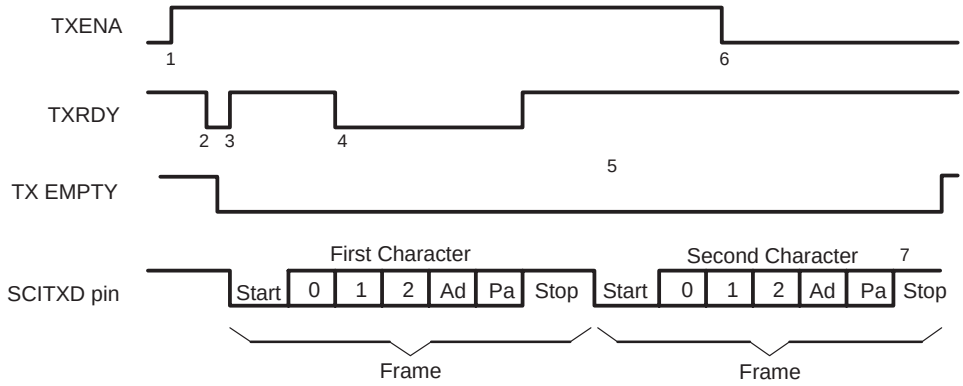
Notes:
1) Flag bit RXENA (SCICTL1, bit 0) goes high to enable the receiver.
2) Data arrives on the SCIRXD pin, start bit detected.
3) Data is shifted from RXSHF to the receiver buffer register (SCIRXBUF); an interrupt is requested. Flag bit RXRDY (SCIRXST, bit 6) goes high to signal that a new character has been received.
4) The program reads SCIRXBUF; flag RXRDY is automatically cleared.
5) The next byte of data arrives on the SCIRXD pin; the start bit is detected, then cleared.
6) Bit RXENA is brought low to disable the receiver. Data continues to be assembled in RXSHF but is not transferred to the receiver buffer register.

1.2.7.2 Transmitter Signals in Communication Modes

Figure 1−9 illustrates an example of transmitter signal timing that assumes the following conditions:

- ❑ Address-bit wake-up mode (address bit does not appear in idle-line mode)
- ❑ Three bits per character

Figure 1−9. SCI TX Signals in Communications Mode



Notes: 1) Bit TXENA (SCICTL1, bit 1) goes high, enabling the transmitter to send data.

2) SCITXBUF is written to; thus, (1) the transmitter is no longer empty, and (2) TXRDY goes low.

3) The SCI transfers data to the shift register (TXSHF). The transmitter is ready for a second character (TXRDY goes high), and it requests an interrupt (to enable an interrupt, bit TX INT ENA — SCICTL2, bit 0 — must be set).

4) The program writes a second character to SCITXBUF after TXRDY goes high (item 3). (TXRDY goes low again after the second character is written to SCITXBUF.)

5) Transmission of the first character is complete. Transfer of the second character to shift register TXSHF begins.

6) Bit TXENA goes low to disable the transmitter; the SCI finishes transmitting the current character.

7) Transmission of the second character is complete; transmitter is empty and ready for new character.

1.2.8 SCI Port Interrupts

The SCI receiver and transmitter can be interrupt controlled. The SCICTL2 register has one flag bit (TXRDY) that indicates active interrupt conditions, and the SCIRXST register has two interrupt flag bits (RXRDY and BRKDT), plus the RX ERROR interrupt flag which is a logical OR of the FE, OE and PE conditions. The transmitter and receiver have separate interrupt-enable bits. When not enabled, the interrupts are not asserted; however, the condition flags remain active, reflecting transmission and receipt status.

The SCI has independent peripheral interrupt vectors for the receiver and transmitter. Peripheral interrupt requests can be either high priority or low priority. This is indicated by the priority bits which are output from the peripheral to the PIE controller. When both RX and TX interrupt requests are made at the same priority level, the receiver always has higher priority than the transmitter, reducing the possibility of receiver overrun.

The operation of peripheral interrupts is described in the peripheral interrupt expansion controller chapter of the *TMS320x281x, 280x System Control and Interrupts Peripheral Reference Guide* (literature number SPRU078).

- If the RX/BK INT ENA bit (SCICTL2, bit 1) is set, the receiver peripheral interrupt request is asserted when one of the following events occurs:
  - The SCI receives a complete frame and transfers the data in the RXSHF register to the SCIRXBUF register. This action sets the RXRDY flag (SCIRXST, bit 6) and initiates an interrupt.
  - A break detect condition occurs (the SCIRXD is low for ten bit periods following a missing stop bit). This action sets the BRKDT flag bit (SCIRXST, bit 5) and initiates an interrupt.
- If the TX INT ENA bit (SCICTL2.0) is set, the transmitter peripheral interrupt request is asserted whenever the data in the SCITXBUF register is transferred to the TXSHF register, indicating that the CPU can write to SCITXBUF; this action sets the TXRDY flag bit (SCICTL2, bit 7) and initiates an interrupt.

Note:

Interrupt generation due to the RXRDY and BRKDT bits is controlled by the RX/BK INT ENA bit (SCICTL2, bit 1). Interrupt generation due to the RX ERROR bit is controlled by the RX ERR INT ENA bit (SCICTL1, bit 6).

1.2.9 SCI Baud Rate Calculations

The internally generated serial clock is determined by the low-speed peripheral clock LSPCLK) and the baud-select registers. The SCI uses the 16-bit value of the baud-select registers to select one of the 64K different serial clock rates possible for a given LSPCLK.

See the bit descriptions in section 2.4, *Baud-Select Registers*, on page 2-8 for the formula to use when calculating the SCI asynchronous baud. Table 1−4 shows the baud-select values for common SCI bit rates.

Table 1−4. Asynchronous Baud Register Values for Common SCI Bit Rates

	LSPCLK Clock Frequency, 37.5 MHz		
Ideal Baud	**BRR**	**Actual Baud**	**% Error**
2400	1952 (7A0h)	2400	0
4800	976 (3D0h)	4798	−0.04
9600	487 (1E7h)	9606	0.06
19200	243 (F3h)	19211	0.06
38400	121 (79h)	38422	0.06

1.2.10 SCI Enhanced Features

The 28x SCI features autobaud detection and transmit/receive FIFO. The following section explains the FIFO operation.

1.2.10.1 SCI FIFO Description

The following steps explain the FIFO features and help with programming the SCI with FIFOs.

1) Reset. At reset the SCI powers up in standard SCI mode and the FIFO function is disabled. The FIFO registers SCIFFTX, SCIFFRX, and SCIFFCT remain inactive.

2) Standard SCI. The standard F24x SCI modes will work normally with TXINT/RXINT interrupts as the interrupt source for the module.

3) FIFO enable. FIFO mode is enabled by setting the SCIFFEN bit in the SCIFFTX register. SCIRST can reset the FIFO mode at any stage of its operation.

4) Active registers. All the SCI registers and SCI FIFO registers (SCIFFTX, SCIFFRX, and SCIFFCT) are active.

5) Interrupts. FIFO mode has two interrupts; one for transmit FIFO, TXINT and one for receive FIFO,RXINT. RXINT is the common interrupt for SCI FIFO receive, receive error, and receive FIFO overflow conditions. The TXINT of the standard SCI will be disabled and this interrupt will service as SCI transmit FIFO interrupt.

6) Buffers. Transmit and receive buffers are supplemented with two 16 level FIFOs. The transmit FIFO registers are 8 bits wide and receive FIFO registers are 10 bits wide. The one word transmit buffer of the standard SCI functions as a transition buffer between the transmit FIFO and shift register. The one word transmit buffer is loaded from transmit FIFO only after the last bit of the shift register is shifted out. With the FIFO enabled, TXSHF is directly loaded after an optional delay value (SCIFFCT), TXBUF is not used.

7) Delayed transfer. The rate at which words in the FIFO are transferred to the transmit shift register is programmable. The SCIFFCT register bits (7−0) FFTXDLY7−FFTXDLY0 define the delay between the word transfer. The delay is defined in the number SCI baud clock cycles. The 8 bit register can define a minimum delay of 0 baud clock cycles and a maximum of 256-baud clock cycles. With zero delay, the SCI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 256 clock delay the SCI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 256 baud clocks between each words. The programmable delay facilitates communication with slow SCI/UARTs with little CPU intervention.

8) FIFO status bits. Both the transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12− 0) that define the number of words available in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO reset the FIFO pointers to zero when these bits *are cleared to 0*. The FIFOs resumes operation from start once these bits are *set to one.*

9) Programmable interrupt levels. Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12−8) match (less than or equal to) the interrupt trigger level bits TXFFIL (bits 4−0). This provides a programmable interrupt trigger for transmit and receive sections of the SCI. Default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO respectively.

   Figure 1−10 and Table 1−5 explain the operation/configuration of SCI interrupts in nonFIFO/FFO mode.

Figure 1−10. SCI FIFO Interrupt Flags and Enable Logic

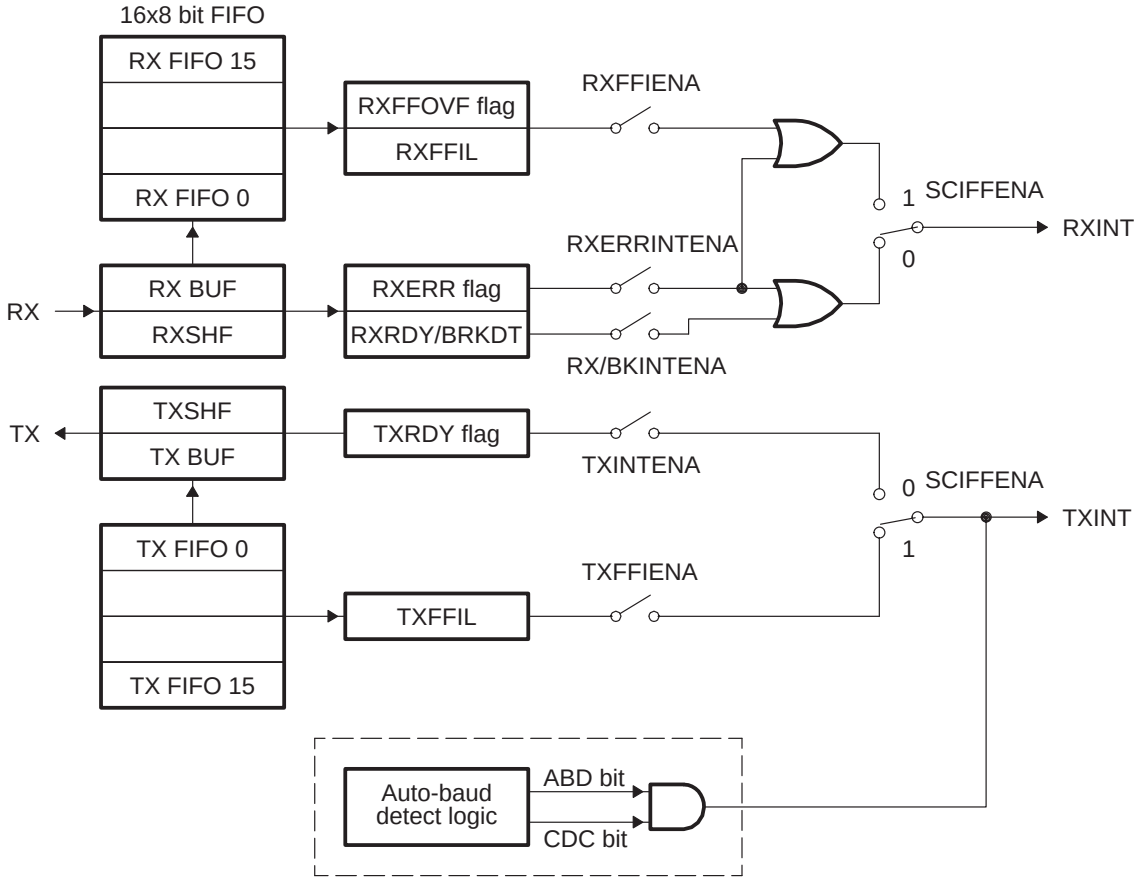


Table 1−5. SCI Interrupt Flags

FIFO Options	SCI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SCIFFENA	Interrupt line
SCI without FIFO	Receive error	RXERR	RXERRINTENA	0	RXINT
	Receive break	BRKDT	RX/BKINTENA	0	RXINT
	Data receive	RXRDY	RX/BKINTENA	0	RXINT
	Transmit empty	TXRDY	TXINTENA	0	TXINT
SCI with FIFO	Receive error and receive break	RXERR	RXERRINTENA	1	RXINT
	FIFO receive	RXFFIL	RXFFIENA	1	RXINT
	Transmit empty	TXFFIL	TXFFIENA	1	TXINT
Auto-baud	Auto-baud detected	ABD	Don’t care	x	TXINT

Notes: 1) RXERR can be set by BRKDT, FE, OE, PE flags. In FIFO mode, BRKDT interrupt is only through RXERR flag
2) FIFO mode TXSHF is directly loaded after delay value, TXBUF is not used.

1.2.10.2 SCI Auto-Baud

Most SCI modules do not have an auto-baud detect logic built-in hardware. These SCI modules are integrated with embedded controllers whose clock rates are dependent on PLL reset values. Often embedded controller clocks change after final design. In the enhanced feature set this module supports an autobaud-detect logic in hardware. The following section explains the enabling sequence for autobaud-detect feature.

1.2.10.3 Autobaud-Detect Sequence

Bits ABD and CDC in SCIFFCT control the autobaud logic. The SCIRST bit should be enabled to make autobaud logic work.

If ABD is set while CDC is 1, which indicates auto-baud alignment, SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit has to be cleared by software. If CDC remains set even after interrupt service, there should be no repeat interrupts.

Step 1: Enable autobaud-detect mode for the SCI by setting the CDC bit (bit 13) in SCIFFCT and clearing the ABD bit (Bit 15) by writing a 1 to ABDCLR bit (bit 14).

Step 2: Initialize the baud register to be 1 or less than a baud rate limit of 500 Kbps.

Step 3: Allow SCI to receive either character ”A” or ”a” from a host at the desired baud rate. If the first character is either ”A” or ”a”. the auto-baud-detect hardware will detect the incoming baud rate and set the ABD bit.

Step 4: The auto-detect hardware will update the baud rate register with the equivalent baud value hex. The logic will also generate an interrupt to the CPU.

Step 5: Respond to the interrupt clear ADB bit by writing a 1 to ABD CLR (bit 14) of SCIFFCT register and disable further autobaud locking by clearing CDC bit by writing a 0.

Step 6: Read the receive buffer for character ”A” or ”a” to empty the buffer and buffer status.

Step 7: If ABD is set while CDC is 1, which indicates autobaud alignment, the SCI transmit FIFO interrupt will occur (TXINT). After the interrupt service CDC bit must be cleared by software.

Note:

At higher baud rates, the slew rate of the incoming data bits can be affected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable autobaud detection at higher baud rates (typically beyond 100k baud) and cause the auto-baud-lock feature to fail.

To avoid this, the followng is recommended:

- ❑ Achieve a baud-lock between the host and 28x SCI boot loader using a lower baud rate.
- ❑ The host may then handshake with the loaded 28x application to set the SCI baud rate register to the desired higher baud rate.

This page is intentionally left blank.

Chapter 2

SCI Registers

The functions of the SCI are software configurable. Sets of control bits, organized into dedicated bytes, are programmed to initialize the desired SCI communications format. This includes operating mode and protocol, baud value, character length, even/odd parity or no parity, number of stop bits, and interrupt priorities and enables.

Topic **Page**

2.1 SCI Module Register Summary

The SCI is controlled and accessed through registers listed in Table 2−1 and Table 2−2, which are described in the sections that follow.

Table 2−1. SCIA Registers

Register Mnemonic	Address	Number of Bits	Description
SCICCR	0x0000−7050	1	SCI-A Communications Control Register
SCICTL1	0x0000−7051	1	SCI-A Control Register 1
SCIHBAUD	0x0000−7052	1	SCI-A Baud Register, High Bits
SCILBAUD	0x0000−7053	1	SCI-A Baud Register, Low Bits
SCICTL2	0x0000−7054	1	SCI-A Control Register 2
SCIRXST	0x0000−7055	1	SCI-A Receive Status Register
SCIRXEMU	0x0000−7056	1	SCI-A Receive Emulation Data Buffer Register
SCIRXBUF	0x0000−7057	1	SCI-A Receive Data Buffer Register
SCITXBUF	0x0000−7059	1	SCI-A Transmit Data Buffer Register
SCIFFTX	0x0000−705A	1	SCI-A FIFO Transmit Register
SCIFFRX	0x0000−705B	1	SCI-A FIFO Receive Register
SCIFFCT	0x0000−705C	1	SCI-A FIFO Control Register
SCIPRI	0x0000−705F	1	SCI-A Priority Control Register

Note: The shaded registers operate in enhanced mode.

Table 2−2. SCIB Registers

Name	Address Range	Number of Bits	Description
SCICCR	0x0000−7750	1	SCI-B Communications Control Register
SCICTL1	0x0000−7751	1	SCI-B Control Register 1
SCIHBAUD	0x0000−7752	1	SCI-B Baud Register, High Bits
SCILBAUD	0x0000−7753	1	SCI-B Baud Register, Low Bits
SCICTL2	0x0000−7754	1	SCI-B Control Register 2
SCIRXST	0x0000−7755	1	SCI-B Receive Status Register
SCIRXEMU	0x0000−7756	1	SCI-B Receive Emulation Data Buffer Register
SCIRXBUF	0x0000−7757	1	SCI-B Receive Data Buffer Register
SCITXBUF	0x0000−7759	1	SCI-B Transmit Data Buffer Register
SCIFFTX	0x0000−775A	1	SCI-B FIFO Transmit Register
SCIFFRX	0x0000−775B	1	SCI-B FIFO Receive Register
SCIFFCT	0x0000−775C	1	SCI-B FIFO Control Register
SCIPRI	0x0000−775F	1	SCI-B Priority Control Register

2.2 SCI Communication Control Register (SCICCR)

SCICCR defines the character format, protocol, and communications mode used by the SCI.

Figure 2−1. SCI Communication Control Register (SCICCR) — Address 7050h

7	6	5	4	3	2	1	0
STOP BITS	EVEN/ODD PARITY	PARITY ENABLE	LOOPBACK ENA	ADDR/IDLE MODE	SCICHAR2	SCICHAR1	SCICHAR0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−3. SCI Communication Control Register (SCICCR) Bit Descriptions

Bit(s)	Name	Description
7	STOP BITS	SCI number of stop bits. This bit specifies the number of stop bits transmitted. The receiver checks for only one stop bit. 1 Two stop bits 0 One stop bit
6	PARITY	SCI parity odd/even selection. If the PARITY ENABLE bit (SCICCR, bit 5) is set, PARITY (bit 6) designates odd or even parity (odd or even number of bits with the value of 1 in both transmitted and received characters). 1 Even parity 0 Odd parity
5	PARITY ENABLE	SCI parity enable. This bit enables or disables the parity function. If the SCI is in the address-bit multiprocessor mode (set using bit 3 of this register), the address bit is included in the parity calculation (if parity is enabled). For characters of less than eight bits, the remaining unused bits should be masked out of the parity calculation. 1 Parity is enabled 0 Parity disabled; no parity bit is generated during transmission or is expected during reception
4	LOOP BACK ENA	Loop Back test mode enable. This bit enables the Loop Back test mode where the Tx pin is internally connected to the Rx pin. 1 Loop Back test mode enabled 0 Loop Back test mode disabled

Table 2−3. SCI Communication Control Register (SCICCR) Bit Descriptions (Continued)

Bit(s)	Name	Description
3	ADDR/IDLE MODE	SCI multiprocessor mode control bit. This bit selects one of the multiprocessor protocols Multiprocessor communication is different from the other communication modes because it uses SLEEP and TXWAKE functions (bits SCICTL1, bit 2 and SCICTL1, bit 3, respectively). The idle-line mode is usually used for normal communications because the address-bit mode adds an extra bit to the frame. The idle-line mode does not add this extra bit and is compatible with RS-232 type communications. 1 Address-bit mode protocol selected 0 Idle-line mode protocol selected
2−0	SCI CHAR2−0	Character-length control bits 2−0. These bits select the SCI character length from one to eight bits. Characters of less than eight bits are right-justified in SCIRXBUF and SCIRXEMU and are padded with leading zeros in SCIRXBUF. SCITXBUF *doesn't need to be padded with leading zeros.* The bit values and character lengths for SCI CHAR2-0 bits are as follows:

SCI CHAR2−0 Bit Values (Binary)

SCI CHAR2	SCI CHAR1	SCI CHAR0	Character Length (Bits)
0	0	0	1
0	0	1	2
0	1	0	3
0	1	1	4
1	0	0	5
1	0	1	6
1	1	0	7
1	1	1	8

2.3 SCI Control Register 1 (SCICTL1)

SCICTL1 controls the receiver/transmitter enable, TXWAKE and SLEEP functions, and the SCI software reset.

Figure 2−2. SCI Control Register 1 (SCICTL1) — Address 7051h

7	6	5	4	3	2	1	0
Reserved	RX ERR INT ENA	SW RESET	Reserved	TXWAKE	SLEEP	TXENA	RXENA
R-0	R/W-0	R/W-0	R-0	R/S-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−4. SCI Control Register 1 (SCICTL1) Bit Descriptions

Bit(s)	Name	Description
7	Reserved	Reads return zero; writes have no effect.
6	RX ERR INT ENA	SCI receive error interrupt enable. Setting this bit enables an interrupt if the RX ERROR bit (SCIRXST, bit 7) becomes set because of errors occurring. 1 Receive error interrupt enabled 0 Receive error interrupt disabled
5	SW RESET	SCI software reset (active low). Writing a 0 to this bit initializes the SCI state machines and operating flags (registers SCICTL2 and SCIRXST) to the reset condition. The SW RESET bit does not affect any of the configuration bits. All affected logic is held in the specified reset state until a 1 is written to SW RESET (the bit values following a reset are shown beneath each register diagram in this section). Thus, after a system reset, re-enable the SCI by writing a 1 to this bit. Clear this bit after a receiver break detect (BRKDT flag, bit SCIRXST, bit 5). SW RESET affects the operating flags of the SCI, but it neither affects the configuration bits nor restores the reset values. Once SW RESET is asserted, the flags are frozen until the bit is deasserted.

Table 2−4. SCI Control Register 1 (SCICTL1) Bit Descriptions (Continued)

Bit(s)	Name	Description
		The affected flags are as follows:

SCI Flag	Register Bit	Value After SW RESET
TXRDY	SCICTL2, bit 7	1
TX EMPTY	SCICTL2, bit 6	0
RXWAKE	SCIRXST, bit 1	0
PE	SCIRXST, bit 2	0
OE	SCIRXST, bit 3	0
FE	SCIRXST, bit 4	0
BRKDT	SCIRXST, bit 5	0
RXRDY	SCIRXST, bit 6	0
RX ERROR	SCIRXST, bit 7	

4	Reserved	Reads return zero; writes have no effect.
3	TXWAKE	SCI transmitter wake-up method select. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3) 1 Transmit feature selected is dependent on the mode, idle-line or address-bit: 0 Transmit feature is not selected In *idle-line* mode: write a 1 to TXWAKE, then write data to register SCITXBUF to generate an idle period of 11 data bits In *address-bit* mode: write a 1 to TXWAKE, then write data to SCITXBUF to set the address bit for that frame to 1 TXWAKE is not cleared by the SW RESET bit (SCICTL1, bit 5); it is cleared by a system reset or the transfer of TXWAKE to the WUT flag.
2	SLEEP	SCI sleep. The TXWAKE bit controls selection of the data-transmit feature, depending on which transmit mode (idle-line or address-bit) is specified at the ADDR/IDLE MODE bit (SCICCR, bit 3). In a multiprocessor configuration, this bit controls the receiver sleep function. Clearing this bit brings the SCI out of the sleep mode. The receiver still operates when the SLEEP bit is set; however, operation does not update the receiver buffer ready bit (SCIRXST, bit 6, RXRDY) or the error status bits (SCIRXST, bit 5−2: BRKDT, FE, OE, and PE) unless the address byte is detected. SLEEP is *not* cleared when the address byte is detected. 1 Sleep mode enabled 0 Sleep mode disabled

Table 2−4. SCI Control Register 1 (SCICTL1) Bit Descriptions (Continued)

Bit(s)	Name	Description
1	TXENA	SCI transmitter enable. Data is transmitted through the SCITXD pin only when TXENA is set. If reset, transmission is halted but only after all data previously written to SCITXBUF has been sent. 1 Transmitter enabled 0 Transmitter disabled
0	RXENA	SCI receiver enable. Data is received on the SCIRXD pin and is sent to the receiver shift register and then the receiver buffers. This bit enables or disables the receiver (transfer to the buffers). Clearing RXENA stops received characters from being transferred to the two receiver buffers and also stops the generation of receiver interrupts. However, the receiver shift register can continue to assemble characters. Thus, if RXENA is set during the reception of a character, the complete character will be transferred into the receiver buffer registers, SCIRXEMU and SCIRXBUF. 1 Send received characters to SCIRXEMU and SCIRXBUF 0 Prevent received characters from transfer into the SCIRXEMU and SCIRXBUF receiver buffers

2.4 SCI Baud-Select Registers (SCIHBAUD, SCILBAUD)

The values in SCIHBAUD and SCILBAUD specify the baud rate for the SCI.

Figure 2−3. Baud-Select MSbyte Register (SCIHBAUD) — Address 7052h

15	14	13	12	11	10	9	8
BAUD15 (MSB)	BAUD14	BAUD13	BAUD12	BAUD11	BAUD10	BAUD9	BAUD8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Figure 2−4. Baud-Select LSbyte Register (SCILBAUD) — Address 7053h

7	6	5	4	3	2	1	0
BAUD7	BAUD6	BAUD5	BAUD4	BAUD3	BAUD2	BAUD1	BAUD0 (LSB)
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -n = value after reset

Table 2−5. Baud-Select Register Bit Descriptions

Bit(s)	Name	Reset	Description
15−0	BAUD15−BAUD0	0	SCI 16-bit baud selection Registers SCIHBAUD (MSbyte) and SCILBAUD (LSbyte) are concatenated to form a 16-bit baud value, BRR. The internally-generated serial clock is determined by the low speed peripheral clock (LSPCLK) signal and the two baud-select registers. The SCI uses the 16-bit value of these registers to select one of 64K serial clock rates for the communication modes. The SCI baud rate is calculated using the following equation: $\text{SCI Asynchronous Baud} = \frac{\text{LSPCLK}}{(\text{BRR} + 1) \times 8}$ Alternatively, $\text{BRR} = \frac{\text{LSPCLK}}{\text{SCI Asynchronous Baud} \times 8} - 1$ Note that the above formulas are applicable only when $1 \le \text{BRR} \le 65535$. If BRR = 0, then $\text{SCI Asynchronous Baud} = \frac{\text{LSPCLK}}{16}$ Where: BRR = the 16-bit value (in decimal) in the baud-select registers.

2.5 SCI Control Register 2 (SCICTL2)

SCICTL2 enables the receive-ready, break-detect, and transmit-ready interrupts as well as transmitter-ready and -empty flags.

Figure 2−5. SCI Control Register 2 (SCICTL2) — Address 7054h

7	6	5 2	1	0
TXRDY	TX EMPTY	Reserved	RX/BK INT ENA	TX INT ENA
R-1	R-1	R-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -n = value after reset

Table 2−6. SCI Control Register 2 (SCICTL2) Bit Descriptions

Bit(s)	Name	Description
7	TXRDY	Transmitter buffer register ready flag. When set, this bit indicates that the transmit data buffer register, SCITXBUF, is ready to receive another character. Writing data to the SCITXBUF automatically clears this bit. When set, this flag asserts a transmitter interrupt request if the interrupt-enable bit, TX INT ENA (SCICTL2.0), is also set. TXRDY is set to 1 by enabling the SW RESET bit (SCICTL.2) or by a system reset. 1 SCITXBUF is ready to receive the next character 0 SCITXBUF is full
6	TX EMPTY	Transmitter empty flag. This flag's value indicates the contents of the transmitter's buffer register (SCITXBUF) and shift register (TXSHF). An active SW RESET (SCICTL1.2), or a system reset, sets this bit. This bit *does not* cause an interrupt request. 1 Transmitter buffer and shift registers are both empty 0 Transmitter buffer or shift register or both are loaded with datal
5−2	Reserved	
1	RX/BK INT ENA	Receiver-buffer/break interrupt enable. This bit controls the interrupt request caused by *either* the RXRDY flag *or* the BRKDT flag (bits SCIRXST.6 and .5) being set. However, RX/BK INT ENA does not prevent the setting of these flags. 1 Enable RXRDY/BRKDT interrupt 0 Disable RXRDY/BRKDT interrupt
0	TX INT ENA	SCITXBUF-register interrupt enable.This bit controls the interrupt request caused by setting the TXRDY flag bit (SCICTL2.7). However, it does not prevent the TXRDY flag from being set (being set indicates that register SCITXBUF is ready to receive another character). 1 Enable TXRDY interrupt 0 Disable TXRDY interrupt

2.6 SCI Receiver Status Register (SCIRXST)

SCIRXST contains seven bits that are receiver status flags (two of which can generate interrupt requests). Each time a complete character is transferred to the receiver buffers (SCIRXEMU and SCIRXBUF), the status flags are updated. Figure 2−7 on page 2-12 shows the relationships between several of the register's bits.

Figure 2−6. SCI Receiver Status Register (SCIRXST) — Address 7055h

7	6	5	4	3	2	1	0
RX ERROR	RXRDY	BRKDT	FE	OE	PE	RXWAKE	Reserved
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read access, W = Write access, -n = value after reset

Table 2−7. SCI Receiver Status Register (SCIRXST) Bit Descriptions

Bit(s)	Name	Description
7	RX ERROR	SCI receiver error flag. The RX ERROR flag indicates that one of the error flags in the receiver status register is set. RX ERROR is a logical OR of the break detect, framing error, overrun, and parity error enable flags (bits 5−2: BRKDT, FE, OE, and PE). A 1 on this bit will cause an interrupt if the RX ERR INT ENA bit (SCICTL1.6) is set. This bit can be used for fast error-condition checking during the interrupt service routine. This error flag cannot be cleared directly; it is cleared by an active SW RESET or by a system reset. 1 Error flag(s) set 0 No error flags set
6	RXRDY	SCI receiver-ready flag. When a new character is ready to be read from the SCIRXBUF register, the receiver sets this bit, and a receiver interrupt is generated if the RX/BK INT ENA bit (SCICTL2.1) is a 1. RXRDY is cleared by a reading of the SCIRXBUF register, by an active SW RESET, or by a system reset. 1 Character ready to be read from SCIRXBUF 0 No new character in SCIRXBUF

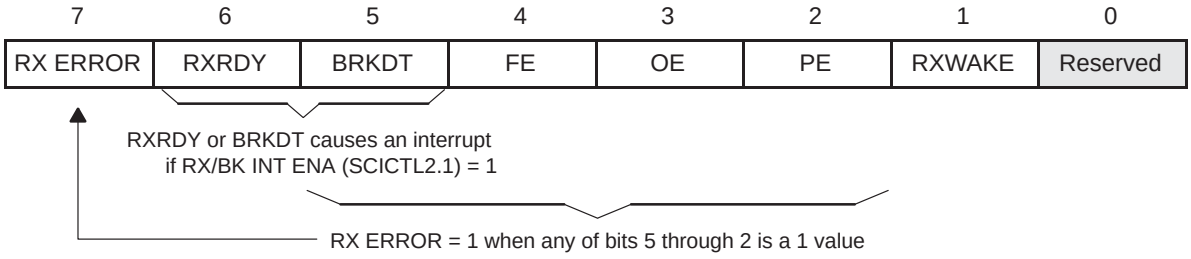
Table 2−7. SCI Receiver Status Register (SCIRXST) Bit Descriptions (Continued)

Bit(s)	Name	Description
5	BRKDT	SCI break-detect flag. The SCI sets this bit when a break condition occurs. A break condition occurs when the SCI receiver data line (SCIRXD) remains continuously low for at least ten bits, beginning after a missing first stop bit. The occurrence of a break causes a receiver interrupt to be generated if the RX/BK INT ENA bit is a 1, but it does not cause the receiver buffer to be loaded. A BRKDT interrupt can occur even if the receiver SLEEP bit is set to 1. BRKDT is cleared by an active SW RESET or by a system reset. It is not cleared by receipt of a character after the break is detected. In order to receive more characters, the SCI must be reset by toggling the SW RESET bit or by a system reset. 1 Break condition occurred 0 No break condition
4	FE	SCI framing-error flag. The SCI sets this bit when an expected stop bit is not found. Only the first stop bit is checked. The missing stop bit indicates that synchronization with the start bit has been lost and that the character is incorrectly framed. The FE bit is reset by a clearing of the SW RESET bit or by a system reset. 1 Framing error detected 0 No framing error detected
3	OE	SCI overrun-error flag. The SCI sets this bit when a character is transferred into registers SCIRXEMU and SCIRXBUF before the previous character is fully read by the CPU or DMAC. The previous character is overwritten and lost. The OE flag bit is reset by an active SW RESET or by a system reset. 1 Overrun error detected 0 No overrun error detected
2	PE	SCI parity-error flag. This flag bit is set when a character is received with a mismatch between the number of 1s and its parity bit. The address bit is included in the calculation. If parity generation and detection is not enabled, the PE flag is disabled and read as 0. The PE bit is reset by an active SW RESET or a system reset. 1 Parity error is detected 0 No parity error **or** parity is disabled

Table 2−7. SCI Receiver Status Register (SCIRXST) Bit Descriptions (Continued)

Bit(s)	Name	Description
1	RXWAKE	Receiver wake-up-detect flag. A value of 1 in this bit indicates detection of a receiver wake-up condition. In the address-bit multiprocessor mode (SCICCR.3 = 1), RXWAKE reflects the value of the address bit for the character contained in SCIRXBUF. In the idle-line multiprocessor mode, RXWAKE is set if the SCIRXD data line is detected as idle. RXWAKE is a read-only flag, cleared by one of the following: ❑ The transfer of the first byte after the address byte to SCIRXBUF ❑ The reading of SCIRXBUF ❑ An active SW RESET ❑ A system reset
0	Reserved	Reads return zero; writes have no effect.

Figure 2−7. Register SCIRXST Bit Associations — Address 7055h



2.7 Receiver Data Buffer Registers (SCIRXEMU, SCIRXBUF)

Received data is transferred from RXSHF to SCIRXEMU and SCIRXBUF. When the transfer is complete, the RXRDY flag (bit SCIRXST.6) is set, indicating that the received data is ready to be read. Both registers contain the same data; they have separate addresses but are not physically separate buffers. The only difference is that reading SCIRXEMU *does not* clear the RXRDY flag; however, reading SCIRXBUF clears the flag.

2.7.1 Emulation Data Buffer (SCIRXEMU)

Normal SCI data-receive operations read the data received from the SCIRXBUF register. The SCIRXEMU register is used principally by the emulator (EMU) because it can continuously read the data received for screen updates without clearing the RXRDY flag. SCIRXEMU is cleared by a system reset.

This is the register that should be used in an emulator watch window to view the contents of the SCIRXBUF register.

SCIRXEMU is not physically implemented; it is just a different address location to access the SCIRXBUF register without clearing the RXRDY flag.

Figure 2−8. Emulation Data Buffer Register (SCIRXEMU) — Address 7056h

7	6	5	4	3	2	1	0
ERXDT7	ERXDT6	ERXDT5	ERXDT4	ERXDT3	ERXDT2	ERXDT1	ERXDT0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read access, -0 = value after reset

2.7.2 Receiver Data Buffer (SCIRXBUF)

When the current data received is shifted from RXSHF to the receiver buffer, flag bit RXRDY is set and the data is ready to be read. If the RX/BK INT ENA bit (SCICTL2.1) is set, this shift also causes an interrupt. When SCIRXBUF is read, the RXRDY flag is reset. SCIRXBUF is cleared by a system reset.

Figure 2−9. SCI Receive Data Buffer Register (SCIRXBUF) — Address 7057h

15	14	13					8
SCIFFFE	SCIFFPE	Reserved					
R−0	R−0	R−0					

7	6	5	4	3	2	1	0
RXDT7	RXDT6	RXDT5	RXDT4	RXDT3	RXDT2	RXDT1	RXDT0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read access, W = Write access, -n = value after reset

Note: Shaded area is applicable only if the FIFO is enabled.

Table 2−8. SCI Receive Data Buffer Register (SCIRXBUF) Bit Descriptions

Bit(s)	Name	Description
15	SCIFFFE	SCIFFFE. SCI FIFO Framing error flag bit
		1 A frame error occurred while receiving the character in bits 7−0. This bit is associated with the character on the top of the FIFO.
		0 No frame error occurred while receiving the character, in bits 7−0. This bit is associated with the character on the top of the FIFO.
14	SCIFFPE	SCIFFPE. SCI FIFO parity error flag bit
		1 A parity error occurred while receiving the character in bits 7−0. This bit is associated with the character on the top of the FIFO.
		0 No parity error occurred while receiving the character, in bits 7−0. This bit is associated with the character on the top of the FIFO.
13−8	Reserved	
7−0	RXDT7−0	Receive Character bits

2.8 SCI Transmit Data Buffer Register (SCITXBUF)

Data bits to be transmitted are written to SCITXBUF. These bits must be right-justified because the leftmost bits are ignored for characters less than eight bits long. The transfer of data from this register to the TXSHF transmitter shift register sets the TXRDY flag (SCICTL2.7), indicating that SCITXBUF is ready to receive another set of data. If bit TX INT ENA (SCICTL2.0) is set, this data transfer also causes an interrupt.

Figure 2−10. Transmit Data Buffer Register (SCITXBUF) — Address 7059h

7	6	5	4	3	2	1	0
TXDT7	TXDT6	TXDT5	TXDT4	TXDT3	TXDT2	TXDT1	TXDT0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

2.9 SCI FIFO Registers (SCIFFTX, SCIFFRX, SCIFFCT)

Figure 2−11. SCI FIFO Transmit (SCIFFTX) Register — Address 705Ah

15	14	13	12	11	10	9	8
SCIRST	SCIFFENA	TXFIFO Reset	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R/W-1	R/W−0	R/W−1	R−0	R−0	R−0	R−0	R−0

7	6	5	4	3	2	1	0
TXFFINT Flag	TXFFINT CLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R−0	W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−9. SCI FIFO Transmit (SCIFFTX) Register Bit Descriptions

Bit(s)	Name	Description	
15	SCIRST	0	Write 0 to reset the SCI transmit and receive channels. SCI FIFO register configuration bits will be left as is.
		1	SCI FIFO can resume transmit or receive. SCIRST should be 1 even for Autobaud logic to work.
14	SCIFFENA	0	SCI FIFO enhancements are disabled
		1	SCI FIFO enhancements are enabled
13	TXFIFO Reset	Transmit FIFO reset	
		0	Reset the FIFO pointer to zero and hold in reset
		1	Re-enable transmit FIFO operation
12−8	TXFFST4−0	00000	Transmit FIFO is empty.
		00001	Transmit FIFO has 1 words
		00010	Transmit FIFO has 2 words
		00011	Transmit FIFO has 3 words
		0xxxx	Transmit FIFO has x words
		10000	Transmit FIFO has 16 words
7	TXFFINT	Transmit FIFO interrupt	
		0	TXFIFO interrupt has not occurred, read-only bit
		1	TXFIFO interrupt has occurred, read-only bit
6	TXFFINT CLR	0	Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero
		1	Write 1 to clear TXFFINT flag in bit 7
5	TXFFIENA	0	TX FIFO interrupt based on TXFFIVL match (less than or equal to) is disabled
		1	TX FIFO interrupt based on TXFFIVL match (less than or equal to) is enabled.
4−0	TXFFIL4−0	TXFFIL4−0 Transmit FIFO interrupt level bits. Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4−0) and FIFO level bits (TXFFIL4−0) match (less than or equal to).	
		Default value should be 0x00000.	

Figure 2−12. SCI FIFO Receive (SCIFFRX) Register — Address 705Bh

15	14	13	12	11	10	9	8
RXFFOVF	RXFFOVR CLR	RXFIFO Reset	RXFIFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0	W−0	R/W−1	R−0	R−0	R−0	R−0	R−0

7	6	5	4	3	2	1	0
RXFFINT Flag	RXFFINT CLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−1	R/W−1

Note: R = Read access, W = Write access, -0 = value after reset

Table 2−10. SCI FIFO Receive (SCIFFRX) Register Bit Descriptions

Bit(s)	Name		Description
15	RXFFOVF	Receive FIFO overflow. This will function as flag, but cannot generate interrupt by itself. This condition will occur while receive interrupt is active. Receive interrupts should service this flag condition.	
		0	Receive FIFO has not overflowed, read-only bit
		1	Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost
14	RXFFOVF CLR	0	Write 0 has no effect on RXFFOVF flag bit, Bit reads back a zero
		1	Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFO Reset	Receive FIFO reset	
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable receive FIFO operation
8−12	RXFFST4−0	00000	Receive FIFO is empty
		00001	Receive FIFO has 1 word
		00010	Receive FIFO has 2 words
		00011	Receive FIFO has 3 words
		0xxxx	Receive FIFO has x words
		10000	Receive FIFO has 16 words
7	RXFFINT	Receive FIFO interrupt	
		0	RXFIFO interrupt has not occurred, read-only bit
		1	RXFIFO interrupt has occurred, read-only bit

Table 2−10. SCI FIFO Receive (SCIFFRX) Register Bit Descriptions (Continued)

Bit(s)	Name	Description
6	RXFFINT CLR	Receive FIFO interrupt clear 0 Write 0 has no effect on RXFIFINT flag bit. Bit reads back a zero. 1 Write 1 to clear RXFFINT flag in bit 7
5	RXFFIENA	Receive FIFO interrupt enable 0 RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be disabled 1 RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be enabled.
4−0	RXFFIL4−0	Receive FIFO interrupt level bits Receive FIFO generates interrupt when the FIFO status bits (RXFFST4−0) and FIFO level bits (RXFFIL4−0) match (i.e., are greater than or equal to). Default value of these bits after reset − 11111. This will avoid frequent interrupts, after reset, as the receive FIFO will be empty most of the time.

Figure 2−13. SCI FIFO Control (SCIFFCT) Register — Address 705Ch

15	14	13	12	8
ABD	ABD CLR	CDC	Reserved	
R-0	W−0	R/W−0	R−0	

7	6	5	4	3	2	1	0
FFTXDLY7	FFTXDLY6	FFTXDLY5	FFTXDLY4	FFTXDLY3	FFTXDLY2	FFTXDLY1	FFTXDLY0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

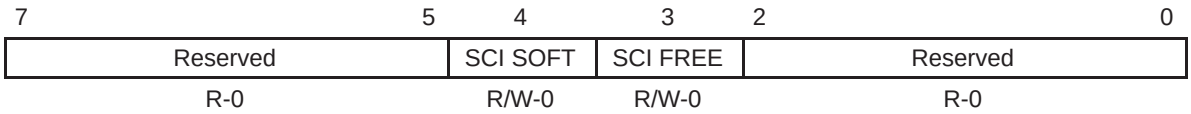
Note: R = Read access, W = Write access, -0 = value after reset

Table 2−11. SCI FIFO Control (SCIFFCT) Register Bit Descriptions

Bit(s)	Name	Description
15	ABD	Auto-baud detect (ABD) bit. 0 Auto-baud detection is not complete. ”A”,”a” character has not been received successfully. 1 Auto-baud hardware has detected ”A” or ”a” character on the SCI receive register. Auto-detect is complete. This will work only if CDC bit is set to enable auto-baud.
14	ABD CLR	ABD-clear bit 0 Write 0 has no effect on ABD flag bit. Bit reads back a zero. 1 Write 1 to clear ABD flag in bit 15.
13	CDC	CDC calibrate A-detect bit 0 Disables auto-baud alignment 1 Enables auto-baud alignment
12−8	Reserved	Reserved
7−0	FFTXDLY7−0	FIFO transfer delay. These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in the number of SCI serial baud clock cycles. The 8 bit register could define a minimum delay of 0 baud clock cycles and a maximum of 256 baud clock cycles In FIFO mode, the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In FIFO mode, TXBUF should not be treated as one additional level of buffer. The delayed transmit feature will help to create an auto-flow scheme without RTS/CTS controls as in standard UARTS.

2.10 Priority Control Register (SCIPRI)

Figure 2−14. SCI Priority Control Register (SCIPRI) — Address 705Fh



Note: R = Read access, W = Write access, -0 = value after reset

Table 2−12. SCI Priority Control Register (SCIPRI) Bit Descriptions

Bit(s)	Name	Description
7−5	Reserved	Reads return zero; writes have no effect.
4−3	SOFT and FREE	These bits determine what occurs when an emulation suspend event occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode), or if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete.
		Bit 4 SOFT / Bit 3 FREE
		0 / 0 — Immediate stop on suspend
		1 / 0 — Complete current receive/transmit sequence before stopping
		x / 1 — Free run. Continues SCI operation regardless of suspend
2−0	Reserved	Reads return zero; writes have no effect.

Appendix A

Revision History

This document was revised to SPRU051B from SPRU051A. The scope of the revisions was limited to technical changes as described in Section A.1. This appendix lists only revisions made in the most recent version.

A.1 Changes Made in This Revision

The following changes were made in this revision:

Global change: Name changed to reflect new devices.

Page	Additions/Modifications/Deletions
2-17	Corrected bit names to RXFFST4−0 from RXFIFST4−0 in Figure 2−12
2-16	Modified description of Bit 14 (SCIFFENA) in Table 2−9

TMS320x281x, 280x DSP Serial Peripheral Interface (SPI) Reference Guide

Literature Number: SPRU059B
June 2002 − Revised November 2004

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third−party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products & application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This guide describes how the serial peripheral interface works on the TMS320x281x and TMS320x280x DSPs.

Related Documentation From Texas Instruments

The following books describe the TMS320x281x and related support tools that are available on the TI website.

TMS320F2801, TMS320F2806, TMS320F2808 Digital Signal Processors (literature number SPRS230) data sheet contains the pinout, signal descriptions, as well as electrical and timing specifications for the F280x devices.

TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, and TMS320C2812 Digital Signal Processors (literature number SPRS174) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320R2811 and TMS320R2812 Digital Signal Processors (literature number SPRS257) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C28x DSP CPU and Instruction Set Reference Guide (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x™ fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320x280x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU716) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage

regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x280x Boot ROM Reference Guide (literature number SPRU722) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x280x Enhanced Capture (eCAP) Module Reference Guide (literature number SPRU807) describes the enhanced Capture Module. It includes the module description and registers.

TMS320x280x Enhanced Pulse Width Modulator (ePWM) Module Reference Guide (literature number SPRU791). The PWM peripheral is an essential part of controlling many of the power related systems found in both commercial and industrial equipments. This guide describes the main areas that include digital motor control, switch mode power supply control, UPS (uninterruptible power supplies), and other forms of power conversion. The PWM peripheral can be considered as performing a DAC function, where the duty cycle is equivalent to a DAC analog value, it is sometimes referred to as a Power DAC.

TMS320x280x Enhanced Quadrature Encoder Pulse (eQEP) Reference Guide (literature number SPRU790) describes the eQEP module, which is used for interfacing with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine in high performance motion and position control systems. It includes the module description and registers.

TMS320x280x System Control and Interrupts Reference Guide (literature number SPRU712) describes the various interrupts and system control features of the 280x digital signal processors (DSPs).

TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide (literature number SPRU074) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

TMS320x281x, 280x Peripheral Reference Guide (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide (literature number SPRU051) describes the SCI that is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

TMS320x281x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU060) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x281x Boot ROM Reference Guide (literature number SPRU095) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x281x Event Manager (EV) Reference Guide (literature number SPRU065) describes the EV modules that provide a broad range of functions and features that are particularly useful in motion control and motor control applications. The EV modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits.

TMS320x281x External Interface (XINTF) Reference Guide (literature number SPRU067) describes the external interface (XINTF) of the 28x digital signal processors (DSPs).

TMS320x281x Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU061) describes the McBSP) available on the C28x devices. The McBSPs allow direct interface between a DSP and other devices in a system.

TMS320x281x System Control and Interrupts Reference Guide (literature number SPRU078) describes the various interrupts and system control features of the 281x digital signal processors (DSPs).

The TMS320C28x Instruction Set Simulator Technical Overview (literature number SPRU608) describes the simulator, available within the

Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU625) describes development using DSP/BIOS.

3.3 V DSP for Digital Motor Control Application Report (literature number SPRA550). New generations of motor control digital signal processors (DSPs) lower their supply voltages from 5 V to 3.3 V to offer higher performance at lower cost. Replacing traditional 5-V digital control circuitry by 3.3-V designs introduce no additional system cost and no significant complication in interfacing with TTL and CMOS compatible components, as well as with mixed voltage ICs such as power transistor gate drivers. Just like 5-V based designs, good engineering practice should be exercised to minimize noise and EMI effects by proper component layout and PCB design when 3.3-V DSP, ADC, and digital circuitry are used in a mixed signal environment, with high and low voltage analog and switching signals, such as a motor control system. In addition, software techniques such as Random PWM method can be used by special features of the Texas Instruments (TI) TMS320x24xx DSP controllers to significantly reduce noise effects caused by EMI radiation.

This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications.The application report first describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V − 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system's noise and EMI effects are summarized in the last section.

Thermo-Electric Cooler Control Using a TMS320F2812 DSP & DRV592 Power Amplifier Application Note (literature number SPRA873). This application report presents a thermoelectric cooler system consisting of a Texas Instruments TMS320F2812 digital signal processor (DSP) and DRV592 power amplifier. The DSP implements a digital proportional-integral-derivative feedback controller using an integrated 12-bit analog-to-digital converter to read the thermistor, and direct output of pulse-width-modulated waveforms to the H-bridge DRV592

power amplifier. The system presented provides up to 6.1 watts of heating or cooling to the laser mount, although the DRV592 amplifier is actually capable of delivering up to 15 watts when configured appropriately. The closed-loop TEC system is seen to achieve ±0.0006°C temperature accuracy, depending on the needed operating temperature range, with a step response settling time of 14 to 16 seconds. A complete description of the experimental system, along with software and software operating instructions, are provided.

Running an Application from Internal Flash Memory on the TMS320F281x DSP Application Report (literature number SPRA958). Several special requirements exist for running an application from on-chip flash memory on the TMS320F28x DSP. These requirements generally do not manifest themselves during development in RAM since the Code Composer Studio™ debugger can mask problems associated with initialized sections and how they are linked to memory. This application report covers the requirements needed to properly configure application software for execution from on-chip flash memory. Requirements for both DSP/BIOS™ and non-DSP/BIOS projects are presented. Some performance considerations and techniques are also discussed. Example code projects are included that run from on-chip flash on the eZdsp™ F2812 development board (or alternately any F2812, F2811, or F2810 DSP board). Code examples that run from internal RAM are also provided for completeness. These code examples provide a starting point for code development, if desired.

Trademarks

Code Composer Studio and C28x are trademarks of Texas Instruments.

Contents

Figures

Tables

Chapter 1

Serial Peripheral Interface (SPI)

The serial peripheral interface (SPI) is a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is normally used for communications between the DSP controller and external peripherals or another controller. Typical applications include external I/O or peripheral expansion via devices such as shift registers, display drivers, and analog-to-digital converters (ADCs). Multidevice communications are supported by the master/slave operation of the SPI. On the C28x™, the port supports a 16-level, receive and transmit FIFO for reducing CPU servicing overhead.

This reference guide is applicable for the SPI found on both the TMS320x280x and TMS3201x281x families of processors. This includes all Flash-based, ROM-based and RAM-based devices within these families.

Topic **Page**

Note: 28x Enhanced Features

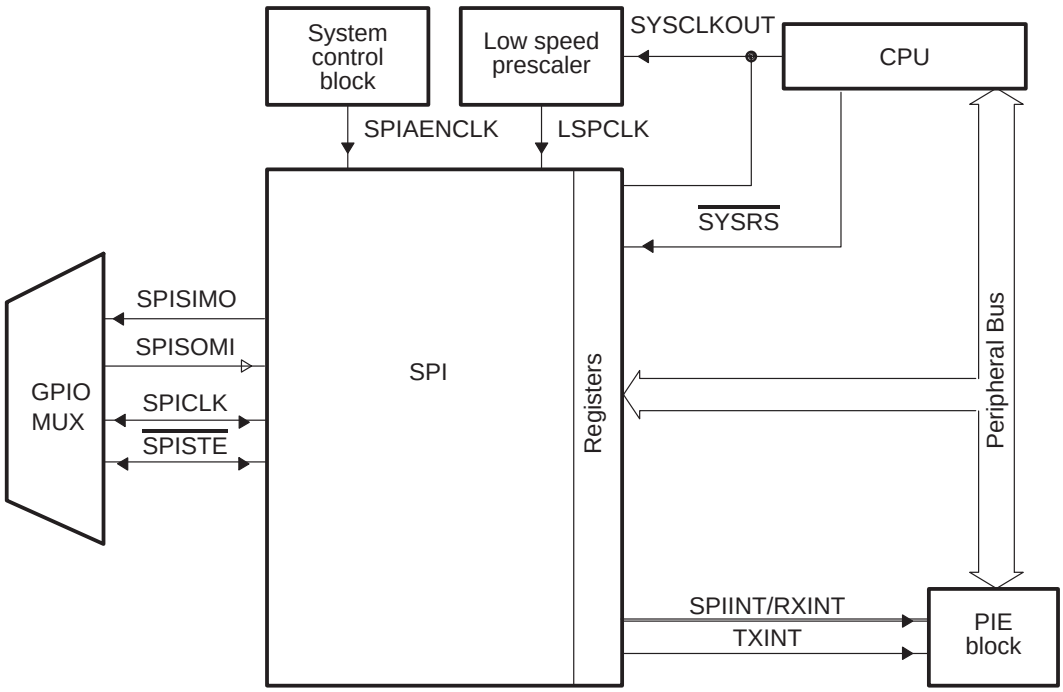
The 28x SPI features several enhancements compared to the 240xA SPI. See section 1.5 for a description of these features.

C28x is a trademark of Texas Instruments.

1.1 Enhanced SPI Module Overview

Figure 1−1 shows the SPI CPU interfaces.

Figure 1−1. SPI CPU Interface



The SPI module features include:

- Four external pins:
  - SPISOMI: SPI slave-output/master-input pin
  - SPISIMO: SPI slave-input/master-output pin
  - $\overline{\text{SPISTE}}$: SPI slave transmit-enable pin
  - SPICLK: SPI serial-clock pin

Note: All four pins can be used as GPIO, if the SPI module is not used.

- Two operational modes: master and slave
- Baud rate: 125 different programmable rates. The maximum baud rate that can be employed is limited by the maximum speed of the I/O buffers used on the SPI pins. See the device-specific data sheet for more details.
- Data word length: one to sixteen data bits

- Four clocking schemes (controlled by clock polarity and clock phase bits) include:
  - Falling edge without phase delay: SPICLK active-high. SPI transmits data on the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
  - Falling edge with phase delay: SPICLK active-high. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge without phase delay: SPICLK inactive-low. SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge with phase delay: SPICLK inactive-low. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
- Simultaneous receive and transmit operation (transmit function can be disabled in software)
- Transmitter and receiver operations are accomplished through either interrupt-driven or polled algorithms.
- 12 SPI module control registers: Located in control register frame beginning at address 7040h.

Note: All registers in this module are 16-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7–0), and the upper byte (15–8) is read as zeros. Writing to the upper byte has no effect.

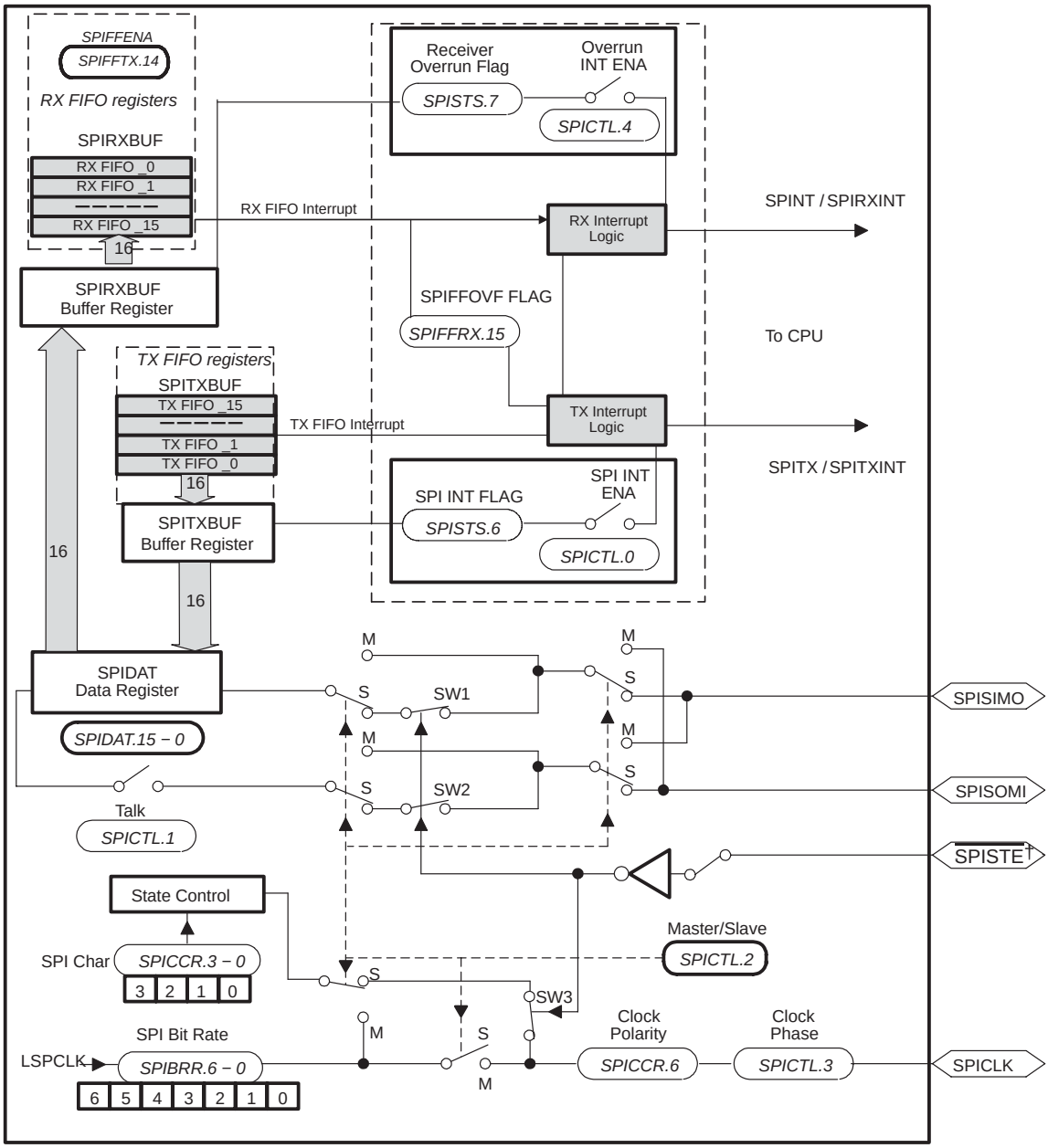
Enhanced feature:

- 16-level transmit/receive FIFO
- Delayed transmit control

1.1.1 SPI Block Diagram

Figure 1−2 is a block diagram of the SPI in slave mode, showing the basic control blocks available on the 28x SPI module.

Figure 1−2. Serial Peripheral Interface Module Block Diagram



† $\overline{\text{SPISTE}}$ of a slave device is driven low by the master.

1.1.2 SPI Module Signal Summary

Signal Name	Description
External Signals	
SPICLK	SPI clock
SPISIMO	SPI slave in, master out
SPISOMI	SPI slave out, master in
$\overline{\text{SPISTE}}$	SPI slave transmit enable
Control	
SPI CLock Rate	LSPCLK
Interrupt signals	
SPIRXINT	Transmit interrupt/ Receive Interrupt in non FIFO mode (referred to as SPI INT) Receive in interrupt in FIFO mode
SPITXINT	Transmit interrupt – FIFO

1.2 Overview of SPI Module Registers

The SPI port operation is configured and controlled by the registers listed in Table 1−1.

Table 1−1. SPI Registers

Name	Address Range	Size (x16)	Description
SPICCR	0x0000−7040	1	SPI Configuration Control Register
SPICTL	0x0000−7041	1	SPI Operation Control Register
SPIST	0x0000−7042	1	SPI Status Register
SPIBRR	0x0000−7044	1	SPI Baud Rate Register
SPIEMU	0x0000−7046	1	SPI Emulation Buffer Register
SPIRXBUF	0x0000−7047	1	SPI Serial Input Buffer Register
SPITXBUF	0x0000−7048	1	SPI Serial Output Buffer Register
SPIDAT	0x0000−7049	1	SPI Serial Data Register
SPIFFTX	0x0000−704A	1	SPI FIFO Transmit Register
SPIFFRX	0x0000−704B	1	SPI FIFO Receive Register
SPIFFCT	0x0000−704C	1	SPI FIFO Control Register
SPIPRI	0x0000−704F	1	SPI Priority Control Register

Note: The registers are mapped to Peripheral Frame 2. This space only allows 16-bit accesses. Using 32-bit accesses produces undefined results.

This SPI has 16-bit transmit and receive capability, with double-buffered transmit and double-buffered receive. All data registers are 16-bits wide.

The SPI is no longer limited to a maximum transmission rate of LSPCLK/8 in slave mode. The maximum transmission rate in *both* slave mode *and* master mode is now LSPCLK/4.

Writes of transmit data to the serial data register, SPIDAT (and the new transmit buffer, SPITXBUF), must be left-justified within a 16-bit register.

The control and data bits for general-purpose bit I/O multiplexing have been removed from this peripheral, along with the associated registers, SPIPC1 (704Dh) and SPIPC2 (704Eh). These bits are now in the General-Purpose I/O registers.

Twelve registers inside the SPI module control the SPI operations:

- SPICCR (SPI configuration control register). Contains control bits used for SPI configuration

- SPI module software reset
- SPICLK polarity selection
- Four SPI character-length control bits

❑ SPICTL (SPI operation control register). Contains control bits for data transmission

- Two SPI interrupt enable bits
- SPICLK phase selection
- Operational mode (master/slave)
- Data transmission enable

❑ SPISTS (SPI status register). Contains two receive buffer status bits and one transmit buffer status bit

- RECEIVER OVERRUN
- SPI INT FLAG
- TX BUF FULL FLAG

❑ SPIBRR (SPI baud rate register). Contains seven bits that determine the bit transfer rate

❑ SPIRXEMU (SPI receive emulation buffer register). Contains the received data. This register is used for emulation purposes only. The SPIRXBUF should be used for normal operation

❑ SPIRXBUF (SPI receive buffer — the serial receive buffer register). Contains the received data

❑ SPITXBUF (SPI transmit buffer — the serial transmit buffer register). Contains the next character to be transmitted

❑ SPIDAT (SPI data register). Contains data to be transmitted by the SPI, acting as the transmit/receive shift register. Data written to SPIDAT is shifted out on subsequent SPICLK cycles. For every bit shifted out of the SPI, a bit from the receive bit stream is shifted into the other end of the shift register

❑ SPIPRI (SPI priority register). Contains bits that specify interrupt priority and determine SPI operation on the XDS™ emulator during program suspensions

1.3 SPI Operation

This section describes the operation of the SPI. Included are explanations of the operation modes, interrupts, data format, clock sources, and initialization. Typical timing diagrams for data transfers are given.

1.3.1 Introduction to Operation

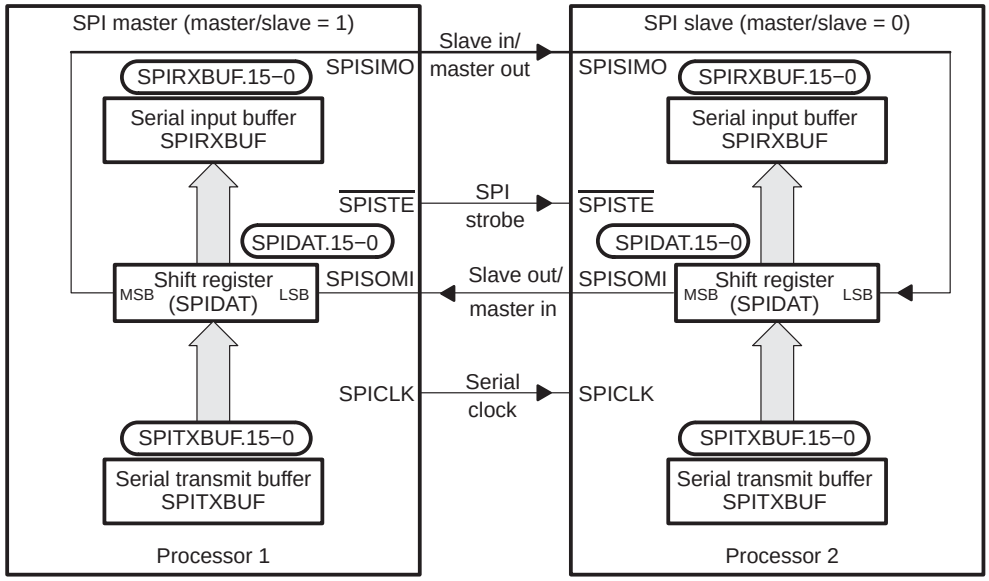
Figure 1−3 shows typical connections of the SPI for communications between two controllers: a master and a slave.

The master initiates data transfer by sending the SPICLK signal. For both the slave and the master, data is shifted out of the shift registers on one edge of the SPICLK and latched into the shift register on the opposite SPICLK clock edge. If the CLOCK PHASE bit (SPICTL.3) is high, data is transmitted and received a half-cycle before the SPICLK transition (see section 1.3.2, *SPI Module Slave and Master Operation Modes*, on page 1-9). As a result, both controllers send and receive data simultaneously. The application software determines whether the data is meaningful or dummy data. There are three possible methods for data transmission:

- ❏ Master sends data; slave sends dummy data.
- ❏ Master sends data; slave sends data.
- ❏ Master sends dummy data; slave sends data.

The master can initiate data transfer at any time because it controls the SPICLK signal. The software, however, determines how the master detects when the slave is ready to broadcast data.

Figure 1−3. SPI Master/Slave Connection



1.3.2 SPI Module Slave and Master Operation Modes

The SPI can operate in master or slave mode. The MASTER/SLAVE bit (SPICTL.2) selects the operating mode and the source of the SPICLK signal.

1.3.2.1 Master Mode

In the master mode (MASTER/SLAVE = 1), the SPI provides the serial clock on the SPICLK pin for the entire serial communications network. Data is output on the SPISIMO pin and latched from the SPISOMI pin.

The SPIBRR register determines both the transmit and receive bit transfer rate for the network. SPIBRR can select 126 different data transfer rates.

Data written to SPIDAT or SPITXBUF initiates data transmission on the SPISIMO pin, MSB (most significant bit) first. Simultaneously, received data is shifted through the SPISOMI pin into the LSB (least significant bit) of SPIDAT. When the selected number of bits has been transmitted, the received data is transferred to the SPIRXBUF (buffered receiver) for the CPU to read. Data is stored right-justified in SPIRXBUF.

When the specified number of data bits has been shifted through SPIDAT, the following events occur:

- ❏ SPIDAT contents are transferred to SPIRXBUF.
- ❏ SPI INT FLAG bit (SPISTS.6) is set to 1.

- If there is valid data in the transmit buffer SPITXBUF, as indicated by the TXBUF FULL bit in SPISTS, this data is transferred to SPIDAT and is transmitted; otherwise, SPICLK stops after all bits have been shifted out of SPIDAT.

- If the SPI INT ENA bit (SPICTL.0) is set to 1, an interrupt is asserted.

In a typical application, the $\overline{\text{SPISTE}}$ pin serves as a chip-enable pin for a slave SPI device. This pin is driven low by the master before transmitting data to the slave and is taken high after the transmission is complete.

1.3.2.2 Slave Mode

In the slave mode (MASTER/SLAVE = 0), data shifts out on the SPISOMI pin and in on the SPISIMO pin. The SPICLK pin is used as the input for the serial shift clock, which is supplied from the external network master. The transfer rate is defined by this clock. The SPICLK input frequency should be no greater than the LSPCLK frequency divided by 4.

Data written to SPIDAT or SPITXBUF is transmitted to the network when appropriate edges of the SPICLK signal are received from the network master. Data written to the SPITXBUF register will be transferred to the SPIDAT register when all bits of the character to be transmitted have been shifted out of SPIDAT. If no character is currently being transmitted when SPITXBUF is written to, the data will be transferred immediately to SPIDAT. To receive data, the SPI waits for the network master to send the SPICLK signal and then shifts the data on the SPISIMO pin into SPIDAT. If data is to be transmitted by the slave simultaneously, and SPITXBUF has not been previously loaded, the data must be written to SPITXBUF or SPIDAT before the beginning of the SPICLK signal.

When the TALK bit (SPICTL.1) is cleared, data transmission is disabled, and the output line (SPISOMI) is put into the high-impedance state. If this occurs while a transmission is active, the current character is completely transmitted even though SPISOMI is forced into the high-impedance state. This ensures that the SPI is still able to receive incoming data correctly. This TALK bit allows many slave devices to be tied together on the network, but only one slave at a time is allowed to drive the SPISOMI line.

The $\overline{\text{SPISTE}}$ pin operates as the slave-select pin. An active-low signal on the $\overline{\text{SPISTE}}$ pin allows the slave SPI to transfer data to the serial data line; an inactive-high signal causes the slave SPI serial shift register to stop and its serial output pin to be put into the high-impedance state. This allows many slave devices to be tied together on the network, although only one slave device is selected at a time.

1.4 SPI Interrupts

This section includes information on the control bits that initialize interrupts, data format, clocking, initialization, and data transfer.

1.4.1 SPI Interrupt Control Bits

Five control bits are used to initialize the SPI interrupts:

- ❏ SPI INT ENA bit (SPICTL.0)
- ❏ SPI INT FLAG bit (SPISTS.6)
- ❏ OVERRUN INT ENA bit (SPICTL.4)
- ❏ RECEIVER OVERRUN FLAG bit (SPISTS.7)
- ❏ SPI PRIORITY bit (SPIPRI.6)

1.4.1.1 SPI INT ENA Bit (SPICTL.0)

When the SPI interrupt-enable bit is set and an interrupt condition occurs, the corresponding interrupt is asserted.

0 Disable SPI interrupts
1 Enable SPI interrupts

1.4.1.2 SPI INT FLAG Bit (SPISTS.6)

This status flag indicates that a character has been placed in the SPI receiver buffer and is ready to be read.

When a complete character has been shifted into or out of SPIDAT, the SPI INT FLAG bit (SPISTS.6) is set, and an interrupt is generated if enabled by the SPI INT ENA bit (SPICTL.0). The interrupt flag remains set until it is cleared by one of the following events:

- ❏ The interrupt is acknowledged (this is different from the C240).
- ❏ The CPU reads the SPIRXBUF (reading the SPIRXEMU does not clear the SPI INT FLAG bit).
- ❏ The device enters IDLE2 or HALT mode with an IDLE instruction.
- ❏ Software clears the SPI SW RESET bit (SPICCR.7).
- ❏ A system reset occurs.

When the SPI INT FLAG bit is set, a character has been placed into the SPIRX-BUF and is ready to be read. If the CPU does not read the character by the time

the next complete character has been received, the new character is written into SPIRXBUF, and the RECEIVER OVERRUN Flag bit (SPISTS.7) is set.

1.4.1.3 OVERRUN INT ENA Bit (SPICTL.4)

Setting the overrun interrupt enable bit allows the assertion of an interrupt whenever the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by SPISTS.7 and by the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector.

0 Disable RECEIVER OVERRUN Flag bit interrupts

1 Enable RECEIVER OVERRUN Flag bit interrupts

1.4.1.4 RECEIVER OVERRUN FLAG Bit (SPISTS.7)

The RECEIVER OVERRUN Flag bit is set whenever a new character is received and loaded into the SPIRXBUF before the previously received character has been read from the SPIRXBUF. The RECEIVER OVERRUN Flag bit must be cleared by software.

1.4.2 Data Format

Four bits (SPICCR.3–0) specify the number of bits (1 to 16) in the data character. This information directs the state control logic to count the number of bits received or transmitted to determine when a complete character has been processed. The following statements apply to characters with fewer than 16 bits:

- Data must be left-justified when written to SPIDAT and SPITXBUF.
- Data read back from SPIRXBUF is right-justified.
- SPIRXBUF contains the most recently received character, right-justified, plus any bits that remain from previous transmission(s) that have been shifted to the left (shown in Example 1−1).

Example 1−1. Transmission of Bit From SPIRXBUF

Conditions:

1) Transmission character length = 1 bit (specified in bits SPICCR.3−0)
2) The current value of SPIDAT = 737Bh

SPIDAT (before transmission)

0	1	1	1	0	0	1	1	0	1	1	1	1	0	1	1

SPIDAT (after transmission)

(TXed) 0 ←

1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x

← (RXed)

SPIRXBUF (after transmission)

1	1	1	0	0	1	1	0	1	1	1	1	0	1	1	x

Note: x = 1 if SPISOMI data is high; x = 0 if SPISOMI data is low; master mode is assumed.

1.4.3 Baud Rate and Clocking Schemes

The SPI module supports 125 different baud rates and four different clock schemes. Depending on whether the SPI clock is in slave or master mode, the SPICLK pin can receive an external SPI clock signal or provide the SPI clock signal, respectively.

- In the slave mode, the SPI clock is received on the SPICLK pin from the external source, and can be no greater than the LSPCLK frequency divided by 4.
- In the master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin, and can be no greater than the LSPCLK frequency divided by 4.

Baud Rate Determination

Equation 1−1 shows how to determine the SPI baud rates.

Equation 1−1. SPI Baud-Rate Calculations

- For SPIBRR = 3 to 127:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$$

- For SPIBRR = 0, 1, or 2:

$$\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4}$$

where:

LSPCLK = Low-speed peripheral clock frequency of the device

SPIBRR = Contents of the SPIBRR in the master SPI device

To determine what value to load into SPIBRR, you must know the device system clock (LSPCLK) frequency (which is device-specific) and the baud rate at which you will be operating.

Example 1−2 shows how to determine the maximum baud rate at which a 240xA can communicate. Assume that LSPCLK = 40 MHz.

Example 1−2. Maximum Baud-Rate Calculation

$$\text{Maximum SPI Baud Rate} = \frac{\text{LSPCLK}}{4} = \frac{40 \times 10^6}{4} = 10 \times 10^6 \text{ bps}$$

SPI Clocking Schemes

The CLOCK POLARITY bit (SPICCR.6) and the CLOCK PHASE bit (SPICTL.3) control four different clocking schemes on the SPICLK pin. The CLOCK POLARITY bit selects the active edge, either rising or falling, of the clock. The CLOCK PHASE bit selects a half-cycle delay of the clock. The four different clocking schemes are as follows:

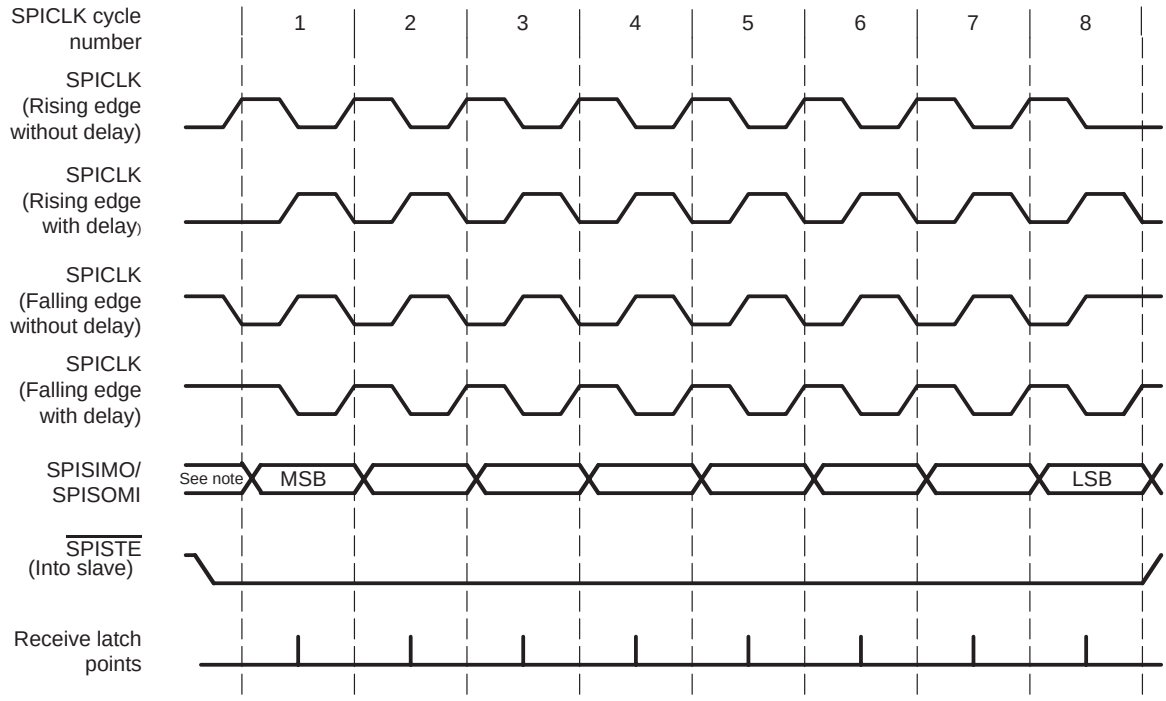
- Falling Edge Without Delay. The SPI transmits data on the falling edge of the SPICLK and receives data on the rising edge of the SPICLK.
- Falling Edge With Delay. The SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge Without Delay. The SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
- Rising Edge With Delay. The SPI transmits data one half-cycle ahead of the rising edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.

The selection procedure for the SPI clocking scheme is shown in Table 1−2. Examples of these four clocking schemes relative to transmitted and received data are shown in Figure 1−4.

Table 1−2. SPI Clocking Scheme Selection Guide

SPICLK Scheme	CLOCK POLARITY (SPICCR.6)	CLOCK PHASE (SPICTL.3)
Rising edge without delay	0	0
Rising edge with delay	0	1
Falling edge without delay	1	0
Falling edge with delay	1	1

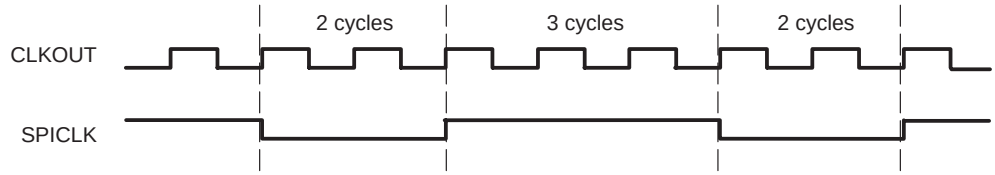
Figure 1−4. SPICLK Signal Options



Note: Previous data bit

For the SPI, SPICLK symmetry is retained only when the result of (SPIBRR+1) is an even value. When (SPIBRR + 1) is an odd value and SPIBRR is greater than 3, SPICLK becomes asymmetrical. The low pulse of SPICLK is one CLKOUT longer than the high pulse when the CLOCK POLARITY bit is clear (0). When the CLOCK POLARITY bit is set to 1, the high pulse of the SPICLK is one CLKOUT longer than the low pulse, as shown in Figure 1−5.

Figure 1−5. SPI: SPICLK-CLKOUT Characteristic When (BRR + 1) is Odd, BRR > 3, and CLOCK POLARITY = 1



1.4.4 Initialization Upon Reset

A system reset forces the SPI peripheral module into the following default configuration:

- ❏ Unit is configured as a slave module (MASTER/SLAVE = 0)
- ❏ Transmit capability is disabled (TALK = 0)
- ❏ Data is latched at the input on the falling edge of the SPICLK signal
- ❏ Character length is assumed to be one bit
- ❏ SPI interrupts are disabled
- ❏ Data in SPIDAT is reset to 0000h
- ❏ SPI module pin functions are selected as general-purpose inputs (this is done in I/O MUX control register B [MCRB])

To change this SPI configuration:

1) Clear the SPI SW RESET bit (SPICCR.7) to 0 to force the SPI to the reset state.

2) Initialize the SPI configuration, format, baud rate, and pin functions as desired.

3) Set the SPI SW RESET bit to 1 to release the SPI from the reset state.

4) Write to SPIDAT or SPITXBUF (this initiates the communication process in the master).

5) Read SPIRXBUF after the data transmission has completed (SPISTS.6 = 1) to determine what data was received.

To prevent unwanted and unforeseen events from occurring during or as a result of initialization changes, clear the SPI SW RESET bit (SPICCR.7) before making initialization changes, and then set this bit after initialization is complete.

Note:

Do not change SPI configuration when communication is in progress.

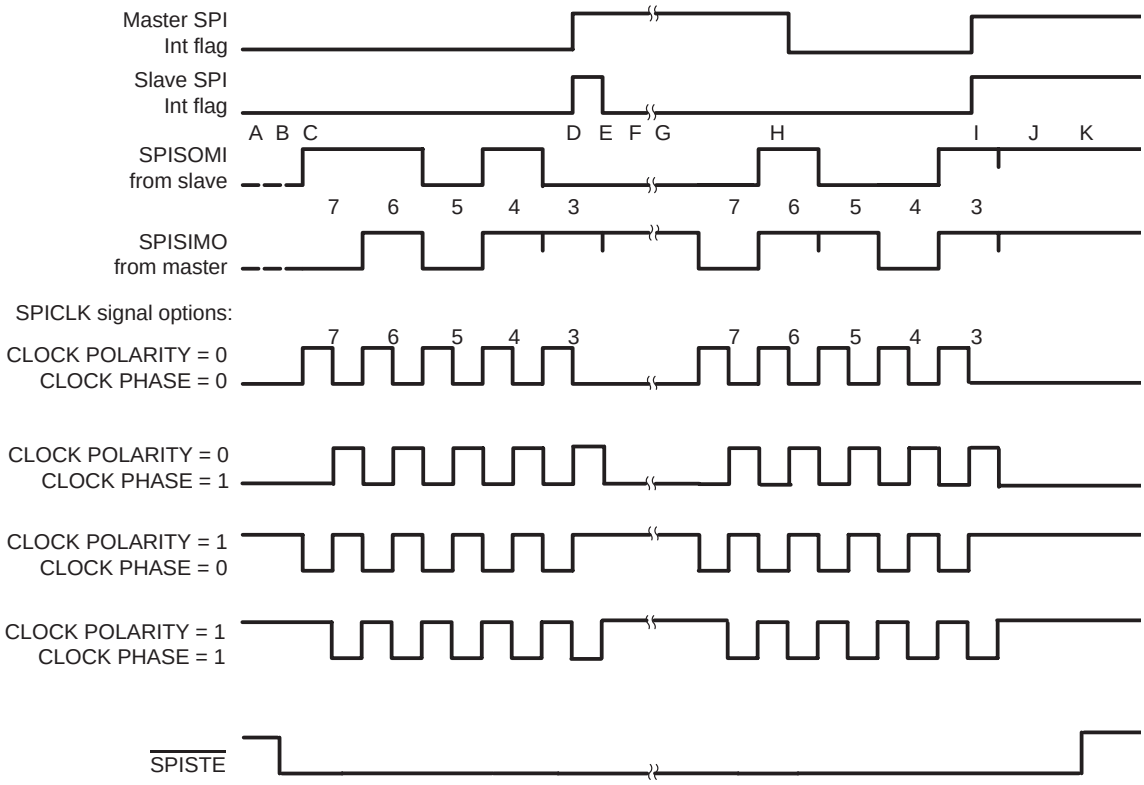
1.4.5 Data Transfer Example

The timing diagram shown in Figure 1−6 illustrates an SPI data transfer between two devices using a character length of five bits with the SPICLK being symmetrical.

The timing diagram with SPICLK unsymmetrical (Figure 1−5) shares similar characterizations with Figure 1−6 except that the data transfer is one CLKOUT cycle longer per bit during the low pulse (CLOCK POLARITY = 0) or during the high pulse (CLOCK POLARITY = 1) of the SPICLK.

Figure 1−6 is applicable for 8-bit SPI only and is not for 24x devices that are capable of working with 16-bit data. The figure is shown for illustrative purposes only.

Figure 1−6. Five Bits per Character



A. Slave writes 0D0h to SPIDAT and waits for the master to shift out the data.
B. Master sets the slave SPISTE signal low (active).
C. Master writes 058h to SPIDAT, which starts the transmission procedure.
D. First byte is finished and sets the interrupt flags.
E. Slave reads 0Bh from its SPIRXBUF (right-justified).
F Slave writes 04Ch to SPIDAT and waits for the master to shift out the data.
G. Master writes 06Ch to SPIDAT, which starts the transmission procedure.
H. Master reads 01Ah from the SPIRXBUF (right−justified).
I. Second byte is finished and sets the interrupt flags.
J. Master reads 89h and the slave reads 8Dh from their respective SPIRXBUF. After the user's software masks off the unused bits, the master receives 09h and the slave receives 0Dh.
K. Master clears the slave SPISTE signal high (inactive).

1.5 SPI FIFO Description

The following steps explain the the FIFO features and help with programming the SPI FIFOs:

1) Reset. At reset the SPI powers up in standard SPI mode, the FIFO function is disabled. The FIFO registers SPIFFTX, SPIFFRX and SPIFFCT remain inactive.

2) Standard SPI. The standard 240x SPI mode will work with SPIINT/SPIRXINT as the interrupt source.

3) Mode change. FIFO mode is enabled by setting the SPIFFEN bit to 1 in the SPIFFTX register. SPIRST can reset the FIFO mode at any stage of its operation.

4) Active registers. All the SPI registers and SPI FIFO registers SPIFFTX, SPIFFRX, and SPIFFCT will be active.

5) Interrupts. FIFO mode has two interrupts one for transmit FIFO, SPITXINT and one for receive FIFO, SPIINT/SPIRXINT. SPIINT/SPIRXINT is the common interrupt for SPI FIFO receive, receive error and receive FIFO overflow conditions. The single SPIINT for both transmit and receive sections of the standard SPI will be disabled and this interrupt will service as SPI receive FIFO interrupt.

6) Buffers. Transmit and receive buffers are supplemented with two 16x16 FIFOs. The one-word transmit buffer (TXBUF) of the standard SPI functions as a transition buffer between the transmit FIFO and shift register. The one-word transmit buffer will be loaded from transmit FIFO only after the last bit of the shift register is shifted out.

7) Delayed transfer. The rate at which transmit words in the FIFO are transferred to transmit shift register is programmable. The SPIFFCT register bits (7−0) FFTXDLY7−FFTXDLY0 define the delay between the word transfer. The delay is defined in number SPI serial clock cycles. The 8-bit register could define a minimum delay of 0 serial clock cycles and a maximum of 256 serial clock cycles. With zero delay, the SPI module can transmit data in continuous mode with the FIFO words shifting out back to back. With the 256 clock delay, the SPI module can transmit data in a maximum delayed mode with the FIFO words shifting out with a delay of 256 SPI clocks between each words. The programmable delay facilitates glueless interface to various slow SPI peripherals, such as EEPROMs, ADC, DAC etc.

8) FIFO status bits. Both transmit and receive FIFOs have status bits TXFFST or RXFFST (bits 12− 0) that define the number of words available

in the FIFOs at any time. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO will reset the FIFO pointers to zero when these bits are set to 1. The FIFOs will resume operation from start once these bits are cleared to zero.

9) Programmable interrupt levels. Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12−8) match (less than or equal to) the interrupt trigger level bits TXFFIL (bits 4−0). This provides a programmable interrupt trigger for transmit and receive sections of the SPI. The default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO respectively.

1.5.1 SPI Interrupts

Figure 1−7. SPI FIFO Interrupt Flags and Enable Logic Generation

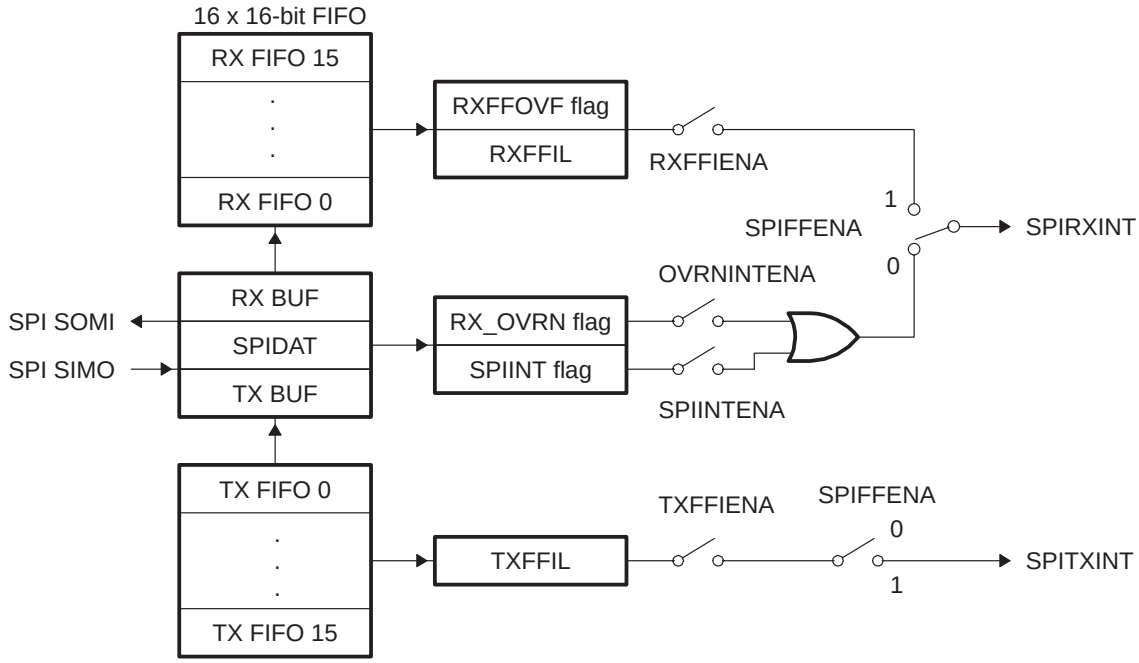


Table 1−3. SPI Interrupt Flag Modes

FIFO Options	SPI Interrupt Source	Interrupt Flags	Interrupt Enables	FIFO Enable SPIFFENA	Interrupt line
SPI without FIFO					
	Receive overrun	RXOVRN	OVRNINTENA	0	SPIRXINT†
	Data receive	SPIINT	SPIINTENA	0	SPIRXINT†
	Transmit empty	SPIINT	SPIINTENA	0	SPIRXINT†
SPI FIFO mode					
	FIFO receive	RXFFIL	RXFFIENA	1	SPIRXINT†
	Transmit empty	TXFFIL	TXFFIENA	1	SPITXINT†

† In nonFIFO mode, SPIRXINT is the same as the SPIINT interrupt in 240x devices.

Chapter 2

SPI Registers and Waveforms

This section contains the registers, bit descriptions, and waveforms.

Topic **Page**

2.1 SPI Control Registers

The SPI is controlled and accessed through registers in the control register file.

2.1.1 SPI Configuration Control Register (SPICCR)

SPICCR controls the setup of the SPI for operation.

Figure 2−1. SPI Configuration Control Register (SPICCR) — Address 7040h

7	6	5	4	3	2	1	0
SPI SW Reset	CLOCK POLARITY	Reserved	SPILBK	SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0
R/W-0	R/W−0	R−0	R−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read access; W = Write access; −x = value after reset

Table 2−1. SPI Configuration Control Register (SPICCR) Bit Descriptions

Bit(s)	Name	Description
7	SPI SW RESET	SPI software reset. When changing configuration, you should clear this bit before the changes and set this bit before resuming operation. 1 SPI is ready to transmit or receive the next character. When the SPI SW RESET bit is a 0, a character written to the transmitter will not be shifted out when this bit is set. A new character must be written to the serial data register. 0 Initializes the SPI operating flags to the reset condition. Specifically, the RECEIVER OVERRUN Flag bit (SPISTS.7), the SPI INT FLAG bit (SPISTS.6), and the TXBUF FULL Flag bit (SPISTS.5) are cleared. The SPI configuration remains unchanged. If the module is operating as a master, the SPICLK signal output returns to its inactive level.
6	CLOCK POLARITY	Shift Clock Polarity. This bit controls the polarity of the SPICLK signal. CLOCK POLARITY and CLOCK PHASE (SPICTL.3) control four clocking schemes on the SPICLK pin. See Section 1.4.3, *SPI Clocking Schemes*, on page 1-13. 1 Data is output on falling edge and input on rising edge. When no SPI data is sent, SPICLK is at high level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: ❑ CLOCK PHASE = 0: Data is output on the falling edge of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal. ❑ CLOCK PHASE = 1: Data is output one half-cycle before the first falling edge of the SPICLK signal and on subsequent rising edges of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal.

Table 2−1. SPI Configuration Control Register (SPICCR) Bit Descriptions (Continued)

Bit(s)	Name	Description
6	CLOCK POLARITY (continued)	0 Data is output on rising edge and input on falling edge. When no SPI data is sent, SPICLK is at low level. The data input and output edges depend on the value of the CLOCK PHASE bit (SPICTL.3) as follows: ❑ CLOCK PHASE = 0: Data is output on the rising edge of the SPICLK signal; input data is latched on the falling edge of the SPICLK signal. ❑ CLOCK PHASE = 1: Data is output one half-cycle before the first rising edge of the SPICLK signal and on subsequent falling edges of the SPICLK signal; input data is latched on the rising edge of the SPICLK signal.
5	Reserved	Reads return zero; writes have no effect.
4	SPILBK	SPI loopback. Loop back mode allows module validation during device testing. This mode is valid only in master mode of the SPI. 1 SPI loop back mode enabled, SIMO/SOMI lines are connected internally. Used for module self tests. 0 SPI loop back mode disabled – default value after reset
3−0	SPI CHAR3 − SPI CHAR0	Character Length Control Bits 3−0. These four bits determine the number of bits to be shifted in or out as a single character during one shift sequence. Table 2−2 lists the character length selected by the bit values.

Table 2−2. Character Length Control Bit Values

SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0	Character Length
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
0	0	1	1	4
0	1	0	0	5
0	1	0	1	6
0	1	1	0	7
0	1	1	1	8
1	0	0	0	9
1	0	0	1	10
1	0	1	0	11
1	0	1	1	12
1	1	0	0	13
1	1	0	1	14

Table 2−2. Character Length Control Bit Values (Continued)

SPI CHAR3	SPI CHAR2	SPI CHAR1	SPI CHAR0	Character Length
1	1	1	0	15
1	1	1	1	16

2.1.2 SPI Operation Control Register (SPICTL)

SPICTL controls data transmission, the SPI's ability to generate interrupts, the SPICLK phase, and the operational mode (slave or master).

Figure 2−2. SPI Operation Control Register (SPICTL) — Address 7041h

7 5	4	3	2	1	0
Reserved	OVERRUN INT ENA	CLOCK PHASE	MASTER/ SLAVE	TALK	SPI INT ENA
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access; W = Write access; −x = value after reset

Table 2−3. SPI Operation Control Register (SPICTL) Bit Descriptions

Bit(s)	Name	Description
7−5	Reserved	Reads return zero; writes have no effect.
4	Overrun INT ENA	Overrun Interrupt Enable. Setting this bit causes an interrupt to be generated when the RECEIVER OVERRUN Flag bit (SPISTS.7) is set by hardware. Interrupts generated by the RECEIVER OVERRUN Flag bit and the SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. 1 Enable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts 0 Disable RECEIVER OVERRUN Flag bit (SPISTS.7) interrupts
3	CLOCK PHASE	SPI Clock Phase Select. This bit controls the phase of the SPICLK signal. CLOCK PHASE and CLOCK POLARITY (SPICCR.6) make four different clocking schemes possible (see Figure 1−4). When operating with CLOCK PHASE high, the SPI (master or slave) makes the first bit of data available after SPIDAT is written and before the first edge of the SPICLK signal, regardless of which SPI mode is being used. 1 SPICLK signal delayed by one half-cycle; polarity determined by the CLOCK POLARITY bit 0 Normal SPI clocking scheme, depending on the CLOCK POLARITY bit (SPICCR.6)

Table 2−3. SPI Operation Control Register (SPICTL) Bit Descriptions (Continued)

Bit(s)	Name	Description
2	MASTER / SLAVE	SPI Network Mode Control. This bit determines whether the SPI is a network master or slave. During reset initialization, the SPI is automatically configured as a network slave. 1 SPI configured as a master. 0 SPI configured as a slave.
1	TALK	Master/Slave Transmit Enable. The TALK bit can disable data transmission (master or slave) by placing the serial data output in the high-impedance state. If this bit is disabled during a transmission, the transmit shift register continues to operate until the previous character is shifted out. When the TALK bit is disabled, the SPI is still able to receive characters and update the status flags. TALK is cleared (disabled) by a system reset. 1 Enables transmission For the 4-pin option, ensure to enable the receiver's $\overline{\text{SPISTE}}$ input pin. 0 Disables transmission: ❑ Slave mode operation: If not previously configured as a general-purpose I/O pin, the SPISOMI pin will be put in the high-impedance state. ❑ Master mode operation: If not previously configured as a general-purpose I/O pin, the SPISIMO pin will be put in the high-impedance state.
0	SPI INT ENA	SPI Interrupt Enable. This bit controls the SPI's ability to generate a transmit/receive interrupt. The SPI INT FLAG bit (SPISTS.6) is unaffected by this bit. 1 Enables interrupt 0 Disables interrupt

2.1.3 SPI Status Register (SPIST)

Figure 2−3. SPI Status Register (SPIST) — Address 7042h

7	6	5	4 … 0
RECEIVER OVERRUN FLAG†‡	SPI INT FLAG†‡	TX BUF FULL FLAG‡	Reserved
R/C-0	R/C−0	R/C−0	R−0

Legend: R = Read access; C = Clear; −x = value after reset
† The RECEIVER OVERRUN FLAG bit and the SPI INT FLAG bit share the same interrupt vector.
‡ Writing a 0 to bits 5, 6, and 7 has no effect.

Table 2−4. SPI Status Register (SPIST) Bit Descriptions

Bit(s)	Name	Description
7	RECEIVER OVERRUN FLAG	SPI Receiver Overrun Flag. This bit is a read/clear-only flag. The SPI hardware sets this bit when a receive or transmit operation completes before the previous character has been read from the buffer. The bit indicates that the last received character has been overwritten and therefore lost (when the SPIRXBUF was overwritten by the SPI module before the previous character was read by the user application). The SPI requests one interrupt sequence each time this bit is set if the OVERRUN INT ENA bit (SPICTL.4) is set high. The bit is cleared in one of three ways: ❑ Writing a 1 to this bit ❑ Writing a 0 to SPI SW RESET (SPICCR.7) ❑ Resetting the system If the OVERRUN INT ENA bit (SPICTL.4) is set, the SPI requests only one interrupt upon the first occurrence of setting the RECEIVER OVERRUN Flag bit. Subsequent overruns will not request additional interrupts if this flag bit is already set. This means that in order to allow *new* overrun interrupt requests the user must clear this flag bit by writing a 1 to SPISTS.7 each time an overrun condition occurs. In other words, if the RECEIVER OVERRUN Flag bit is left set (not cleared) by the interrupt service routine, another overrun interrupt will not be immediately re-entered when the interrupt service routine is exited. However, the RECEIVER OVERRUN Flag bit should be cleared during the interrupt service routine because the RECEIVER OVERRUN Flag bit and SPI INT FLAG bit (SPISTS.6) share the same interrupt vector. This will alleviate any possible doubt as to the source of the interrupt when the next byte is received.
6	SPI INT FLAG	SPI Interrupt Flag. SPI INT FLAG is a read-only flag. The SPI hardware sets this bit to indicate that it has completed sending or receiving the last bit and is ready to be serviced. The received character is placed in the receiver buffer at the same time this bit is set. This flag causes an interrupt to be requested if the SPI INT ENA bit (SPICTL.0) is set. This bit is cleared in one of three ways: ❑ Reading SPIRXBUF ❑ Writing a 0 to SPI SW RESET (SPICCR.7) ❑ Resetting the system
5	TX BUF FULL FLAG	SPI Transmit Buffer Full Flag. This read-only bit gets set to 1 when a character is written to the SPI Transmit buffer SPITXBUF. It is cleared when the character is automatically loaded into SPIDAT when the shifting out of a previous character is complete. It is cleared at reset.
4−0	Reserved	Reads return zero; writes have no effect.

2.1.4 SPI Baud Rate Register (SPIBRR)

SPIBRR contains the bits used for baud-rate selection.

Figure 2−4. SPI Baud Rate Register (SPIBRR) — Address 7044h

7	6	5	4	3	2	1	0
Reserved	SPI BIT RATE 6	SPI BIT RATE 5	SPI BIT RATE 4	SPI BIT RATE 3	SPI BIT RATE 2	SPI BIT RATE 1	SPI BIT RATE 0
R-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−5. SPI Baud Rate Register (SPIBRR) Bit Descriptions

Bit(s)	Name	Description
7	Reserved	Reads return zero; writes have no effect.
6−0	SPI BIT RATE 6− SPI BIT RATE 0	SPI Bit Rate (Baud) Control. These bits determine the bit transfer rate if the SPI is the network master. There are 125 data-transfer rates (each a function of the CPU clock, LSPCLK) that can be selected. One data bit is shifted per SPICLK cycle. (SPICLK is the baud rate clock output on the SPICLK pin.) If the SPI is a network slave, the module receives a clock on the SPICLK pin from the network master; therefore, these bits have no effect on the SPICLK signal. The frequency of the input clock from the master should not exceed the slave SPI's SPICLK signal divided by 4. In master mode, the SPI clock is generated by the SPI and is output on the SPICLK pin. The SPI baud rates are determined by the following formula: ❑ For SPIBRR = 3 to 127: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$ ❑ For SPIBRR = 0, 1, or 2: $\text{SPI Baud Rate} = \frac{\text{LSPCLK}}{4}$ where: LSPCLK = Function of CPU clock frequency X low-speed peripheral clock of the device SPIBRR = Contents of the SPIBRR in the master SPI device

2.1.5 SPI Emulation Buffer Register (SPIRXEMU)

SPIRXEMU contains the received data. Reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). This is not a real register but a dummy address from which the contents of SPIRXBUF can be read by the emulator without clearing the SPI INT FLAG.

Figure 2−5. SPI Emulation Buffer Register (SPIRXEMU) — Address 7046h

15	14	13	12	11	10	9	8
ERXB15	ERXB14	ERXB13	ERXB12	ERXB11	ERXB10	ERXB9	ERXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
ERXB7	ERXB6	ERXB5	ERXB4	ERXB3	ERXB2	ERXB1	ERXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read access, -0 = value after reset

Table 2−6. SPI Emulation Buffer Register (SPIRXEMU) Bit Descriptions

Bit(s)	Name	Description
15−0	ERXB15−ERXB0	Emulation Buffer Received Data. SPIRXEMU functions almost identically to SPIRXBUF, except that reading SPIRXEMU does not clear the SPI INT FLAG bit (SPISTS.6). Once the SPIDAT has received the complete character, the character is transferred to SPIRXEMU and SPIRXBUF, where it can be read. At the same time, SPI INT FLAG is set. This mirror register was created to support emulation. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6). In the normal operation of the emulator, the control registers are read to continually update the contents of these registers on the display screen. SPIRXEMU was created so that the emulator can read this register and properly update the contents on the display screen. Reading SPIRXEMU does not clear the SPI INT FLAG bit, but reading SPIRXBUF clears this flag. In other words, SPIRXEMU enables the emulator to emulate the true operation of the SPI more accurately. It is recommended that you view SPIRXEMU in the normal emulator run mode.

2.1.6 SPI Serial Receive Buffer Register (SPIRXBUF)

SPIRXBUF contains the received data. Reading SPIRXBUF clears the SPI INT FLAG bit (SPISTS.6).

Figure 2−6. SPI Serial Receive Buffer Register (SPIRXBUF) — Address 7047h

15	14	13	12	11	10	9	8
RXB15	RXB14	RXB13	RXB12	RXB11	RXB10	RXB9	RXB8
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RXB7	RXB6	RXB5	RXB4	RXB3	RXB2	RXB1	RXB0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Legend: R = Read access, -0 = value after reset

Table 2−7. SPI Serial Receive Buffer Register (SPIRXBUF Bit Descriptions

Bit(s)	Name	Description
15−0	RXV15 − RXB0	Received Data. Once SPIDAT has received the complete character, the character is transferred to SPIRXBUF, where it can be read. At the same time, the SPI INT FLAG bit (SPISTS.6) is set. Since data is shifted into the SPI's most significant bit first, it is stored right-justified in this register.

2.1.7 SPI Serial Transmit Buffer Register (SPITXBUF)

SPITXBUF stores the next character to be transmitted. Writing to this register sets the TX BUF FULL Flag bit (SPISTS.5). When transmission of the current character is complete, the contents of this register are automatically loaded in SPIDAT and the TX BUF FULL Flag is cleared. If no transmission is currently active, data written to this register falls through into the SPIDAT register and the TX BUF FULL Flag is not set.

In master mode, if no transmission is currently active, writing to this register initiates a transmission in the same manner that writing to SPIDAT does.

Figure 2−7. SPI Serial Transmit Buffer Register (SPITXBUF) — Address 7048h

15	14	13	12	11	10	9	8
TXB15	TXB14	TXB13	TXB12	TXB11	TXB10	TXB9	TXB8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
TXB7	TXB6	TXB5	TXB4	TXB3	TXB2	TXB1	TXB0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−8. SPI Serial Transmit Buffer Register (SPITXBUF) Bit Descriptions

Bit(s)	Name	Description
15−0	TXV15−TXV0	Transmit Data Buffer. This is where the next character to be transmitted is stored. When the transmission of the current character has completed, if the TX BUF FULL Flag bit is set, the contents of this register is automatically transferred to SPIDAT, and the TX BUF FULL Flag is cleared. **Note:** Writes to SPITXBUF must be left-justified.

2.1.8 SPI Serial Data Register (SPIDAT)

SPIDAT is the transmit/receive shift register. Data written to SPIDAT is shifted out (MSB) on subsequent SPICLK cycles. For every bit (MSB) shifted out of the SPI, a bit is shifted into the LSB end of the shift register.

Figure 2−8. SPI Serial Data Register (SPIDAT) — Address 7049h

15	14	13	12	11	10	9	8
SDAT15	SDAT14	SDAT13	SDAT12	SDAT11	SDAT10	SDAT9	SDAT8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
SDAT7	SDAT6	SDAT5	SDAT4	SDAT3	SDAT2	SDAT1	SDAT0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−9. SPI Serial Data Register (SPIDAT)Bit Descriptions

Bit(s)	Name	Description
15−0	SDAT15−SDAT0	Serial data. Writing to the SPIDAT performs two functions: ❑ It provides data to be output on the serial output pin if the TALK bit (SPICTL.1) is set. ❑ When the SPI is operating as a master, a data transfer is initiated. When initiating a transfer, see the CLOCK POLARITY bit (SPICCR.6) described in section 2.1.1, *SPI Configuration Control Register* on page 2-2, and the CLOCK PHASE bit (SPICTL.3) described in section 2.1.2, *SPI Operation Control Register* on page 2-4, for the requirements. In master mode, writing dummy data to SPIDAT initiates a receiver sequence. Since the data is not hardware-justified for characters shorter than sixteen bits, transmit data must be written in left-justified form, and received data read in right-justified form.

2.1.9 SPI FIFO Transmit, Receive, and Control Registers

Figure 2−9. SPI FIFO Transmit (SPIFFTX) Register − Address 704Ah

15	14	13	12	11	10	9	8
SPIRST	SPIFFENA	TXFIFO	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R/W-1	R/W−0	R/W−1	R−0	R−0	R−0	R−0	R−0

7	6	5	4	3	2	1	0
TXFFINT Flag	TXFFINT CLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R/W-0	W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−10. SPI FIFO Transmit (SPIFFTX) Register Bit Descriptions

Bit(s)	Name	Description
15	SPIRST	SPI reset
		0 Write 0 to reset the SPI transmit and receive channels. The SPI FIFO register configuration bits will be left as is.
		1 SPI FIFO can resume transmit or receive. No effect to the SPI registers bits.
14	SPIFFENA	SPI FIFO enhancements enable
		0 SPI FIFO enhancements are disabled
		1 SPI FIFO enhancements are enabled
13	TXFIFO Reset	Transmit FIFO reset
		0 Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1 Re-enable Transmit FIFO operation
8−12	TXFFST4−0	Transmit FIFO status
		00000 Transmit FIFO is empty.
		00001 Transmit FIFO has 1 word.
		00010 Transmit FIFO has 2 words.
		00011 Transmit FIFO has 3 words.
		0xxxx Transmit FIFO has x words.
		10000 Transmit FIFO has 16 words.
7	TXFFINT	TXFIFO interrupt
		0 TXFIFO interrupt has not occurred, This is a read-only bit.
		1 TXFIFO interrupt has occurred, This is a read-only bit.
6	TXFFINT CLR	TXFIFO clear
		0 Write 0 has no effect on TXFIFINT flag bit, Bit reads back a zero.
		1 Write 1 to clear TXFFINT flag in bit 7.
5	TXFFIENA	TX FIFO interrupt enable
		0 TX FIFO interrupt based on TXFFIVL match (less than or equal to) will be disabled .
		1 TX FIFO interrupt based on TXFFIVL match (less than or equal to) will be enabled.
0−4	TXFFIL4−0	00000 TXFFIL4−0 transmit FIFO interrupt level bits. Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4−0) and FIFO level bits (TXFFIL4−0) match (less than or equal to).
		Default value is 0x00000.

Figure 2−10. SPI FIFO Receive (SPIFFRX) Register − Address 704Bh

15	14	13	12	11	10	9	8
RXFFOVF Flag	RXFFOVF CLR	RXFIFO Reset	RXFFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0	W−0	R/W−1	R−0	R−0	R−0	R−0	R−0

7	6	5	4	3	2	1	0
RXFFINT Flag	RXFFINT CLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−1	R/W−1

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−11. SPI FIFO Receive (SPIFFRX) Register Bit Descriptions

Bit(s)	Name	Description	
15	RXFFOVF	Receive FIFO overflow flag	
		0	Receive FIFO has not overflowed. This is a read-only bit.
		1	Receive FIFO has overflowed, read-only bit. More than 16 words have been received in to the FIFO, and the first received word is lost.
14	RXFFOVF	Receive FIFO overflow clear	
	CLR	0	Write 0 does not affect RXFFOVF flag bit, Bit reads back a zero
		1	Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFO Reset	Receive FIFO reset	
		0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable transmit FIFO operation
8−12	RXFFST4−0	00000	Receive FIFO is empty.
		00001	Receive FIFO has 1 word.
		00010	Receive FIFO has 2 words.
		00011	Receive FIFO has 3 words.
		0xxxx	Receive FIFO has x word.
			Note: 10000: Receive FIFO has 16 words.
7	RXFFINT	Receive FIFO interrupt	
		0	RXFIFO interrupt has not occurred. This is a read-only bit.
		1	RXFIFO interrupt has occurred. This is a read-only bit.
6	RXFFINT CLR	Receive FIFO interrupt clear	
		0	Write 0 has no effect on RXFIFINT flag bit, Bit reads back a zero.
		1	Write 1 to clear RXFFINT flag in bit 7.

Table 2−11. SPI FIFO Receive (SPIFFRX) Register Bit Descriptions (Continued)

Bit(s)	Name	Description
5	RXFFIENA	RX FIFO interrupt enable 0 RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be disabled. 1 RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be enabled.
4−0	RXFFIL4−0	Receive FIFO interrupt level bits 11111 Receive FIFO will generate interrupt when the FIFO status bits (RXFFST4−0) and FIFO level bits (RXFFIL4−0) match (greater than or equal to) the default value of these bits after reset − 11111. This will avoid frequent interrupts after reset, as the receive FIFO will be empty most of the time.

Figure 2−11. SPI FIFO Control (SPIFFCT) Register − Address 704Ch

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
FFTXDLY7	FFTXDLY6	FFTXDLY5	FFTXDLY4	FFTXDLY3	FFTXDLY2	FFTXDLY1	FFTXDLY0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−12. SPI FIFO Control (SPIFFCT) Register Bit Descriptions

Bit(s)	Name	Description
15−8	Reserved	Reserved
7−0	FFTXDLY7−0	FIFO transmit delay bits These bits define the delay between every transfer from FIFO transmit buffer to transmit shift register. The delay is defined in number SPI serial clock cycles. The 8 bit register could define a minimum delay of 0 serial clock cycles and a maximum of 25 serial clock cycles. In the FIFO mode the buffer (TXBUF) between the shift register and the FIFO should be filled only after the shift register has completed shifting of the last bit. This is required to pass on the delay between transfers to the data stream. In the FIFO mode TXBUF should not be treated as one additional level of buffer.

2.1.10 SPI Priority Control Register (SPIPRI)

Figure 2−12. SPI Priority Control Register (SPIPRI) — Address 704Fh

7	6	5	4	3	0
Reserved		SPI SUSP SOFT	SPI SUSP FREE	Reserved	
R-0		R/W	R/W−0	R−0	

Legend: R = Read access, W = Write access, -0 = value after reset

Table 2−13. SPI Priority Control Register (SPIPRI) Bit Descriptions

Bit(s)	Name	Description		
7−6	Reserved	Reads return zero; writes have no effect.		
5−4	SPI SUSP SOFT SPI SUSP FREE	These bits determine what occurs when an emulation suspend occurs (for example, when the debugger hits a breakpoint). The peripheral can continue whatever it is doing (free-run mode) or, if in stop mode, it can either stop immediately or stop when the current operation (the current receive/transmit sequence) is complete.		
		Bit 5 SOFT	Bit 4 FREE	
		0	0	Transmission will stop after midway in the bit stream while TSPEND is asserted. Once TSUSPEND is deasserted without a system reset, the remainder of the bits pending in the DATBUF will be shifted. Example: If SPIDAT has shifted 3 out of 8 bits, the communication will freeze right there. However, if TSUSPEND is later deasserted without resetting the SPI, SPI will start transmitting from where it had stopped (fourth bit in this case) and will transmit 8 bits from that point. The SCI module operates differently.
		1	0	If the emulation suspend occurs before the start of a transmission, (i.e., before the first SPICLK pulse) then the transmission will not occur. If the emulation suspend occurs after the start of a transmission, then the data will be shifted out to completion. When the start of transmission occurs is dependent on the baud rate used. *Standard SPI mode:* Stop after transmitting the words in the shift register and buffer. That is after TXBUF and SPIDAT is empty. *In the FIFO mode:* Stop after transmitting the words in the shift register and buffer. That is after TX FIFO and SPIDAT is empty.
		x	1	Free run, continue SPI operation regardless of suspend or when the suspend occurred.
3−0	Reserved	Reads return zero; writes have no effect.		

2.2 SPI Example Waveforms

Figure 2−13. CLOCK POLARITY = 0, CLOCK PHASE = 0 (All data transitions are during the rising edge, non-delayed clock. Inactive level is low.)

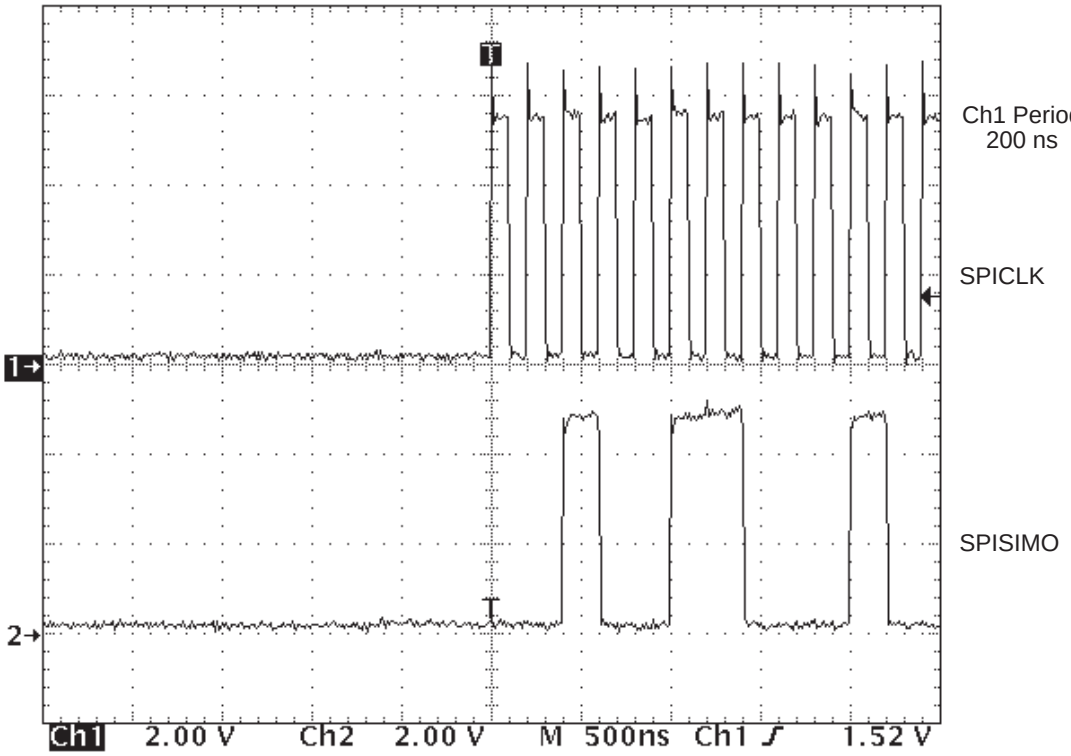


Figure 2−14. CLOCK POLARITY = 0, CLOCK PHASE = 1 (All data transitions are during the rising edge, but delayed by half clock cycle. Inactive level is low.)

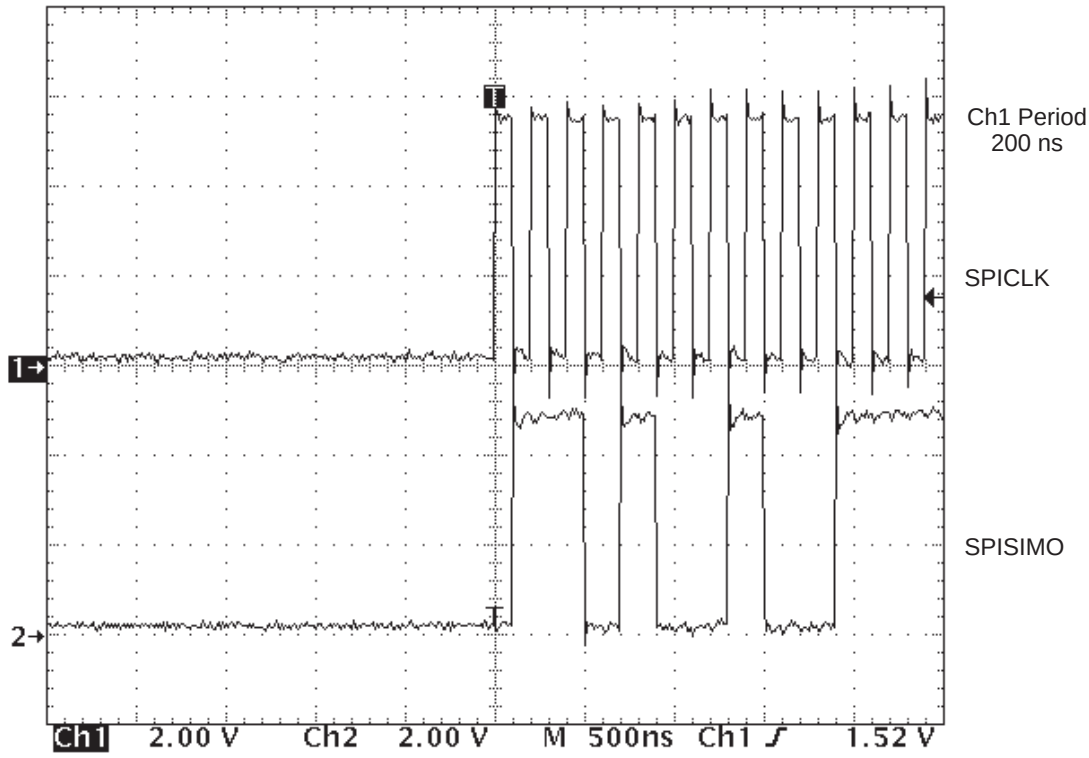


Figure 2−15. CLOCK POLARITY = 1, CLOCK PHASE = 0 (All data transitions are during the falling edge. Inactive level is high.)

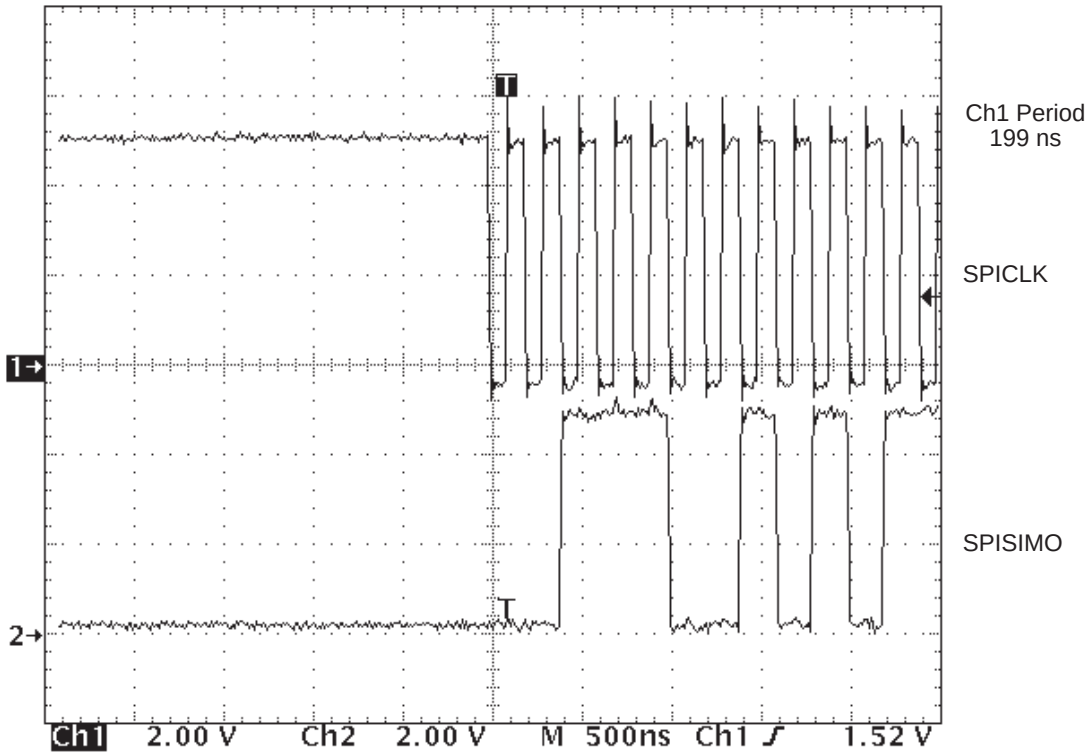


Figure 2−16. CLOCK POLARITY = 1, CLOCK PHASE = 1 (All data transitions are during the falling edge, but delayed by half clock cycle. Inactive level is high.)

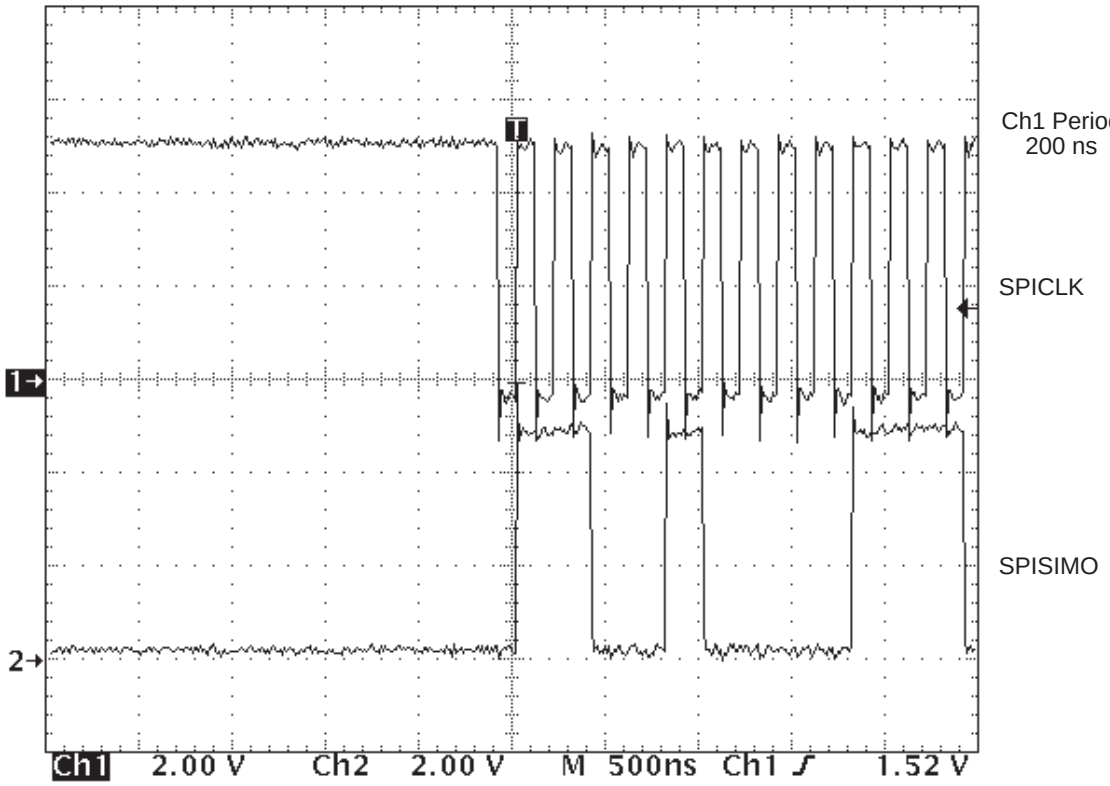


Figure 2−17. $\overline{\text{SPISTE}}$ Behavior in Master Mode (Master lowers $\overline{\text{SPISTE}}$ during the entire 16 bits of transmission.)

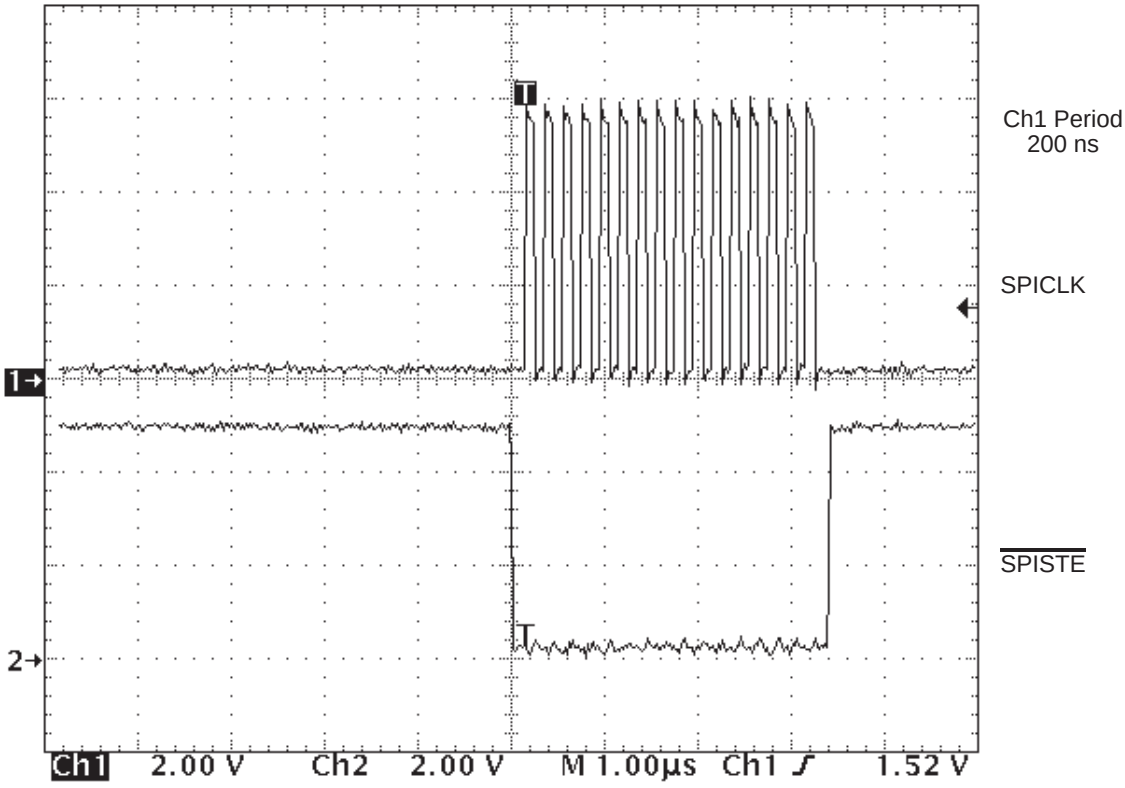
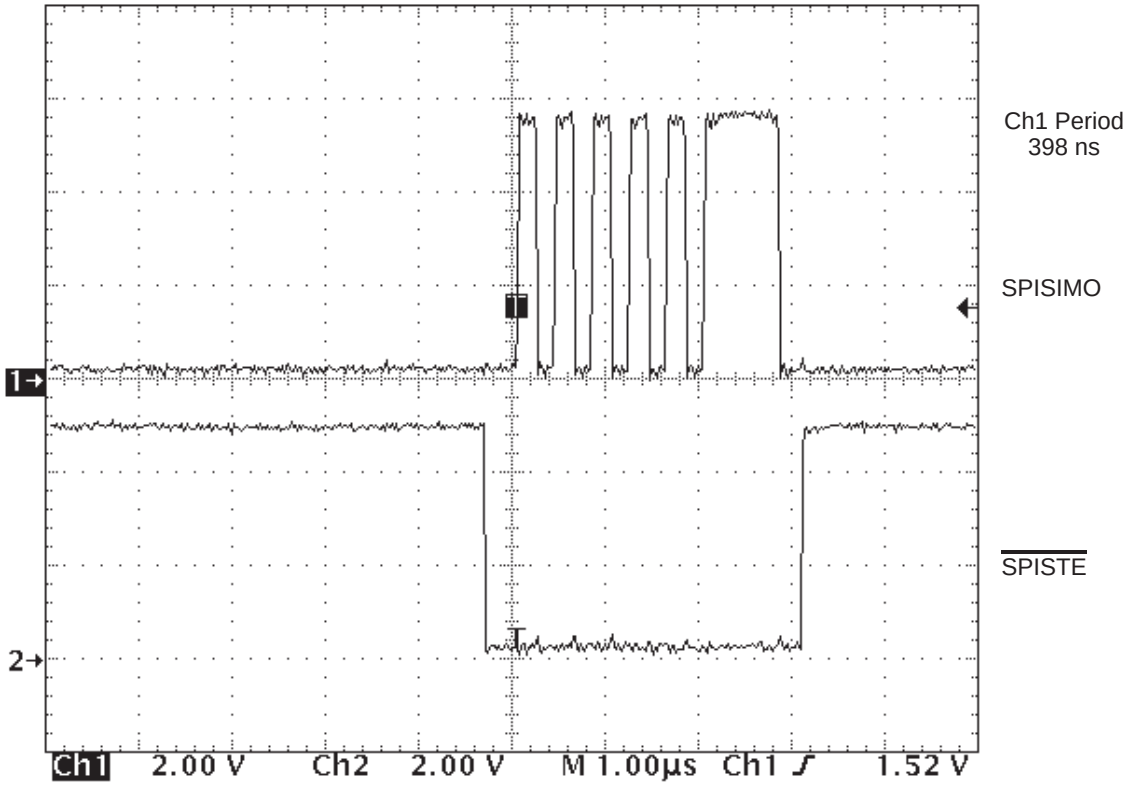


Figure 2−18. $\overline{\text{SPISTE}}$ Behavior in Slave Mode (Slave's $\overline{\text{SPISTE}}$ is lowered during the entire 16 bits of transmission.)



Appendix A

Revision History

This document was revised to SPRU059B from SPRU059A.

The scope of the revisions was limited to adding technical changes as described on the next page.

A.1 Changes Made in This Revision

The following changes were made in this revision:

Page	Additions/Modifications/Deletions
1-1	Changed the title to include new devices and added a paragraph including those devices
1-4	Added a note to Figure 1−2
1-5	Modified the description of $\overline{\text{SPISTE}}$
1-10	Modified the last paragraph in Master Mode description
2-12	Modified the description for bit 14 (SPIFFENA) in Table 2−10

Index

W

TMS320x281x DSP Analog-to-Digital Converter (ADC) Reference Guide

Literature Number: SPRU060D
June 2002 − Revised July 2005

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This guide describes how analog−to−digital converter (ADC) module works in TMS320x281x devices.

Related Documentation From Texas Instruments

The following books describe the TMS320x281x and related support tools that are available on the TI website.

TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, and TMS320C2812 Digital Signal Processors (literature number SPRS174) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320R2811 and TMS320R2812 Digital Signal Processors (literature number SPRS257) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C28x DSP CPU and Instruction Set Reference Guide (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x™ fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320x281x Boot ROM Reference Guide (literature number SPRU095) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x281x System Control and Interrupts Reference Guide (literature number SPRU078) describes the various interrupts and system control features of the 281x digital signal processors (DSPs).

TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide (literature number SPRU074) describes the eCAN that uses

established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

TMS320x281x Event Manager (EV) Reference Guide (literature number SPRU065) describes the EV modules that provide a broad range of functions and features that are particularly useful in motion control and motor control applications. The EV modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits.

TMS320x281x External Interface (XINTF) Reference Guide (literature number SPRU067) describes the external interface (XINTF) of the 28x digital signal processors (DSPs).

TMS320x281x Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU061) describes the McBSP) available on the C28x devices. The McBSPs allow direct interface between a DSP and other devices in a system.

TMS320x281x, 280x Peripheral Reference Guide (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide (literature number SPRU051) describes the SCI that is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

TMS320x281x, 280x Serial Peripheral Interface (SPI) Reference Guide (literature number SPRU059) describes the SPI − a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit−transfer rate. The SPI is used for communications between the DSP controller and external peripherals or another controller.

TMS320x281x System Control and Interrupts Reference Guide (literature number SPRU078) describes the various interrupts and system control features of the 281x digital signal processors (DSPs).

The TMS320C28x Instruction Set Simulator Technical Overview (literature number SPRU608) describes the simulator, available within the

Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU625) describes development using DSP/BIOS.

3.3 V DSP for Digital Motor Control Application Report (literature number SPRA550). This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications. The application report first describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V − 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system's noise and EMI effects are summarized in the last section.

Thermo-Electric Cooler Control Using a TMS320F2812 DSP & DRV592 Power Amplifier Application Note (literature number SPRA873). This application report presents a thermoelectric cooler system consisting of a Texas Instruments TMS320F2812 digital signal processor (DSP) and DRV592 power amplifier. A complete description of the experimental system, along with software and software operating instructions, are provided.

Running an Application from Internal Flash Memory on the TMS320F281x DSP Application Report (literature number SPRA958). This application report covers the requirements needed to properly configure application software for execution from on-chip flash memory. Requirements for both DSP/BIOS™ and non-DSP/BIOS projects are presented. Some performance considerations and techniques are also discussed. Example code projects are included that run from on-chip flash on the eZdsp™ F2812 development board (or alternately any F2812, F2811, or F2810 DSP board). Code examples that run from internal RAM are also provided for completeness. These code examples provide a starting point for code development, if desired.

Trademarks

Code Composer Studio and C28x are trademarks of Texas Instruments.

This page intentionally left blank.

Contents

Figures

Tables

Examples

Chapter 1

Analog-to-Digital Converter (ADC)

The TMS320F28x™ ADC module is a 12-bit pipelined analog-to-digital converter (ADC). The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample-and-hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

This reference guide is applicable for the ADC found on the TMS320x281x family of processors. This includes all Flash-based, ROM-based, and RAM-based devices within the 281x family.

Topic **Page**

TMS320F28x is a trademark of Texas Instruments.

1.1 Features

The ADC module has 16 channels, configurable as two independent 8-channel modules to service event managers A and B. The two independent 8-channel modules can be cascaded to form a 16-channel module. Although there are multiple input channels and two sequencers, there is only one converter in the ADC module. Figure 1−1 shows the block diagram of the F2810, F2811, and F2812 ADC module.

The two 8-channel modules have the capability to autosequence a series of conversions, each module has the choice of selecting any one of the respective eight channels available through an analog MUX. In the cascaded mode, the autosequencer functions as a single 16-channel sequencer. On each sequencer, once the conversion is complete, the selected channel value is stored in its respective ADCRESULT register. Autosequencing allows the system to convert the same channel multiple times, allowing the user to perform oversampling algorithms. This gives increased resolution over traditional single-sampled conversion results.

Functions of the ADC module include:

- 12-bit ADC core with built-in dual sample-and-hold (S/H)
- Simultaneous sampling or sequential sampling modes
- Analog input: 0 V to 3 V
- Fast conversion time runs at 25 MHz, ADC clock, or 12.5 MSPS
- 16-channel, multiplexed inputs
- Autosequencing capability provides up to 16 “autoconversions” in a single session. Each conversion can be programmed to select any 1 of 16 input channels
- Sequencer can be operated as two independent 8-state sequencers or as one large 16-state sequencer (i.e., two cascaded 8-state sequencers)
- Sixteen result registers (individually addressable) to store conversion values
  - The digital value of the input analog voltage is derived by:

$$\text{Digital Value} = 4095 \times \frac{\text{Input Analog Voltage} - \text{ADCLO}}{3}$$

- Multiple triggers as sources for the start-of-conversion (SOC) sequence
  - S/W − software immediate start

- EVA − Event manager A (multiple event sources within EVA)
- EVB − Event manager B (multiple event sources within EVB)
- External pin

❑ Flexible interrupt control allows interrupt request on every end-of-sequence (EOS) or every other EOS

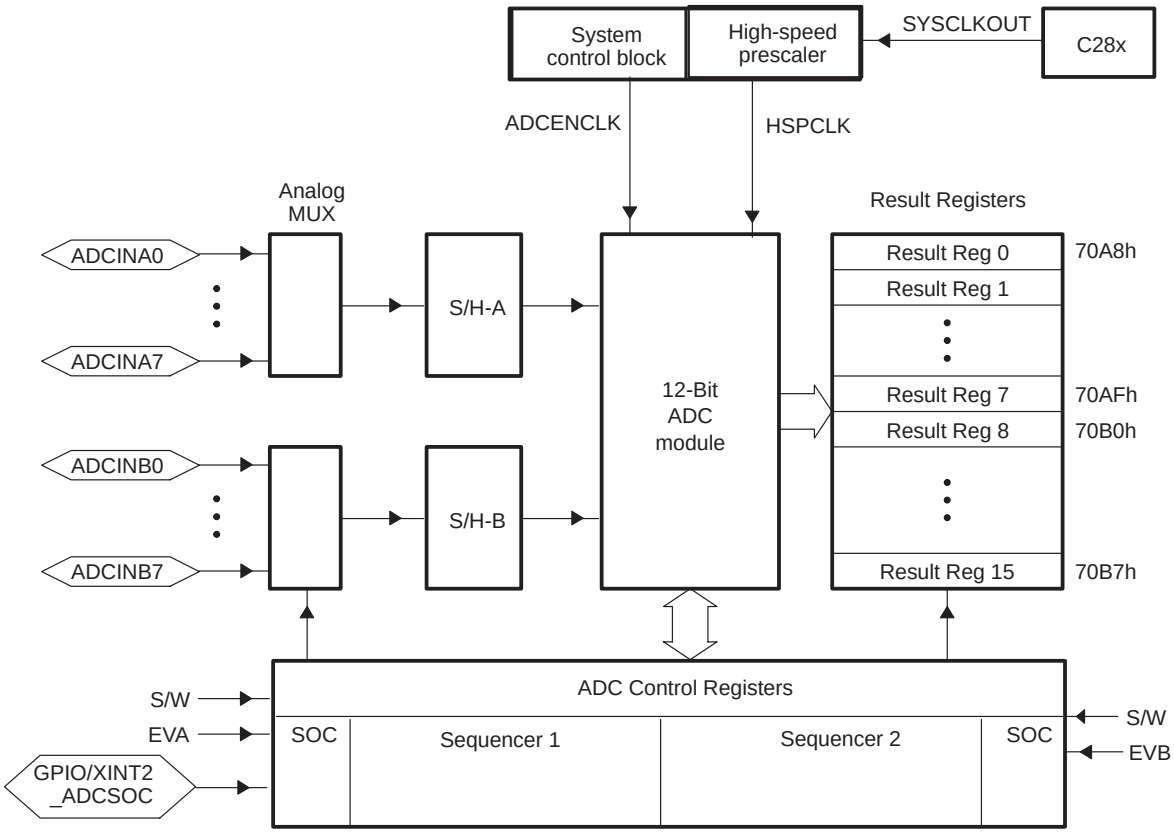
❑ Sequencer can operate in “start/stop” mode, allowing multiple “time-sequenced triggers” to synchronize conversions

❑ EVA and EVB triggers can operate independently in dual-sequencer mode

❑ Sample-and-hold (S/H) acquisition time window has separate prescale control

❑ Sequencer override mode enhancement is available only in the F2810/F2811/F2812 silicon after revision B.

Figure 1−1. Block Diagram of the ADC Module



To obtain the specified accuracy of the ADC, proper board layout is very critical. To the best extent possible, traces leading to the ADCINxx pins should not run in close proximity to the digital signal paths. This is to minimize switching noise on the digital lines from getting coupled to the ADC inputs. Furthermore, proper isolation techniques must be used to isolate the ADC module power pins from the digital supply.

Table 1−1. ADC Registers

NAME	ADDRESS RANGE	SIZE (x16)†	DESCRIPTION
ADCTRL1	0x0000−7100	1	ADC Control Register 1
ADCTRL2	0x0000−7101	1	ADC Control Register 2
ADCMAXCONV	0x0000−7102	1	ADC Maximum Conversion Channels Register
ADCCHSELSEQ1	0x0000−7103	1	ADC Channel Select Sequencing Control Register 1
ADCCHSELSEQ2	0x0000−7104	1	ADC Channel Select Sequencing Control Register 2
ADCCHSELSEQ3	0x0000−7105	1	ADC Channel Select Sequencing Control Register 3
ADCCHSELSEQ4	0x0000−7106	1	ADC Channel Select Sequencing Control Register 4
ADCASEQSR	0x0000−7107	1	ADC Auto-Sequence Status Register
ADCRESULT0	0x0000−7108	1	ADC Conversion Result Buffer Register 0
ADCRESULT1	0x0000−7109	1	ADC Conversion Result Buffer Register 1
ADCRESULT2	0x0000−710A	1	ADC Conversion Result Buffer Register 2
ADCRESULT3	0x0000−710B	1	ADC Conversion Result Buffer Register 3
ADCRESULT4	0x0000−710C	1	ADC Conversion Result Buffer Register 4
ADCRESULT5	0x0000−710D	1	ADC Conversion Result Buffer Register 5
ADCRESULT6	0x0000−710E	1	ADC Conversion Result Buffer Register 6
ADCRESULT7	0x0000−710F	1	ADC Conversion Result Buffer Register 7
ADCRESULT8	0x0000−7110	1	ADC Conversion Result Buffer Register 8
ADCRESULT9	0x0000−7111	1	ADC Conversion Result Buffer Register 9
ADCRESULT10	0x0000−7112	1	ADC Conversion Result Buffer Register 10
ADCRESULT11	0x0000−7113	1	ADC Conversion Result Buffer Register 11
ADCRESULT12	0x0000−7114	1	ADC Conversion Result Buffer Register 12
ADCRESULT13	0x0000−7115	1	ADC Conversion Result Buffer Register 13
ADCRESULT14	0x0000−7116	1	ADC Conversion Result Buffer Register 14
ADCRESULT15	0x0000−7117	1	ADC Conversion Result Buffer Register 15
ADCTRL3	0x0000−7118	1	ADC Control Register 3
ADCST	0x0000−7119	1	ADC Status Register
reserved	0x0000−711A 0x0000−711F	6	

† The registers in this table are mapped to Peripheral Frame 1. This space only allows 16-bit accesses. 32-bit accesses produce undefined results.

1.2 Autoconversion Sequencer Principle of Operation

The ADC sequencer consists of two independent 8-state sequencers (SEQ1 and SEQ2) that can also be cascaded together to form one 16-state sequencer (SEQ). The word “state” represents the number of autoconversions that can be performed with the sequencer. Block diagrams of the single (16-state, cascaded) and dual (two 8-state, separated) sequencer modes are shown in Figure 1−4 and Figure 1−5, respectively.

In both cases, the ADC has the ability to autosequence a series of conversions. This means that each time the ADC receives a start-of-conversion request, it can perform multiple conversions automatically. For every conversion, any one of the available 16 input channels can be selected through the analog mux. After conversion, the digital value of the selected channel is stored in the appropriate result register (ADCRESULTn). (The first result is stored in ADCRESULT0, the second result in ADCRESULT1, and so on). It is also possible to sample the same channel multiple times, allowing the user to perform “over-sampling”, which gives increased resolution over traditional single-sampled conversion results.

Note: Dual-Sequencer Mode

In the sequential sampling dual-sequencer mode, a pending SOC request from either sequencer is taken up as soon as the sequence initiated by the currently active sequencer is completed. For example, assume that the A/D converter is busy catering to SEQ2 when an SOC request from SEQ1 occurs. The A/D converter will start SEQ1 immediately after completing the request in progress on SEQ2. If SOC requests are pending from both SEQ1 and SEQ2, the SOC for SEQ1 has priority. For example, assume that the A/D converter is busy catering to SEQ1. During that process, SOC requests from both SEQ1 and SEQ2 are made. When SEQ1 completes its already active sequence, the SOC request for SEQ2 will remain pending.

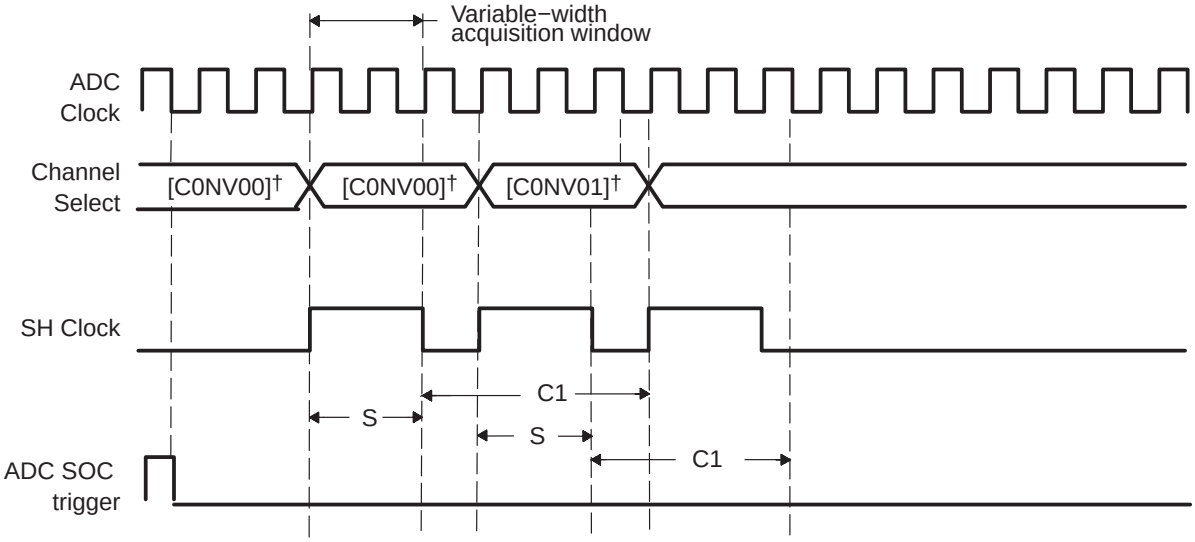
The ADC can also operate in simultaneous sampling mode or sequential sampling mode. For each conversion (or pair of conversions in simultaneous sampling mode), the current CONVxx bit field defines the pin (or pair of pins) to be sampled and converted. In sequential sampling mode, all four bits of CONVxx define the input pin. The MSB defines with which sample-and-hold buffer the input pin is associated, and the three LSBs define the offset. For example, if CONVxx contains the value 0101b, ADCINA5 is the selected input pin. If it contains the value 1011b, ADCINB3 is the selected input pin. In simultaneous sampling mode, the MSB of the CONVxx register is discarded. Each sample and hold buffer samples the associated pin given by the offset provided in the three LSBs of the CONVxx register. For instance, if the

CONVxx register contains the value 0110b, ADCINA6 is sampled by S/H-A and ADCINB6 is sampled by S/H-B. If the value is 1001b, ADCINA1 is sampled by S/H-A and ADCINB1 is sampled by S/H-B. The voltage in S/H-A is converted first, followed by the S/H-B voltage. The result of the S/H-A conversion is placed in the current ADCRESULTn register (ADCRESULT0 for SEQ1, assuming the sequencer has been reset). The result of the S/H-B conversion is placed in the next ADCRESULTn register (ADCRESULT1 for SEQ1, assuming the sequencer has been reset). The result register pointer is then increased by two (to point to ADCRESULT2 for SEQ1, assuming the sequencer had originally been reset).

1.2.1 Sequential Sampling Mode

Figure 1−2 shows the timing of sequential sampling mode. In this example, the ACQ_PS3−0 bits are set to 0001b.

Figure 1−2. Sequential Sampling Mode (SMODE=0)



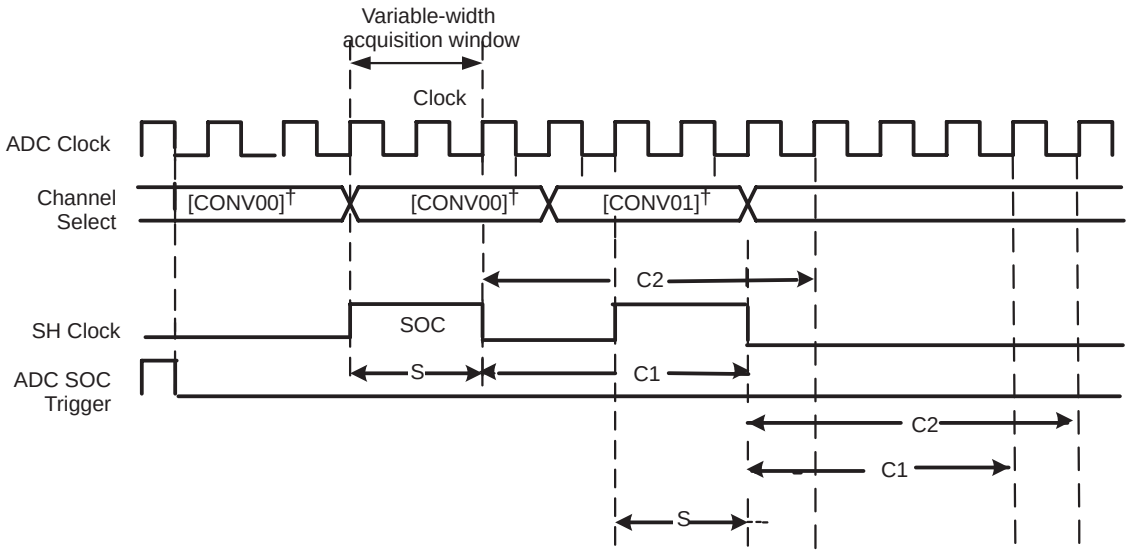
† ADC channel address contained in [CONV00] 4-bit register; CONV00 for SEQ1 and CONV08 for SEQ2

Legend: C1 − Duration of time for result register update
S − Acquisition window

1.2.2 Simultaneous Sampling Mode

Figure 1−3 describes the timing of simultaneous sampling mode. In this example, the ACQ_PS3 bits are set to 0001b.

Figure 1−3. Simultaneous Sampling Mode (SMODE=1)



† ADC channel address contained in [CONV00] 4-bit register;
[CONV00] means A0/B0 channels;
[CONV01] means A1/B1 channels.

Legend: C1 − Duration of time for Ax channel result in result register
C2 − Duration of time for Bx channel result in result register
S − Acquisition window

Figure 1−4. Block Diagram of Autosequenced ADC in Cascaded Mode

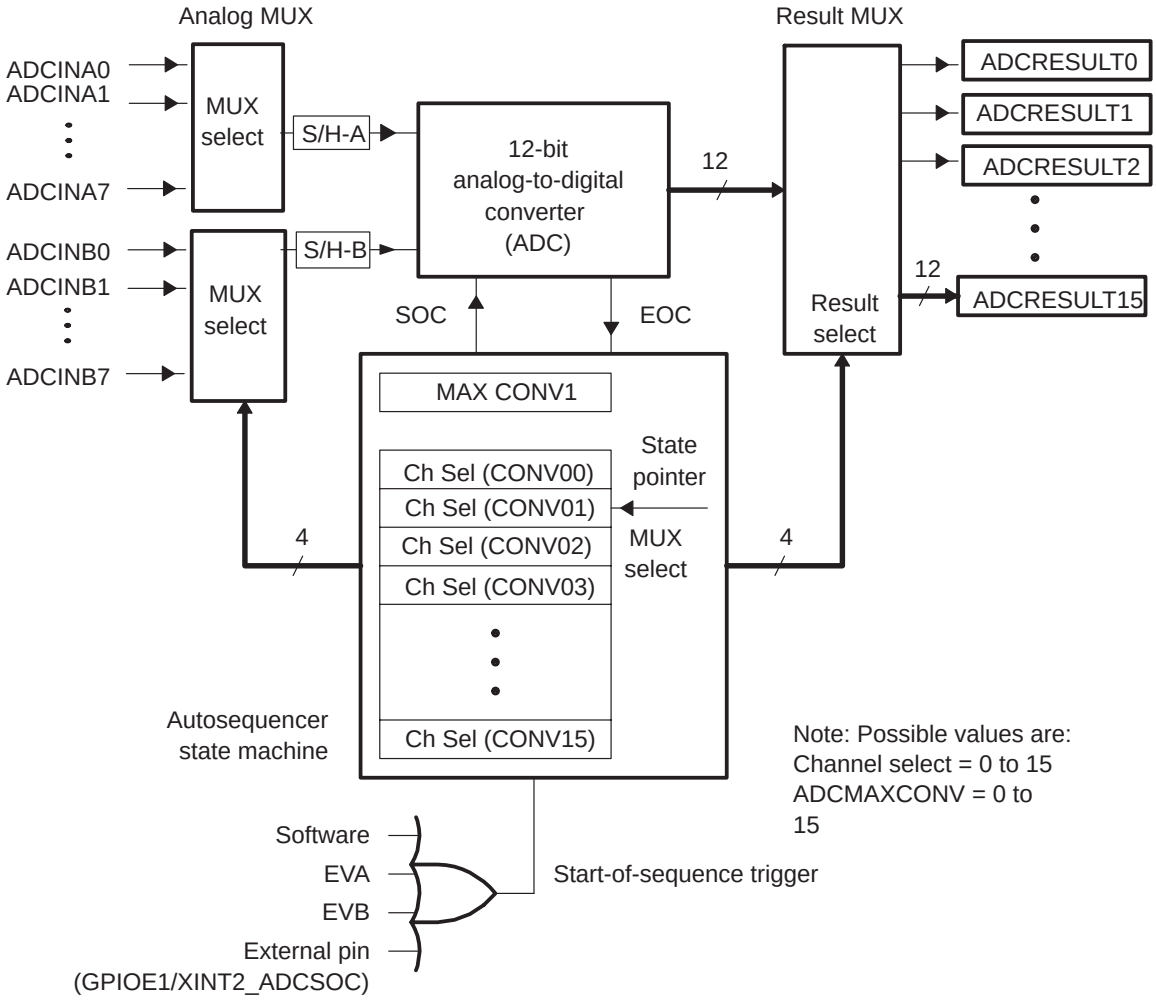
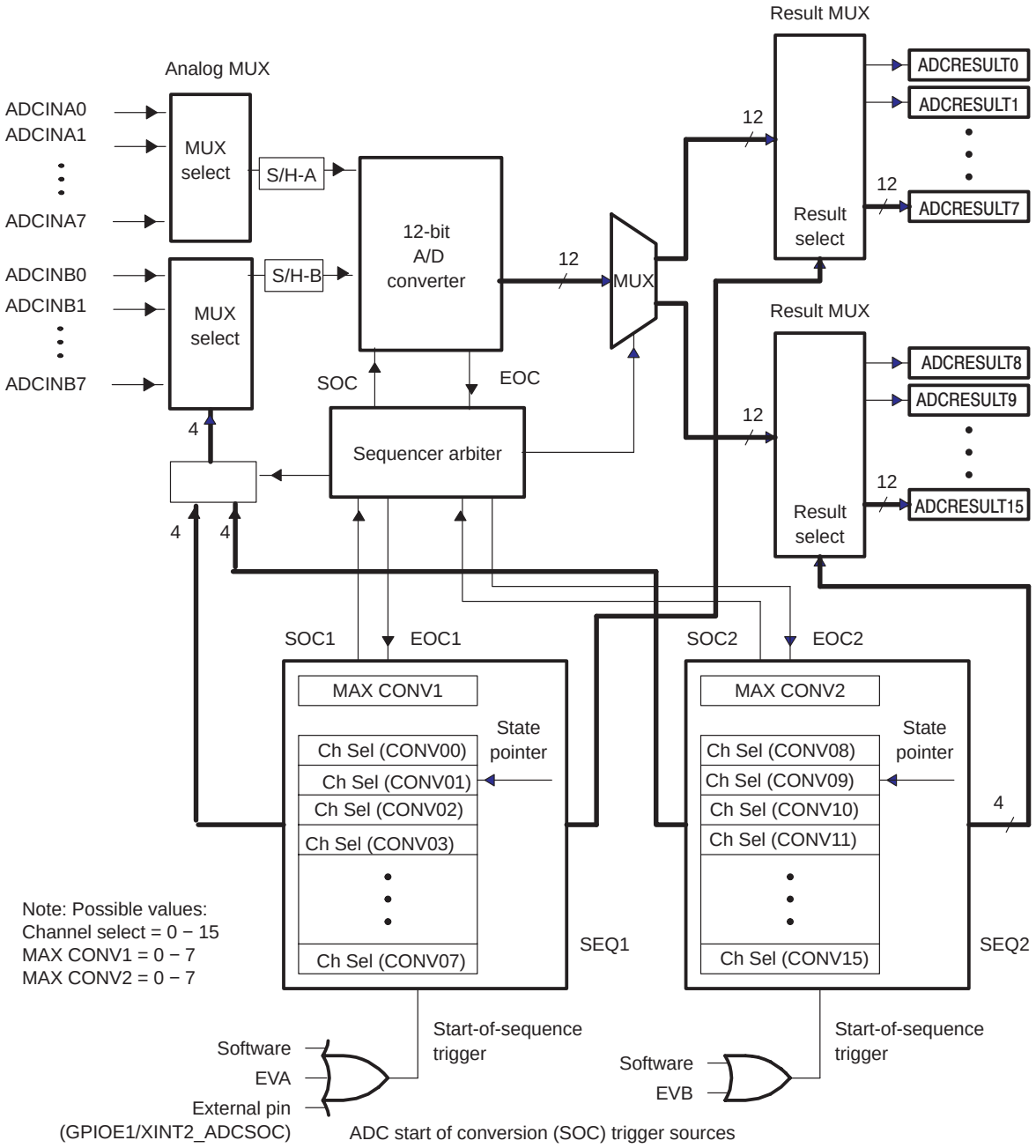


Figure 1−5. Block Diagram of Autosequenced ADC With Dual Sequencers



Note: One ADC Shared in Dual-Sequencer Mode

There is only one ADC in the DSP. This converter is shared by the two sequencers in dual-sequencer mode.

The sequencer operation for both 8-state and 16-state modes is almost identical; the few differences are highlighted in Table 1−2.

Table 1−2. Comparison of Single and Cascaded Operating Modes

Feature	Single 8-state sequencer #1 (SEQ1)	Single 8-state sequencer #2 (SEQ2)	Cascaded 16-state sequencer (SEQ)
Start-of-conversion (SOC) triggers	EVA, software, external pin	EVB, software	EVA, EVB, software, external pin
Maximum number of autoconversions (i.e., sequence length)	8	8	16
Autostop at end-of-sequence (EOS)	Yes	Yes	Yes
Arbitration priority	High	Low	Not applicable
ADC conversion result register locations	0 to 7	8 to 15	0 to 15
ADCCHSELSEQn bit field assignment	CONV00 to CONV07	CONV08 to CONV15	CONV00 to CONV15

For convenience, the sequencer states will be subsequently referred to as:

- For SEQ1: CONV00 to CONV07
- For SEQ2: CONV08 to CONV15
- For Cascaded SEQ: CONV00 to CONV15

The analog input channel selected for each sequenced conversion is defined by CONVnn bit fields in the ADC input channel select sequencing control registers (ADCCHSELSEQn). CONVnn is a 4-bit field that specifies any one of the 16 channels for conversion. Since a maximum of 16 conversions in a sequence is possible when using the sequencers in cascaded mode, 16 such 4-bit fields (CONV00 – CONV15) are available and are spread across four 16-bit registers (ADCCHSELSEQ1 – ADCCHSELSEQ4). The CONVnn bits can have any value from 0 to 15. The analog channels can be chosen in any desired order and the same channel may be selected multiple times.

1.2.3 Simultaneous Sampling Dual Sequencer Mode Example

Example initialization:

```
AdcRegs.ADCTRL3.bit.SMODE_SEL = 1;       // Setup simultaneous sampling mode
AdcRegs.ADCMAXCONV.all = 0x0033;         // 4 double conv's each sequencer (8 total)
AdcRegs.ADCCHSELSEQ1.bit.CONV00 = 0x0;       // Setup conv from ADCINA0 & ADCINB0
AdcRegs.ADCCHSELSEQ1.bit.CONV01 = 0x1;       // Setup conv from ADCINA1 & ADCINB1
AdcRegs.ADCCHSELSEQ1.bit.CONV02 = 0x2;       // Setup conv from ADCINA2 & ADCINB2
AdcRegs.ADCCHSELSEQ1.bit.CONV03 = 0x3;       // Setup conv from ADCINA3 & ADCINB3
AdcRegs.ADCCHSELSEQ3.bit.CONV08 = 0x4;       // Setup conv from ADCINA4 & ADCINB4
AdcRegs.ADCCHSELSEQ3.bit.CONV09 = 0x5;       // Setup conv from ADCINA5 & ADCINB5
AdcRegs.ADCCHSELSEQ3.bit.CONV10 = 0x6;       // Setup conv from ADCINA6 & ADCINB6
AdcRegs.ADCCHSELSEQ3.bit.CONV11 = 0x7;       // Setup conv from ADCINA7 & ADCINB7
```

If SEQ1 and SEQ2 were both executed, the results would go to the following RESULT registers:

```
ADCINA0 -> ADCRESULT0
ADCINB0 -> ADCRESULT1
ADCINA1 -> ADCRESULT2
ADCINB1 -> ADCRESULT3
ADCINA2 -> ADCRESULT4
ADCINB2 -> ADCRESULT5
ADCINA3 -> ADCRESULT6
ADCINB3 -> ADCRESULT7
ADCINA4 -> ADCRESULT8
ADCINB4 -> ADCRESULT9
ADCINA5 -> ADCRESULT10
ADCINB5 -> ADCRESULT11
ADCINA6 -> ADCRESULT12
ADCINB6 -> ADCRESULT13
ADCINA7 -> ADCRESULT14
ADCINB7 -> ADCRESULT15
```

1.2.4 Simultaneous Sampling Cascaded Sequencer Mode Example

```
AdcRegs.ADCTRL3.bit.SMODE_SEL = 1;      // Setup simultaneous sampling mode
AdcRegs.ADCTRL1.bit.SEQ_CASC  = 1;      // Setup cascaded sequencer mode
AdcRegs.ADCMAXCONV.all           = 0x0007; // 8 double conv's (16 total)
AdcRegs.ADCCHSELSEQ1.bit.CONV00  = 0x0;    // Setup conv from ADCINA0 & ADCINB0
AdcRegs.ADCCHSELSEQ1.bit.CONV01  = 0x1;    // Setup conv from ADCINA1 & ADCINB1
AdcRegs.ADCCHSELSEQ1.bit.CONV02  = 0x2;    // Setup conv from ADCINA2 & ADCINB2
AdcRegs.ADCCHSELSEQ1.bit.CONV03  = 0x3;    // Setup conv from ADCINA3 & ADCINB3
AdcRegs.ADCCHSELSEQ2.bit.CONV04  = 0x4;    // Setup conv from ADCINA4 & ADCINB4
AdcRegs.ADCCHSELSEQ2.bit.CONV05  = 0x5;    // Setup conv from ADCINA5 & ADCINB5
AdcRegs.ADCCHSELSEQ2.bit.CONV06  = 0x6;    // Setup conv from ADCINA6 & ADCINB6
AdcRegs.ADCCHSELSEQ2.bit.CONV07  = 0x7;    // Setup conv from ADCINA7 & ADCINB7
```

If the cascaded SEQ was executed, the results would go to the following ADCRESULT registers:

```
ADCINA0 -> ADCRESULT0
ADCINB0 -> ADCRESULT1
ADCINA1 -> ADCRESULT2
ADCINB1 -> ADCRESULT3
ADCINA2 -> ADCRESULT4
ADCINB2 -> ADCRESULT5
ADCINA3 -> ADCRESULT6
ADCINB3 -> ADCRESULT7
ADCINA4 -> ADCRESULT8
ADCINB4 -> ADCRESULT9
ADCINA5 -> ADCRESULT10
ADCINB5 -> ADCRESULT11
ADCINA6 -> ADCRESULT12
ADCINB6 -> ADCRESULT13
ADCINA7 -> ADCRESULT14
ADCINB7 -> ADCRESULT15
```

1.3 Uninterrupted Autosequenced Mode

The following description applies to the 8-state sequencers (SEQ1 or SEQ2). In this mode, SEQ1/SEQ2 can autosequence up to eight conversions of any channel in a single sequencing session (16 when sequencers are cascaded together). Figure 1−6 shows the flow diagram. The result of each conversion is stored in one of the eight result registers (ADCRESULT0 – ADCRESULT7 for SEQ1 and ADCRESULT8 – ADCRESULT15 for SEQ2). These registers are filled from the lowest address to the highest address.

The number of conversions in a sequence is controlled by MAX CONVn (a 3-bit or 4-bit field in the ADCMAXCONV register), which is automatically loaded into the sequencing counter status bits (SEQ CNTR3 – 0) in the autosequence status register (ADCASEQSR) at the start of an autosequenced conversion session. The MAX CONVn field can have a value ranging from zero to seven (0 to 15 when sequencers are cascaded together). SEQ CNTRn bits count down from their loaded value as the sequencer starts from state CONV00 and continues sequentially (CONV01, CONV02, and so on) until SEQ CNTRn has reached zero. The number of conversions completed during an autosequencing session is equal to (MAX CONVn + 1).

Example 1−1. Conversion in Dual-Sequencer Mode Using SEQ1

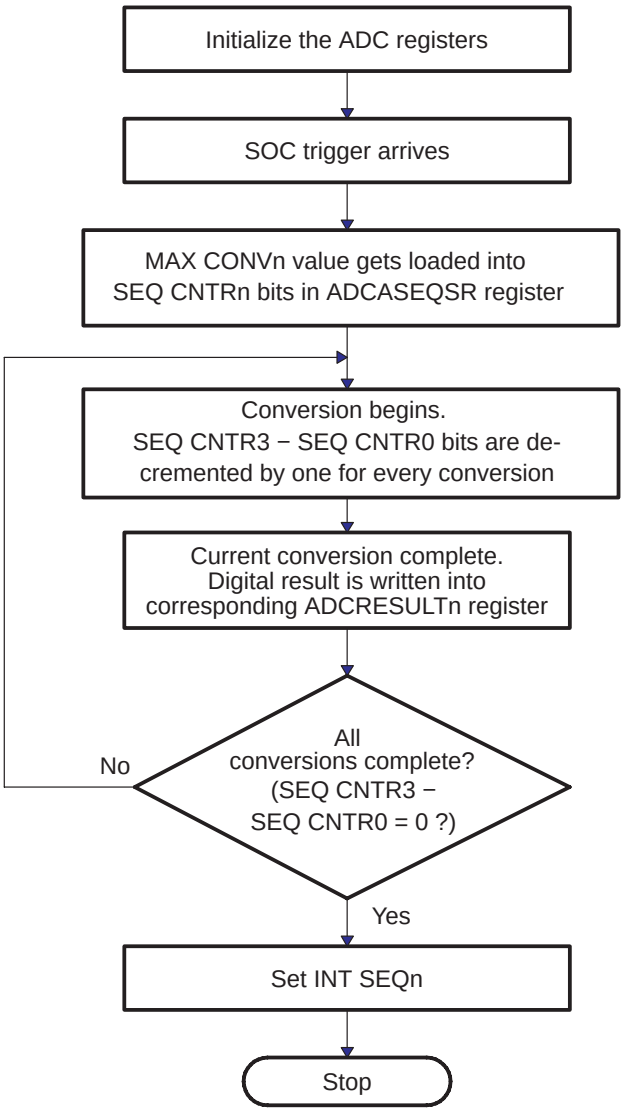
Suppose seven conversions are desired from SEQ1 (i.e., inputs ADCINA2 and ADCINA3 twice, then ADCINA6, ADCINA7, and ADCINB4 must be converted as part of the autosequenced session), then MAX CONV1 should be set to 6 and the ADCCHSELSEQn registers should be set to the values shown in the table below:

	Bits 15−12	Bits 11−8	Bits 7−4	Bits 3−0	
70A3h	3	2	3	2	ADCCHSELSEQ1
70A4h	x	12	7	6	ADCCHSELSEQ2
70A5h	x	x	x	x	ADCCHSELSEQ3
70A6h	x	x	x	x	ADCCHSELSEQ4

Note: Values are in decimal, and x = don’t care

Conversion begins once the start-of-conversion (SOC) trigger is received by the sequencer. The SOC trigger also loads the SEQ CNTRn bits. Those channels that are specified in the ADCCHSELSEQn registers are taken up for conversion, in the predetermined sequence. The SEQ CNTRn bits are decremented by one automatically after every conversion. Once SEQ CNTRn reaches zero, two things can happen, depending on the status of the continuous run bit (CONT RUN) in the ADCTRL1 register.

Figure 1−6. Flow Chart for Uninterrupted Autosequenced Mode



Note: Flow chart corresponds to CONT RUN bit = 0 and INT MOD SEQn bit = 0.

- If CONT RUN is set, the conversion sequence starts all over again automatically (i.e., SEQ CNTRn gets reloaded with the original value in MAX CONV1 and SEQ1 state is set to CONV00 [See Section 1.7 for more options). In this case, to avoid overwriting the data, you must ensure that the result registers are read before the next conversion sequence begins. The arbitration logic designed into the ADC ensures that the result registers are not corrupted should a contention arise (ADC module trying to write into the result registers while you try to read from them at the same time).

- If CONT RUN is not set, the sequencer stays in the last state (CONV06, in this example) and SEQ CNTRn continues to hold a value of zero. To repeat the sequence on the next SOC, the sequencer must be reset using the RST SEQn bit prior to the next SOC.

If the interrupt flag is set every time SEQ CNTRn reaches zero (INT ENA SEQn = 1 and INT MOD SEQ1 = 0), you can (if needed) manually reset the sequencer (using the RST SEQn bit in the ADCTRL2 register) in the interrupt service routine (ISR). This causes the SEQn state to be reset to its original value (CONV00 for SEQ1 and CONV08 for SEQ2). This feature is useful in the Start/Stop operation of the sequencer. Example 1−1 also applies to SEQ2 and the cascaded 16-state sequencer (SEQ) with differences outlined in Table 1−2.

1.3.1 Sequencer Start/Stop Mode (Sequencer “Start/Stop” Operation With Multiple “Time-Sequenced Triggers”)

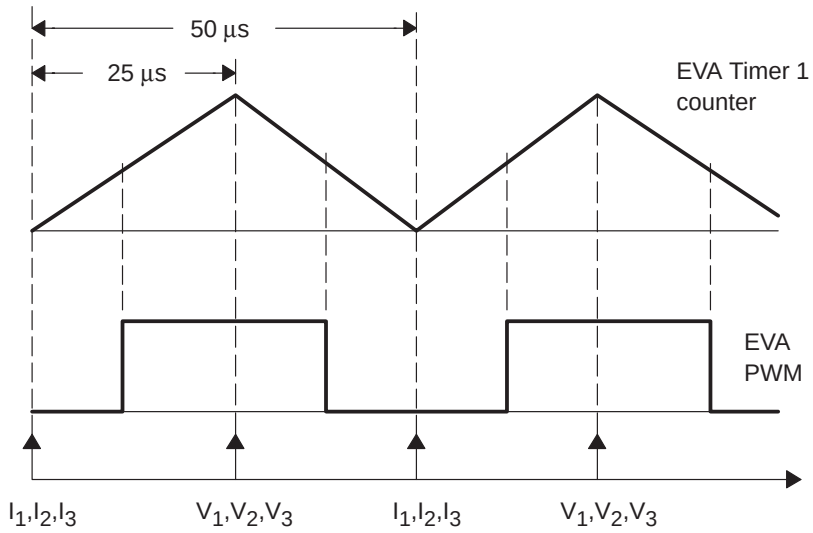
In addition to the uninterrupted autosequenced mode, any sequencer (SEQ1, SEQ2, or SEQ) can be operated in a stop/start mode which is synchronized to multiple start-of-conversion (SOC) triggers, separated in time. This mode is identical to Example 1−1, but the sequencer is allowed to be retriggered without being reset to the initial state CONV00, once it has finished its first sequence (i.e., the sequencer is not reset in the interrupt service routine). Therefore, when one conversion sequence ends, the sequencer stays in the current conversion state. The continuous run bit (CONT RUN) in the ADCTRL1 register must be set to zero (i.e., disabled) for this mode.

Example 1−2. Sequencer Start/Stop Operation

Requirement: To start three autoconversions (e.g., I_1,I_2,I_3) off trigger 1 (underflow) and three autoconversions (e.g., V_1,V_2,V_3) off trigger 2 (period). Triggers 1 and 2 are separated in time by, say, 25 μs and are provided by Event Manager A (EVA). See Figure 1−7. Only SEQ1 is used in this case.

Note: Triggers 1 and 2 may be an SOC signal from EVA, external pin, or software. The same trigger source may occur twice to satisfy the dual-trigger requirement of this example.

Figure 1−7. Example of Event Manager Triggers to Start the Sequencer



Here MAX CONV1 is set to 2 and the ADC Input Channel Select Sequencing Control Registers (ADCCHSELSEQn) are set to:

	Bits 15−12	Bits 11−8	Bits 7−4	Bits 3−0	
70A3h	V_1	I_3	I_2	I_1	ADCCHSELSEQ1
70A4h	x	x	V_3	V_2	ADCCHSELSEQ2
70A5h	x	x	x	x	ADCCHSELSEQ3
70A6h	x	x	x	x	ADCCHSELSEQ4

Once reset and initialized, SEQ1 waits for a trigger. With the first trigger, three conversions with channel-select values of: CONV00 (I_1), CONV01 (I_2), and CONV02 (I_3) are performed. SEQ1 then waits at current state for another trigger. Twenty-five microseconds later when the second trigger arrives, another three conversions occur, with channel-select values of CONV03 (V_1), CONV04 (V_2), and CONV05 (V_3).

The value of MAX CONV1 is automatically loaded into SEQ CNTRn for both trigger cases. If a different number of conversions are required at the second trigger point, you must (at some appropriate time before the second trigger) change the value of MAX CONV1 through software, otherwise, the current

(originally loaded) value will be reused. This can be done by an ISR that changes the value of MAX CONV1 at the appropriate time. The interrupt operation modes are described in section 1.3.4, *Interrupt Operation During Sequenced Conversions,* on page 1-18.

At the end of the second autoconversion session, the ADC result registers will have the following values:

Buffer Register	ADC conversion result buffer
ADCRESULT0	I_1
ADCRESULT1	I_2
ADCRESULT2	I_3
ADCRESULT3	V_1
ADCRESULT4	V_2
ADCRESULT5	V_3
ADCRESULT6	x
ADCRESULT7	x
ADCRESULT8	x
ADCRESULT9	x
ADCRESULT10	x
ADCRESULT11	x
ADCRESULT12	x
ADCRESULT13	x
ADCRESULT14	x
ADCRESULT15	x

At this point, SEQ1 keeps “waiting” at the current state for another trigger. Now, the user can reset SEQ1 (by software) to state CONV00 and repeat the same trigger1,2 sessions.

1.3.2 Simultaneous Sampling Mode

The ADC has the ability to sample two ADCINxx inputs simultaneously, provided that one input is from the range ADCINA0 − ADCINA7 and the other input is from the range ADCINB0 − ADCINB7. Furthermore, the two inputs must have the same sample-and-hold offset (i.e., ADCINA4 and ADCINB4, but not ADCINA7 and ADCINB6). To put the ADC into simultaneous sampling mode, the SMODE_SEL bit in the ADCTRL3 register must be set. See section 1.2 for details.

1.3.3 Input Trigger Description

Each sequencer has a set of trigger inputs that can be enabled/disabled. The valid input triggers for SEQ1, SEQ2, and cascaded SEQ is as follows:

SEQ1 (sequencer 1)	SEQ2 (sequencer 2)	Cascaded SEQ
Software trigger (software SOC)	Software trigger (software SOC)	Software trigger (software SOC)
Event manager A (EVA SOC)	Event manager B (EVB SOC)	Event manager A (EVA SOC)
External SOC pin		Event manager B (EVB SOC)
		External SOC pin

Note that:

- An SOC trigger can initiate an autoconversion sequence whenever a sequencer is in an idle state. An idle state is either CONV00 prior to receiving a trigger, or any state which the sequencer lands on at the completion of a conversion sequence, i.e., when SEQ CNTRn has reached a count of zero.
- If an SOC trigger occurs while a current conversion sequence is underway, it sets the SOC SEQn bit (which would have been cleared on the commencement of a previous conversion sequence) in the ADCTRL2 register. If yet another SOC trigger occurs, it is lost (i.e., when the SOC SEQn bit is already set (SOC pending), subsequent triggers will be ignored).
- Once triggered, the sequencer cannot be stopped/halted in mid sequence. The program must either wait until an End-of-Sequence (EOS) or initiate a sequencer reset, which brings the sequencer immediately back to the idle start state (CONV00 for SEQ1 and cascaded cases; CONV08 for SEQ2).
- When SEQ1/2 are used in cascaded mode, triggers going to SEQ2 are ignored, while SEQ1 triggers are active. Cascaded mode can be viewed as SEQ1 with 16 states instead of eight.

1.3.4 Interrupt Operation During Sequenced Conversions

The sequencer can generate interrupts under two operating modes. These modes are determined by the Interrupt-Mode-Enable Control bits in ADCTRL2.

A variation of Example 1−2 can be used to show how interrupt mode 1 and mode 2 are useful under different operating conditions.

Case 1: Number of samples in the first and second sequences are not equal

❑ Mode 1 Interrupt operation (i.e., Interrupt request occurs at every EOS)

1) Sequencer is initialized with MAX CONVn = 1 for converting I_1 and I_2

2) At ISR “a”, MAX CONVn is changed to 2 (by software) for converting V_1, V_2, and V_3

3) At ISR “b”, the following events take place :

   1) MAX CONVn is changed to 1 again for converting I_1 and I_2.

   2) Values I_1, I_2, V_1, V_2, and V_3 are read from ADC result registers.

   3) The sequencer is reset.

4) Steps 2 and 3 are repeated. Note that the interrupt flag is set every time SEQ CNTRn reaches zero and both interrupts are recognized.

Case 2: Number of samples in the first and second sequences are equal

❑ Mode 2 Interrupt operation (i.e., Interrupt request occurs at every other EOS)

1) Sequencer is initialized with MAX CONVn = 2 for converting I_1, I_2, and I_3 (or V_1, V_2, and V_3).

2) At ISR “b” and “d”, the following events take place :

   1) Values I_1, I_2, I_3,V_1, V_2, and V_3 are read from ADC result registers.

   2) The sequencer is reset.

3) Step 2 is repeated.

Case 3: Number of samples in the first and second sequences are equal (with dummy read)

- ❑ Mode 2 Interrupt operation (i.e., Interrupt request occurs at every other EOS)

1) Sequencer is initialized with MAX CONVn = 2 for I_1, I_2, x sampling

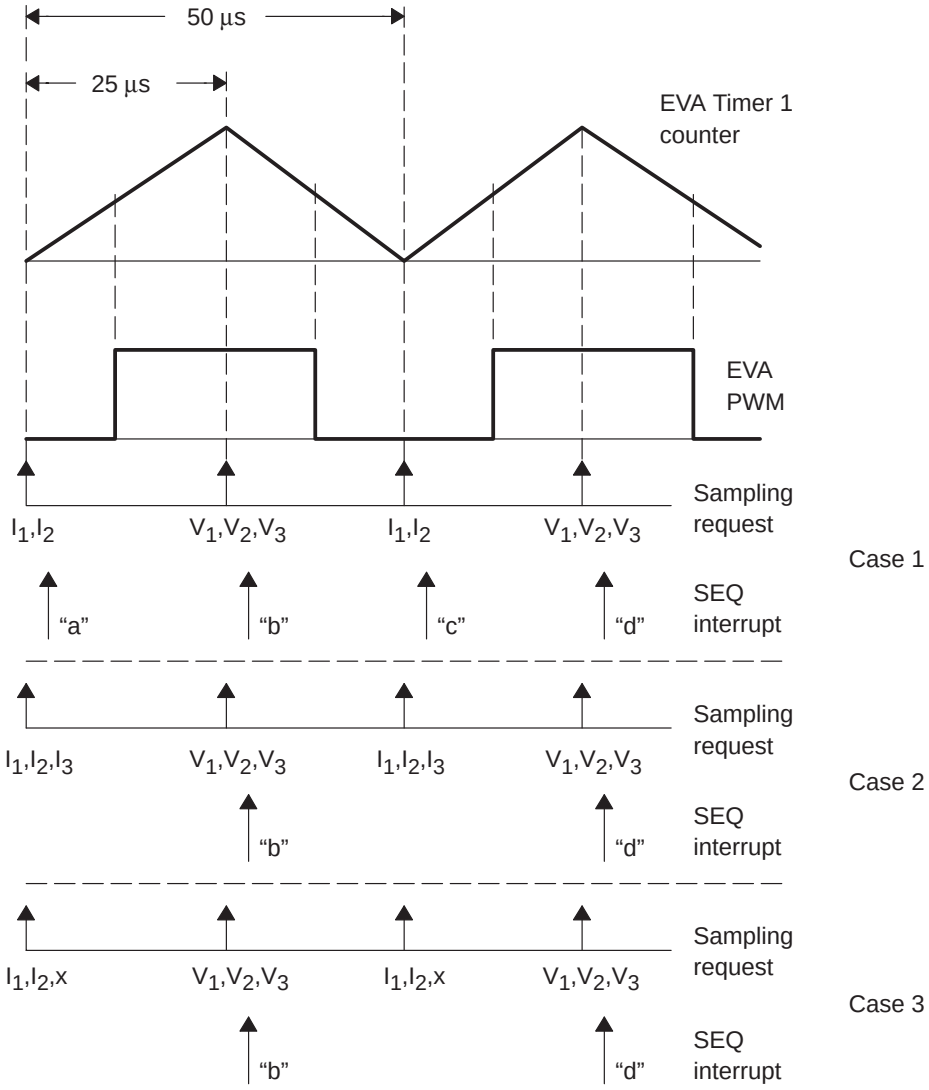
2) At ISR “b” and “d”, the following events take place :

   1) Values I_1, I_2, x,V_1, V_2, and V_3 are read from ADC result registers.

   2) The sequencer is reset.

3) Step 2 is repeated. Note that the third I-sample (x) is a dummy sample, and is not really required. However, to minimize ISR overhead and CPU intervention, advantage is taken of the “every other” Interrupt request feature of Mode 2.

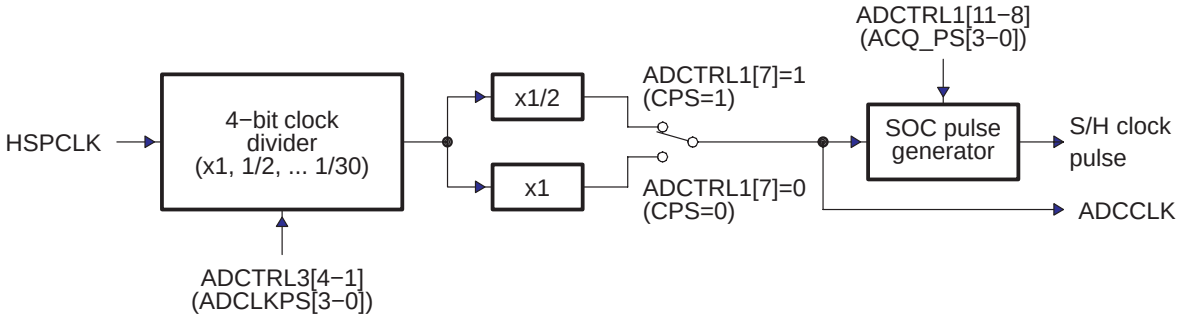
Figure 1−8. Interrupt Operation During Sequenced Conversions



1.4 ADC Clock Prescaler

The peripheral clock HSPCLK is divided down by the ADCCLKPS[3:0] bits of the ADCTRL3 register. An extra divide-by-two is provided via the CPS bit of the ADCTRL1 register. In addition, the ADC can be tailored to accommodate variations in source impedances by widening the sampling/acquisition period. This is controlled by the ACQ_PS3-0 bits in the ADCTRL1 register. These bits do not affect the conversion portion of the S/H and conversion process, but do extend the length of time in which the sampling portion takes by extending the start of the conversion pulse. See Figure 1−9.

Figure 1−9. ADC Core Clock and Sample-and-Hold (S/H)Clock

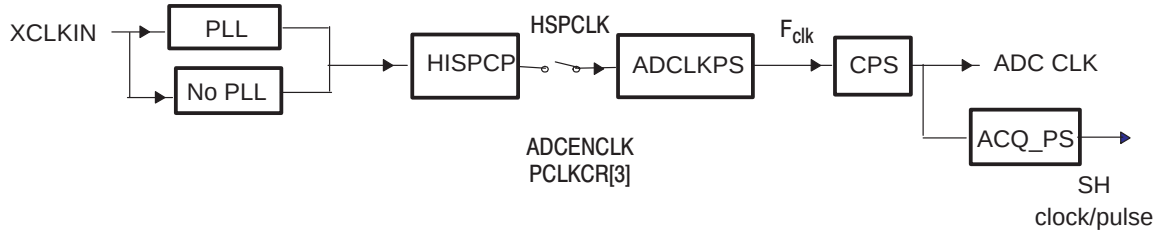


Note: See register bit definition for clock divider ratio and S/H pulse control. S/H pulse width determines the size of acquisition window (the time period for which sampling switch is closed).

1.4.1 ADC-module clock and sample rate

The ADC module has several prescaler stages to generate any desired ADC operating clock speed. The following diagram defines the clock selection stages that feed the ADC module.

Figure 1−10. Clock Chain to the ADC



Example 1−3. Clock Chain to the ADC

XCLKIN	PLLCR[3:0]	HISPCLK	ADCTRL3[4−1]	ADCTRL1[7]	ADC_CLK	ADCTRL1[11−8]	SH Width
	0000b	HSPCP = 0	ADCLKPS = 0	CPS=1		ACQ_PS = 0	
30 MHz	15 MHz	15 MHz	15 MHz	7.5 MHz	7.5 MHz	SH pulse clock	1
	1010b	HSPCP = 3	ADCLKPS = 2	CPS = 1		ACQ_PS = 15	
30 MHz	150 MHz	150/2 X 3 = 25 MHz	25/2 X 2 = 6.25 MHz	6.25/2x1 = 3.125 MHz	3.125 MHz	SH pulse/clock = 16	16

1.5 Low-Power Modes

The ADC supports three separate power sources each controlled by independent bits in the ADCTRL3 register. These three bits combine to make up three power levels: ADC power up, ADC power down, and ADC off.

Table 1−3. Power Options

Power Level	ADCBGRFDN1	ADCBGRFDN0	ADCPWDN
ADC power up	1	1	1
ADC power down	1	1	0
ADC off	0	0	0
Reserved	1	0	X
Reserved	0	1	X

1.6 Power-up Sequence

The ADC resets to the ADC off state. When powering up the ADC, use the following sequence:

1) If external reference is desired, enable this mode using bit 8 in the ADCCTRL3 Register. This mode must be enabled before band gap is powered. This avoids internal reference signals (ADCREFP and ADCREFM) driving external reference sources if present o the board.

2) Power up the reference and bandgap circuits for at least 7 ms before powering up the rest of the ADC analog circuitry.

3) After the ADC has been fully powered up, an additional delay of at least 20 μs is required before performing the first ADC conversion.

When powering down the ADC, all three bits can be cleared simultaneously. The ADC power level must be controlled via software and they are independent of the state of the device power modes.

Note: Follow Power-up Sequence

For reliability and accuracy, the power-up sequence must be followed precisely. See the most recent data sheet for timing data.

1.7 Sequencer Override Feature

Note: Override Feature on F2810/F2811/F2812 Rev C Silicon and Later

The sequencer override feature is not available on Revisions A and B of the F2810/F2812 silicon. It will be available on all subsequent revisions.

In normal operation, sequencers SEQ1, SEQ2 or cascaded SEQ1 help to convert selected ADC channels and store them in the respective ADCRESULTn registers, sequentially. The sequence naturally wraps around at the end of the MAX CONVn setting. With the sequencer override feature, the natural wraparound of the sequencers can be controlled in software. The sequencer override feature was added as bit 5 of the ADC Control Register 1 (ADCCTRL1).

For example, assume the SEQ OVRD bit is 0 and the ADC is in cascaded-sequencer, continuous-conversion mode with MAX CONV1 set to 7. Normally, the sequencer would increment sequentially and update up to ADCRESULT7 register with ADC conversions and wraps around to 0. At the end of the ADCRESULT7 register update, the relevant interrupt flag would be set.

With the new SEQ OVRD bit set to 1, the sequencer updates seven result registers and does *not* wrap around to 0. Instead, the sequencer will increment sequentially and update the ADCRESULT8 register onwards until the ADCRESULT15 register is reached. After updating ADCRESULT15 register, the natural wrap around to 0 will occur. This feature treats the result registers (0−15) like a FIFO for sequential data capture from the ADC. This feature is very helpful to capture ADC data when ADC conversions are done at the maximum data rate.

Recommendations and caution on sequencer override feature:

- After reset, this SEQ OVRD bit will be 0; therefore the sequencer override feature remains disabled.
- When SEQ _OVRD bit is set for all nonzero values of MAX CONVn, the related interrupt flag bit will be set for every MAX CONVn count of result register update.

  For example, if ADCMAXCONV is set to 3, then the interrupt flag for the selected sequencer will be set every three result register updates. The wrap-around always occurs at the end of the sequencer (i.e., after ADCRESULT15 register update in cascaded sequencer mode).
- This will be functional in conversions using SEQ1, SEQ2 and cascaded sequencers using SEQ1.

- It is recommended that this feature not be enabled/controlled dynamically within the program. Always enable this feature during the ADC module initialization.
- In continuous conversion mode, if a sequencer reset is needed:

  Set the CONT RUN bit to 0, wait 2 cycles in the ADC clock domain, then reset the sequencer. CONT RUN can then be set back to 1.
- In continuous-conversion mode with sequencer changes, the ADC channel address uses the preset values in CONVxx registers. If continuous conversions of the same channel are needed then all the CONVxx registers should have the same channel address.

  For example, to get 16 contiguous samples for the ADCINA0 channel using the sequencer override feature, all 16 CONVxx registers should be set to 0x0000.

1.8 Pin Biasing − External Reference

Beginning with silicon revision D† of the F281x, a new feature was added to help improve the gain error of the ADC. By setting bit 8 in ADCTRL3 (EXTREF) the user can feed in controlled voltages on the ADCREFP and ADCREFM pins of the device (Figure 1−11). For best gain performance an ideal delta specified in the data manual is required between the ADCREFP and ADCREFM pins. Typically this number is 1 V. An absolute specification for each of these pins is listed in the data manual as well. Typically these numbers are 2 V ± 5% for the ADCREFP pin and 1 V ± 5% for the ADCREFM pin.

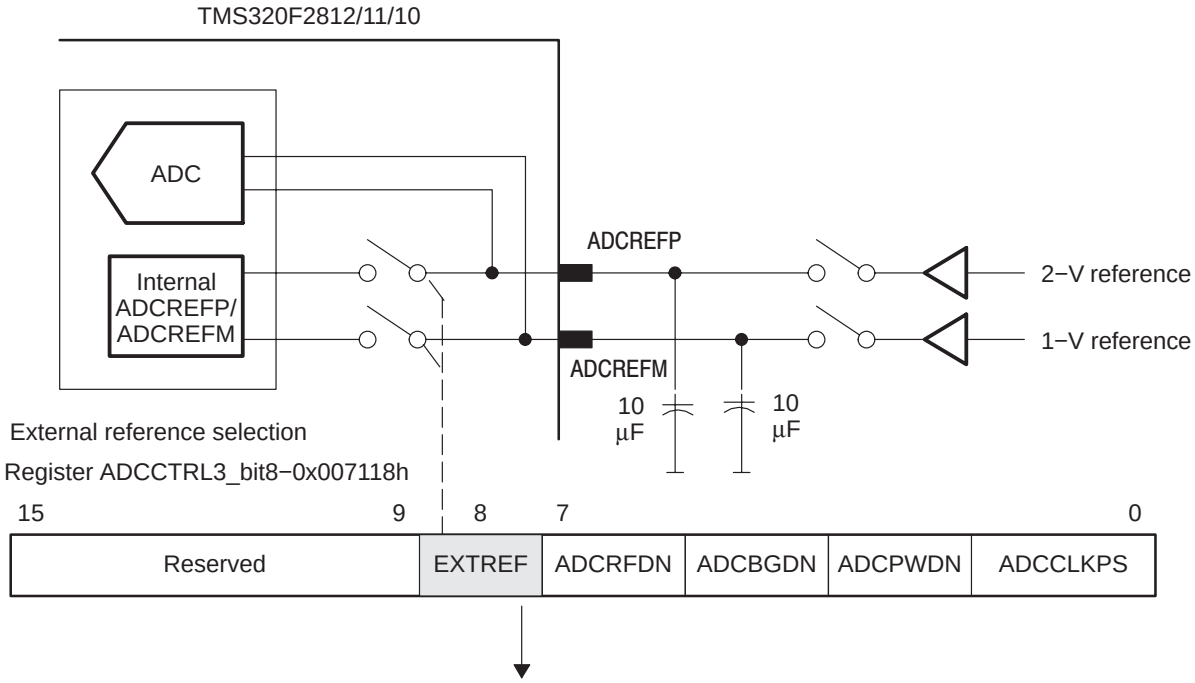
Figure 1−12 provides a reference circuit to get this delta with off-the-shelf components. The circuit uses a precision band gap voltage reference that is then passed through a resistor tree to reduce the voltage properly. The voltages are then buffered before going to the pins of the device. Proper implementation of this circuit is important for two reasons:

1) Stability of ADCREFP and ADCREFM is crucial for best performance of the ADC. While these voltages appear static, they are dynamically used in the ADC system. During every conversion the voltages on these pins are sampled and must settle within a fraction of the ADC clock rate.

2) ADCREFP and ADCREFM sink/source current during ADC operation. If buffers are not used, transients on the line will change the current flow in the resistor tree, changing the intended reference value, resulting in a non-optimal gain error.

† See the *TMS320F2810\F2811\F2812, TMS320C2810\C2811\C2812 DSP Silicon Errata* (literature number SPRZ193 to determine the silicon revision of your device.)

Figure 1−11. ADC External Reference Changes



New bit EXTREF to enable ADCREFM and ADCREFM as input references as follows:

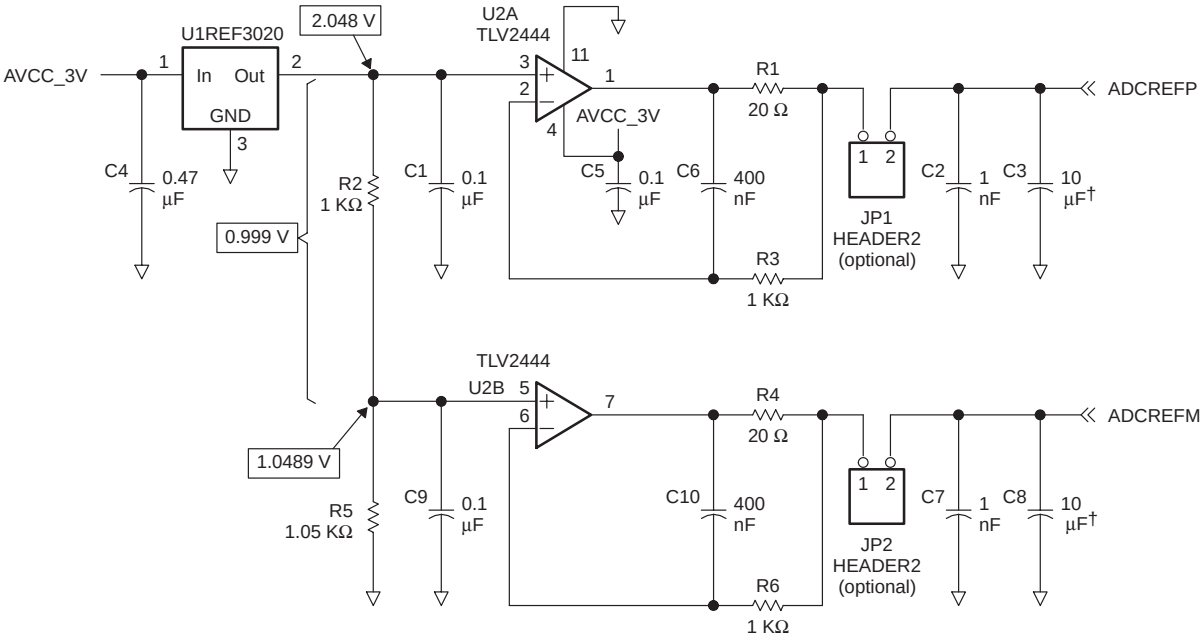
0: Default, ADCREFP (2 V) and ADCREFM (1 V) pins are outputs for internal reference sources

1: Enables ADCREFP (2 V) and ADCREFM (1 V) pins as external reference voltage inputs

Note:

Do not enable internal references when external references are connected to reference pins

Figure 1−12. F281x ADC External Reference Schematics



† Capacitors C3/C8 are required to be low ESR. See the *TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811,TMS320C2812 DSPs Data Manual* (literature number SPRS174) for the recommended part number for your device.

1.8.1 F281x ADC External Reference Considerations

Hardware:

The example circuit provides 2.048 and 1.0489-V references to ADCREFP and ADCREFM pins using single reference source. Reference voltage should be buffered with voltage follower circuits to minimize the loading on these references signals.

The difference in this reference (ADCREFP − ADCREFM) should be 1 V (0.999 V) to keep the gain accuracy to 1%

The overall accuracy of the ADC will depend on the tolerances of the components used, particularly the resistors used to generate 1 V across references ADCREFP and ADCREFM. Care should be taken to place the components close to the ADCREFP, ADCREFM pins in the board layout.

Software:

1) After F281x device initialization, enable clock to ADC module.

2) Set the new register bit for ADCREFP, ADCREFM to be inputs. Since ADCREFP/M comes as output by default, enabling the direction before ADC power up will avoid conflicts.

3) Enable ADC power up sequence for bandgap reference as recommended.

1.8.2 Initialization Software// Adc initialization −

1.8.2.1 C−Example code using DSP281x header file format

```
// Adc initialization – C–Example code using DSP281x header file format.
void InitAdc(void)
{
AdcRegs.ADCTRL3.all = 0x0100;             // Set bit 8 high to enable external
                                          // ADCREFP/REFM sourcing
                                          // See Note 1.
asm(" rpt #10 || nop");                   // Time to enable of external ref
AdcRegs.ADCTRL3.bit.ADCBGRFDN = 0x3;      // Power up bandgap/reference circuits
DELAY_US(ADC_usDELAY);                    // Delay before powering up ADC
                                          // See Note 2.
AdcRegs.ADCTRL3.bit.ADCPWDN = 1;          // Power up rest of ADC
DELAY_US(ADC_usDELAY2);                   // Delay after powering up ADC
}
```

Notes: 1) The EXTREF bit field is not defined in the F281x header files. Modification to this bit can be done as above with the "all" suffix to the ADCTRL3 registers. For explicit control, make the appropriate changes to the ADC section of the header files.

2) ADC_usDELAY is an assembly macro in DSP281x header examples.

Chapter 2

ADC Registers

This chapter contains the ADC registers and bit definitions, with the registers grouped by function.

Topic **Page**

2.1 ADC Control Registers

Figure 2−1. ADC Control Register 1 (ADCTRL1) (Address Offset 00h)

15	14	13	12	11	10	9	8
Reserved	RESET	SUSMOD1	SUSMOD0	ACQ PS3	ACQ PS2	ACQ PS1	ACQ PS0
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3			0
CPS	CONT RUN	SEQ1 OVRD	SEQ CASC	Reserved			
R/W-0	R/W-0	R/W-0	R/W-0	R-0			

Note: R = Read, W = Write, -n = value after reset

Bit(s)	Name	Description
15	Reserved	Reads return a zero. Writes have no effect.

14	RESET	ADC module software reset. This bit causes a master reset on the entire ADC module. All register bits and sequencer state machines are reset to the initial state as occurs when the device reset pin is pulled low (or after a power-on reset).
		This is a one-time-effect bit, meaning this bit is self-cleared immediately after it is set to 1. Read of this bit always returns a 0. Also, the reset of ADC has a latency of two clock cycles (that is, other ADC control register bits should not be modified until two cycles after the instruction that resets the ADC.
		0 No effect
		1 Resets entire ADC module (bit is then set back to 0 by ADC logic)
		Note: The ADC module is reset during a system reset. If an ADC module reset is desired at any other time, you can do so by writing a 1 to this bit. After two ADC clock domain cycles, you can then write the appropriate values to the ADCCTRL1 register bits. The example below assumes 150−MHz DSP Clock and 25−MHz ADCCLK
		Assembly code:
		`MOV ADCTRL1, #01xxxxxxxxxxxxxxb; Resets the ADC (RESET = 1)`
		`RPT #10 ||`
		`NOP ; Provides the required delay`
		`; between writes to ADCTRL1`
		`MOV ADCTRL1, #00xxxxxxxxxxxxxxb ; Configures ADCTRL1 to user-`
		`; desired value`
		Note that the second MOV is not required if the default configuration is sufficient.

Figure 2−1.ADC Control Register 1 (ADCTRL1) (Address Offset 00h) (Continued)

Bit(s)	Name	Description
13−12	SUSMOD1−SUSMOD0	Emulation-suspend mode. These bits determine what occurs when an emulation-suspend occurs (due to the debugger hitting a breakpoint, for example). 0 0 Mode 0. Emulation suspend is ignored. 0 1 Mode 1. Sequencer and other wrapper logic stops after current **sequence** is complete, final result is latched, and state machine is updated. 1 0 Mode 2. Sequencer and other wrapper logic stops after current **conversion** is complete, result is latched, and state machine is updated. 1 1 Mode 3. Sequencer and other wrapper logic stops immediately on emulation suspend.
11−8	ACQ_PS3 − ACQ_PS0	Acquisition window size. This bit field controls the width of SOC pulse, which, in turn, determines for what time duration the sampling switch is closed. The width of SOC pulse is ADCTRL1[11:8] + 1 times the ADCLK period.
7	CPS	Core clock prescaler. The prescaler is applied to divided device peripheral clock, HSPCLK. 0 ADCCLK = F_{clk} /1 1 ADCCLK = F_{clk}/2 **Note:** CLK = Prescaled HSPCLK (ADCCLKPS3−0)
6	CONT RUN	Continuous run. This bit determines whether the sequencer operates in continuous conversion mode or start-stop mode. This bit can be written while a current conversion sequence is active. This bit will take effect at the end of the current conversion sequence; i.e., software can set/clear this bit until EOS has occurred, for valid action to be taken. In the continuous conversion mode, there is no need to reset the sequencer; however, the sequencer must be reset in the start-stop mode to put the converter in state CONV00. 0 Start-stop mode. Sequencer stops after reaching EOS. On the next SOC, the sequencer starts from the state where it ended unless a sequencer reset is performed. 1 Continuous conversion mode. After reaching EOS, the sequencer starts all over again from state CONV00 (for SEQ1 and cascaded) or CONV08 (for SEQ2).

Figure 2−1.ADC Control Register 1 (ADCTRL1) (Address Offset 00h) (Continued)

Bit(s)	Name	Description
5	SEQ OVRD	Sequencer override. Provides additional sequencer flexibility in continuous run mode by overriding the wrapping around at the end of conversions set by MAX CONVn. This bit is not available in revisions A and B of the silicon; in those revisions, it is a reserved read-only bit. 0 Disabled − Allows the sequencer to wrap around at the end of conversions set by MAX CONVn. 1 Enabled − Overrides the sequencer from wrapping around at the end of conversions set by MAX CONVn. Wraparound occurs only at the end of the sequencer.
4	SEQ CASC	Cascaded sequencer operation. This bit determines whether SEQ1 and SEQ2 operate as two 8-state sequencers or as a single 16-state sequencer (SEQ). 0 Dual-sequencer mode. SEQ1 and SEQ2 operate as two 8-state sequencers. 1 Cascaded mode. SEQ1 and SEQ2 operate as a single 16-state sequencer (SEQ).
3−0	Reserved	Reads return zero. Writes have no effect.

Figure 2−2. ADC Control Register 2 (ADCTRL2) (Address Offset 01h)

15	14	13	12	11	10	9	8
EVB SOC SEQ	RST SEQ1	SOC SEQ1	Reserved	INT ENA SEQ1	INT MOD SEQ1	Reserved	EVA SOC SEQ1
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0
7	6	5	4	3	2	1	0
EXT SOC SEQ1	RST SEQ2	SOC SEQ2	Reserved	INT ENA SEQ2	INT MOD SEQ2	Reserved	EVB SOC SEQ2
R/W-0	R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0	R/W-0

Note: R = Read access, W = Write access, S = Set only, C = Clear, -0 = value after reset

Bit(s)	Name	Description
15	EVB SOC SEQ	EVB SOC enable for cascaded sequencer *(Note: This bit is active only in cascaded mode.)* 0 No action. 1 Setting this bit allows the cascaded sequencer to be started by an Event Manager B signal. The Event Manager can be programmed to start a conversion on various events.

Figure 2−2.ADC Control Register 2 (ADCTRL2) (Address Offset 01h (Continued)

Bit(s)	Name	Description
14	RST SEQ1	Reset sequencer1 Writing a 1 to this bit resets the sequencer immediately to an initial “pretriggered” state, i.e., waiting for a trigger at CONV00. A currently active conversion sequence will be aborted. 0 No action. 1 Immediately reset sequencer to state CONV00
13	SOC SEQ1	Start-of-conversion (SOC) trigger for Sequencer 1 (SEQ1). This bit can be set by the following triggers: − S/W – Software writing a 1 to this bit − EVA – Event Manager A − EVB – Event Manager B (only in cascaded mode) − EXT – External pin (i.e., the ADCSOC pin) When a trigger occurs, there are three possibilities: **Case 1:** SEQ1 idle and SOC bit clear SEQ1 starts immediately (under arbiter control). This bit is set and cleared, allowing for any “pending” trigger requests. **Case 2:** SEQ1 busy and SOC bit clear Bit is set signifying a trigger request is pending. When SEQ1 finally starts after completing current conversion, this bit is cleared. **Case 3:** SEQ1 busy and SOC bit set Any trigger occurring in this case is ignored (lost). 0 Clears a pending SOC trigger. Note: If the sequencer has already started, this bit is automatically cleared, and hence, writing a zero has no effect; i.e., an already started sequencer cannot be stopped by clearing this bit. 1 Software trigger − Start SEQ1 from currently stopped position (i.e., Idle mode)

Note: The RST SEQ1 (ADCTRL2.14) and the SOC SEQ1 (ADCTRL2.13) bits should not be set in the same instruction. This resets the sequencer, but does not start the sequence. The correct sequence of operation is to set the RST SEQ1 bit first, and the SOC SEQ1 bit in the following instruction. This ensures that the sequencer is reset and a new sequence started. This sequence applies to the RST SEQ2 (ADCTRL2.6) and SOC SEQ2 (ADCTRL2.5) bits also.

12	Reserved	Reads return a zero. Writes have no effect.
11	INT ENA SEQ1	SEQ1 interrupt enable. This bit enables the interrupt request to CPU by INT SEQ1. 0 Interrupt request by INT SEQ1 is disabled. 1 Interrupt request by INT SEQ1 is enabled.

Figure 2−2.ADC Control Register 2 (ADCTRL2) (Address Offset 01h (Continued)

Bit(s)	Name	Description
10	INT MOD SEQ1	SEQ1 interrupt mode. This bit selects SEQ1 interrupt mode. It affects the setting of INT SEQ1 at the end of the SEQ1 conversion sequence. 0 INT SEQ1 is set at the end of every SEQ1 sequence. 1 INT SEQ1 is set at the end of every other SEQ1 sequence.
9	Reserved	Reads return a zero. Writes have no effect.
8	EVA SOC SEQ1	Event Manager A SOC mask bit for SEQ1 0 SEQ1 cannot be started by EVA trigger. 1 Allows SEQ1/SEQ to be started by Event Manager A trigger. The Event Manager can be programmed to start a conversion on various events.
7	EXT SOC SEQ1	External signal start-of-conversion bit for SEQ1 0 No action 1 Setting this bit enables an ADC autoconversion sequence to be started by a signal from the ADCSOC device pin.
6	RST SEQ2	Reset SEQ2 0 No action 1 Immediately resets SEQ2 to an initial “pretriggered” state, i.e., waiting for a trigger at CONV08. A currently active conversion sequence will be aborted.
5	SOC SEQ2	Start of conversion trigger for sequencer 2 (SEQ2). *(Only applicable in dual-sequencer mode; ignored in cascaded mode.)* This bit can be set by the following triggers: − S/W – Software writing of 1 to this bit − EVB – Event Manager B When a trigger occurs, there are three possibilities: **Case 1:** SEQ2 idle and SOC bit clear SEQ2 starts immediately (under arbiter control) and the bit is cleared, allowing for any pending trigger requests. **Case 2:** SEQ2 busy and SOC bit clear Bit is set signifying a trigger request is pending. When SEQ2 finally starts after completing current conversion, this bit will be cleared. **Case 3:** SEQ2 busy and SOC bit set Any trigger occurring in this case will be ignored (lost).

Figure 2−2.ADC Control Register 2 (ADCTRL2) (Address Offset 01h (Continued)

Bit(s)	Name	Description
		0 Clears a Pending SOC trigger **Note:** If the sequencer has already started, this bit will automatically be cleared, and writing a zero has no effect; i.e., an already started sequencer cannot be stopped by clearing this bit. 1 Starts SEQ2 from currently stopped position (i.e., Idle mode)
4	Reserved	Reads return a zero. Writes have no effect.
3	INT ENA SEQ2	SEQ2 interrupt enable. This bit enables or disables an interrupt request to the CPU by INT SEQ2. 0 Interrupt request by INT SEQ2 is disabled. 1 Interrupt request by INT SEQ2 is enabled.
2	INT MOD SEQ2	SEQ2 interrupt mode. This bit selects SEQ2 interrupt mode. It affects the setting of INT SEQ2 at the end of the SEQ2 conversion sequence. 0 INT SEQ2 is set at the end of every SEQ2 sequence. 1 INT SEQ2 is set at the end of every other SEQ2 sequence.
1	Reserved	Reads return a zero. Writes have no effect.
0	EVB SOC SEQ2	Event Manager B SOC mask bit for SEQ2. 0 SEQ2 cannot be started by EVB trigger. 1 Allows SEQ2 to be started by Event Manager B trigger. The Event Manager can be programmed to start a conversion on various events.

Figure 2−3. ADC Control Register 3 (ADCTRL3)(Address Offset 18h)

15	9	8
Reserved		EXTREF
R-0		R/W-0

7	6	5	4	1	0
ADCBGRFDN1	ADCBGRFDN0	ADCPWDN	ADCCLKPS[3:0]		SMODE_SEL
R/W-0	R/W-0	R/W-0	R/W-0		R/W-0

Bit(s)	Name	Description
15−9	Reserved	Reads return a zero. Writes have no effect.

Figure 2−3.ADC Control Register 3 (ADCTRL3)(Address Offset 18h) (Continued)

Bit(s)	Name	Description
8	EXTREF	Enable pins ADCREFM and ADCREFP as input references 0 ADCREFP(2V) and ADCREFM(1V) pins are outputs for internal reference sources. 1 ADCREFP(2V) and ADCREFM(1V) pins are inputs for external reference voltages.
7−6	ADCBGRFDN[1:0]	ADC bandgap and reference power down. These bits control the power up and power down of the bandgap and reference circuitry inside the analog core. See Section 1.6 for power-up sequence requirements. 00 The bandgap and reference circuitry is powered down. 11 The bandgap and reference circuitry is powered up.
5	ADCPWDN	ADC power down. This bit controls the power up and power down of all the analog circuitry inside the analog core except the bandgap and reference circuitry. See Section 1.6 for power-up sequence requirements. 0 All analog circuitry inside the core except the bandgap and reference circuitry is powered down. 1 The analog circuitry inside the core is powered up.

Figure 2−3.ADC Control Register 3 (ADCTRL3)(Address Offset 18h) (Continued)

Bit(s)	Name	Description
4−1	ADCCLKPS [3:0]	Core clock divider. 28x peripheral clock, HSPCLK, is divided by 2*ADCCLKPS[3−0], except when ADCCLKPS[3−0] is 0000, in which case HSPCLK is directly passed on. The divided clock is further divided by ADCTRL1[7]+1 to generate the core clock, ADCLK.

ADCCLKPS [3:0]	Core Clock Divider	ADCLK
0000	0	HSPCLK/(ADCTRL1[7] + 1)
0001	1	HSPCLK/[2*(ADCTRL1[7] + 1)]
0010	2	HSPCLK/[4*(ADCTRL1[7] + 1)]
0011	3	HSPCLK/[6*(ADCTRL1[7] + 1)]
0100	4	HSPCLK/[8*(ADCTRL1[7] + 1)]
0101	5	HSPCLK/[10*(ADCTRL1[7] + 1)]
0110	6	HSPCLK/[12*(ADCTRL1[7] + 1)]
0111	7	HSPCLK/[14*(ADCTRL1[7] + 1)]
1000	8	HSPCLK/[16*(ADCTRL1[7] + 1)]
1001	9	HSPCLK/[18*(ADCTRL1[7] + 1)]
1010	10	HSPCLK/[20*(ADCTRL1[7] + 1)]
1011	11	HSPCLK/[22*(ADCTRL1[7] + 1)]
1100	12	HSPCLK/[24*(ADCTRL1[7] + 1)]
1101	13	HSPCLK/[26*(ADCTRL1[7] + 1)]
1110	14	HSPCLK/[28*(ADCTRL1[7] + 1)]
1111	15	HSPCLK/[30*(ADCTRL1[7] + 1)]

0	SMODE SEL	Sampling mode select. This bit selects either sequential or simultaneous sampling mode.

0 Sequential sampling mode is selected.

1 Simultaneous sampling mode is selected.

2.2 Maximum Conversion Channels Register (ADCMAXCONV)

Figure 2−4. Maximum Conversion Channels Register (ADCMAXCONV) (Offset Address 02h)

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved	MAX CONV2_2	MAX CONV2_1	MAX CONV2_0	MAX CONV1_3	MAX CONV1_2	MAX CONV1_1	MAX CONV1_0
R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, x = undefined, -0 = value after reset

Bit(s)	Name	Description
15−7	Reserved	Reads return a zero. Writes have no effect.
6−0	MAX CONVn	MAX CONVn bit field defines the maximum number of conversions executed in an autoconversion session. The bit fields and their operation vary according to the sequencer modes (dual/cascaded). − For SEQ1 operation, bits MAX CONV1_2 – 0 are used. − For SEQ2 operation, bits MAX CONV2_2 – 0 are used. − For SEQ operation, bits MAX CONV1_3 – 0 are used. An autoconversion session always starts with the initial state and continues sequentially until the end state if allowed. The result buffer is filled in a sequential order. Any number of conversions between 1 and (MAX CONVn +1) can be programmed for a session.

Example 2−1. ADCMAXCONV Register Bit Programming

If only five conversions are required, then MAX CONVn is set to four.

Case 1: Dual mode SEQ1 and cascaded mode
Sequencer goes from CONV00 to CONV04, and the five conversion results are stored in the registers Result 00 to Result 04 of the Conversion Result Buffer.

Case 2: Dual mode SEQ2
Sequencer goes from CONV08 to CONV12, and the five conversion results are stored in the registers Result 08 to Result 12 of the Conversion Result Buffer.

MAX CONV1 Value >7 for Dual-Sequencer Mode

If a value for MAX CONV1, which is greater than 7, is chosen for the dual-sequencer mode (i.e., two separate 8-state sequencers), then SEQ CNTRn

will continue counting past seven, causing the sequencer to wrap around to CONV00 and continue counting.

Table 2−1. Bit Selections for MAX CONV1 for Various Number of Conversions

ADCMAXCONV[3−0]	Number of conversions
0000	1
0001	2
0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
1001	10
1010	11
1011	12
1100	13
1101	14
1110	15
1111	16

2.3 Autosequence Status Register (ADCASEQSR)

Figure 2−5. Autosequence Status Register (ADCASEQSR) (Address Offset 07h)

15	14	13	12	11	10	9	8
Reserved				SEQ CNTR 3	SEQ CNTR 2	SEQ CNTR 1	SEQ CNTR 0
R-0				R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
Reserved	SEQ2 STATE2	SEQ2 STATE1	SEQ2 STATE0	SEQ1 STATE3	SEQ1 STATE2	SEQ1 STATE1	SEQ1 STATE0
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Note: R = Read access, x = undefined, -0 = value after reset

Bit(s)	Name	Description
15−12	Reserved	Reads return a zero. Writes have no effect.
11−8	SEQ CNTR 3−0	Sequencing counter status bits. The SEQ CNTRn 4-bit status field is used by SEQ1, SEQ2, and the cascaded sequencer. SEQ2 is irrelevant in cascaded mode. The Sequencer Counter bit field, SEQ CNTR(3−0), is initialized to the value in MAX CONV at the start of a conversion sequence. After each conversion (or a pair of conversions in simultaneous sampling mode) in an auto conversion sequence, the Sequencer Counter decreases by 1. The SEQ CNTRn bits can be read at any time during the countdown process to check status of the sequencer. This value, together with the SEQ1 and SEQ2 busy bits, uniquely identifies the progress or state of the active sequencer at any point in time.

SEQ CNTRn (read only)	Number of conversions remaining
0000	1 or 0, depending on the busy bit
0001	2
0010	3
0011	4
0100	5
0101	6
0110	7
0111	8
1000	9
1001	10
1010	11
1011	12
1100	13
1101	14
1110	15
1111	16

Figure 2−5.Autosequence Status Register (ADCASEQSR) (Address Offset 07h) (Continued)

Bit(s)	Name	Description
7	Reserved	Reads return a zero. Writes have no effect.
6−0	SEQ2 STATE2 – SEQ2 STATE0 and SEQ1 STATE3 – SEQ1 STATE0	SEQ2 STATE2−0 and SEQ1 STATE3−0 bit fields are the pointers of SEQ2 and SEQ1, respectively. These bits are reserved for TI testing and should not be used in customer applications.

2.4 ADC Status and Flag Register (ADCST)

Figure 2−6. ADC Status and Flag Register (ADCST) (Address Offset 19h)

15							8
Reserved							
R-0							

7	6	5	4	3	2	1	0
EOS BUF2	EOS BUF1	INT SEQ2 CLR	INT SEQ1 CLR	SEQ2 BSY	SEQ1 BSY	INT SEQ2	INT SEQ1
R-0	R-0	R/W-0	R/W-0	R-0	R-0	R-0	R-0

This register is a dedicated status and flag register. The bits in this register are either read-only status or flag bits, or read-return-zero condition clearing bits.

Bit(s)	Name	Description
15−8	Reserved	Reads return a zero. Writes have no effect.
7	EOS BUF2	End of sequence buffer bit for SEQ2. This bit is not used and remains as zero in interrupt mode 0, i.e. when ADCTRL2[2]=0. In interrupt mode 1, i.e. when ADCTRL2[2]=1, it toggles on every end of sequence of SEQ2. This bit is cleared on device reset and is not affected by sequencer reset or clearing of the corresponding interrupt flag.
6	EOS BUF1	End of sequence buffer bit for SEQ1. This bit is not used and remains as zero in interrupt mode 0, i.e. when ADCTRL2[10]=0. In interrupt mode 1, i.e. when ADCTRL2[10]=1, it toggles on every end of sequence of SEQ1. This bit is cleared on device reset and is not affected by sequencer reset or clearing of the corresponding interrupt flag.
5	INT SEQ2 CLR SEQ2	Interrupt clear bit. Read of this bit always returns 0. The clear action is a one-shot event following a write of 1 to this bit. 0 Writing a zero to this bit has no effect. 1 Writing a 1 to this bit clears the SEQ2 interrupt flag bit, INT SEQ2.
4	INT SEQ1 CLR SEQ1	Interrupt clear bit. Read of this bit always returns 0. The clear action is a one-shot event following a write of 1 to this bit. 0 Writing a zero to this bit has no effect. 1 Writing a 1 to this bit clears the SEQ1 interrupt flag bit, INT SEQ1.
3	SEQ2 BSY	SEQ2 busy status bit. 0 SEQ2 is in idle, waiting for trigger. 1 SEQ2 is in progress. Write to this bit has no effect.
2	SEQ1 BSY	SEQ1 busy status bit. Write to this bit has no effect. 0 SEQ1 is in idle, waiting for trigger. 1 SEQ1 is in progress.

Figure 2−6.ADC Status and Flag Register (ADCST) (Address Offset 19h) (Continued)

Bit(s)	Name	Description
1	INT SEQ2	SEQ2 interrupt flag bit. Write to this bit has no effect. In interrupt mode 0, i.e. when ADCTRL2[2]=0, this bit is set on every end of sequence of Seq 2. In interrupt mode 1, i.e., when ADCTRL2[2]=1, this bit is set on an end of sequence of Seq 2 if EOS_BUF2 is set. 0 No SEQ2 interrupt event. 1 SEQ2 interrupt event occurred.
0	INT SEQ1	SEQ1 interrupt flag bit. Write to this bit has no effect. In interrupt mode 0, i.e. when ADCTRL2[10]=0, this bit is set on every end of sequence of Seq 1. In interrupt mode 1, i.e., when ADCTRL2[10]=1, this bit is set on an end of sequence of Seq 1 if EOS_BUF1 is set. 0 No SEQ1 interrupt event. 1 SEQ1 interrupt event occurred.

2.5 ADC Input Channel Select Sequencing Control Registers

Figure 2−7. ADC Input Channel Select Sequencing Control Registers (ADCCHSELSEQ1) (Address Offset 03h)

15 12	11 8	7 4	3 0
CONV03	CONV02	CONV01	CONV00
R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, -0 = value after reset

Figure 2−8. ADC Input Channel Select Sequencing Control Registers (ADCCHSELSEQ2) (Address Offset 04h)

15 12	11 8	7 4	3 0
CONV07	CONV06	CONV05	CONV04
R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, -0 = value after reset

Figure 2−9. ADC Input Channel Select Sequencing Control Registers (ADCCHSELSEQ3) (Address Offset 05h)

15 12	11 8	7 4	3 0
CONV11	CONV10	CONV09	CONV08
R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, -0 = value after reset

Figure 2−10. ADC Input Channel Select Sequencing Control Registers (ADCCHSELSEQ4) (Address Offset 06h)

15 12	11 8	7 4	3 0
CONV15	CONV14	CONV13	CONV12
R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, -0 = value after reset

Each of the 4-bit fields, CONVnn, selects one of the 16 MUXed analog input ADC channels for an autosequenced conversion.

Table 2−2. CONVnn Bit Values and the ADC Input Channels Selected

CONVnn Value	ADC Input Channel Selected
0000	ADCINA0
0001	ADCINA1
0010	ADCINA2
0011	ADCINA3
0100	ADCINA4
0101	ADCINA5
0110	ADCINA6
0111	ADCINA7
1000	ADCINB0
1001	ADCINB1
1010	ADCINB2
1011	ADCINB3
1100	ADCINB4
1101	ADCINB5
1110	ADCINB6
1111	ADCINB7

2.6 ADC Conversion Result Buffer Registers (ADCRESULTn)

In the cascaded sequencer mode, registers ADCRESULT8 through ADCRESULT15 holds the results of the ninth through sixteenth conversions. The ADCRESULTn registers are left justified.

Figure 2−11. ADC Conversion Result Buffer Registers (ADCRESULTn) − (Address Offset 08h − 17h)

15	14	13	12	11	10	9	8
D11	D10	D9	D8	D7	D6	D5	D4
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
D3	D2	D1	D0	Reserved	Reserved	Reserved	Reserved
R-0	R-0	R-0	R-0	R-0	R-0	R-0	R-0

Appendix A

Revision History

This document was revised to SPRU060D from SPRU060C. The scope of the revisions was limited to technical changes as described in Section A.1. This appendix lists only revisions made in the most recent version.

A.1 Changes Made in This Revision

The following changes were made in this revision:

Page	Additions/Modifications/Deletions
1-28	Added section 1.8 on pin biasing
1-27	Added a bullet to Section 1.7
1.8	Modified the first paragraph of Section 1.8, Pin Biasing − External Reference
1-28	Modified the second numbered reason under Section 1.8
1-29	Modified Figure 1−11
1-30	Added a note to Figure 1−12
1-31	Added a note to section 1.8.2.1 on modification of header files
2-2	Changed RPT #12 to RPT #10 and changed note in Figure 2−1 on bit 14 description.
2-3	Changed description of bit 7 in Figure 2−1

TMS320x281x DSP Multichannel Buffered Serial Port (McBSP) Reference Guide

Literature Number: SPRU061B
May 2003 − Revised November 2004

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third−party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products & application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This guide describes how Multichannel Buffered Serial Port (McBSP) works in the TMS320x281x devices.

Related Documentation From Texas Instruments

The following books describe the TMS320x281x and related support tools that are available on the TI website.

TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, and TMS320C2812 Digital Signal Processors (literature number SPRS174) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320R2811 and TMS320R2812 Digital Signal Processors (literature number SPRS257) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C28x DSP CPU and Instruction Set Reference Guide (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x™ fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320x281x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU060) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x281x Boot ROM Reference Guide (literature number SPRU095) describes the purpose and features of the bootloader (factory-pro-

grammed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide (literature number SPRU074) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

TMS320x281x Event Manager (EV) Reference Guide (literature number SPRU065) describes the EV modules that provide a broad range of functions and features that are particularly useful in motion control and motor control applications. The EV modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits.

TMS320x281x External Interface (XINTF) Reference Guide (literature number SPRU067) describes the external interface (XINTF) of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Peripheral Reference Guide (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide (literature number SPRU051) describes the SCI that is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

TMS320x281x, 280x Serial Peripheral Interface (SPI) Reference Guide (literature number SPRU059) describes the SPI − a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit−transfer rate. The SPI is used for communications between the DSP controller and external peripherals or another controller.

TMS320x281x System Control and Interrupts Reference Guide (literature number SPRU078) describes the various interrupts and system control features of the 281x digital signal processors (DSPs).

The TMS320C28x Instruction Set Simulator Technical Overview (literature number SPRU608) describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU625) describes development using DSP/BIOS.

3.3 V DSP for Digital Motor Control Application Report (literature number SPRA550). New generations of motor control digital signal processors (DSPs) lower their supply voltages from 5 V to 3.3 V to offer higher performance at lower cost. Replacing traditional 5-V digital control circuitry by 3.3-V designs introduce no additional system cost and no significant complication in interfacing with TTL and CMOS compatible components, as well as with mixed voltage ICs such as power transistor gate drivers. Just like 5-V based designs, good engineering practice should be exercised to minimize noise and EMI effects by proper component layout and PCB design when 3.3-V DSP, ADC, and digital circuitry are used in a mixed signal environment, with high and low voltage analog and switching signals, such as a motor control system. In addition, software techniques such as Random PWM method can be used by special features of the Texas Instruments (TI) TMS320x24xx DSP controllers to significantly reduce noise effects caused by EMI radiation.

This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications.The application report first describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V − 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system's noise and EMI effects are summarized in the last section.

Thermo-Electric Cooler Control Using a TMS320F2812 DSP & DRV592 Power Amplifier Application Note (literature number SPRA873). This application report presents a thermoelectric cooler system consisting of a Texas Instruments TMS320F2812 digital signal processor (DSP) and DRV592 power amplifier. The DSP implements a digital proportional-integral-derivative feedback controller using an integrated

12-bit analog-to-digital converter to read the thermistor, and direct output of pulse-width-modulated waveforms to the H-bridge DRV592 power amplifier. The system presented provides up to 6.1 watts of heating or cooling to the laser mount, although the DRV592 amplifier is actually capable of delivering up to 15 watts when configured appropriately. The closed-loop TEC system is seen to achieve ±0.0006°C temperature accuracy, depending on the needed operating temperature range, with a step response settling time of 14 to 16 seconds. A complete description of the experimental system, along with software and software operating instructions, are provided.

Running an Application from Internal Flash Memory on the TMS320F281x DSP Application Report (literature number SPRA958). Several special requirements exist for running an application from on-chip flash memory on the TMS320F28x DSP. These requirements generally do not manifest themselves during development in RAM since the Code Composer Studio™ debugger can mask problems associated with initialized sections and how they are linked to memory. This application report covers the requirements needed to properly configure application software for execution from on-chip flash memory. Requirements for both DSP/BIOS™ and non-DSP/BIOS projects are presented. Some performance considerations and techniques are also discussed. Example code projects are included that run from on-chip flash on the eZdsp™ F2812 development board (or alternately any F2812, F2811, or F2810 DSP board). Code examples that run from internal RAM are also provided for completeness. These code examples provide a starting point for code development, if desired.

Trademarks

Code Composer Studio and C28x are trademarks of Texas Instruments.

Contents

Figures

Tables

Chapter 1

Overview

The multichannel buffered serial port (McBSP) provides a direct serial interface between a digital signal processor (DSP) and other devices in a system.

This reference guide is applicable for the McBSP found on the TMS320x281x family of processors. This includes all Flash-based, ROM-based, and RAM-based devices within the 281x family.

If you have used a McBSP in other TI DSPs, note that there are differences in this implementation, which are outlined in Table 1−1.

Topic **Page**

1.1 Introduction to the McBSP

The McBSP peripheral provides an interface between a 28x device and McBSP-compatible devices such as the VBAP, AIC, Combo Codecs. The external components attached to the McBSP can synchronously transmit/receive 8/16/32-bit serial data.

1.1.1 Key Features of the McBSP

The McBSP provides:

- ❏ Full-duplex communication
- ❏ Double-buffered transmission and triple-buffered reception, which allow a continuous data stream
- ❏ Independent clocking and framing for reception and for transmission
- ❏ 128 channels for transmission and for reception
- ❏ Multichannel selection modes that enable you to allow or block transfers in each of the channels
- ❏ DMA replaced with two 16-level, 32-bit FIFOs
- ❏ Support for A-bis mode
- ❏ Direct interface to industry-standard codecs, analog interface chips (AICs), and other serially connected A/D and D/A devices
- ❏ Support for external generation of clock signals and frame-synchronization (frame-sync) signals
- ❏ A programmable sample rate generator for internal generation and control of frame-sync signals
- ❏ Programmable internal clock and frame generation
- ❏ Programmable polarity for frame-synchronization and data clocks
- ❏ Support for SPI devices

- Support fractional T1/E1. Direct interface to:
  - T1/E1 framers
  - MVIP switching compatible and ST-BUS compliant devices including:
    - MVIP framers
    - H.100 framers
    - SCSA framers
  - IOM-2 compliant devices
  - AC97-compliant devices (The necessary multiphase frame capability is provided.)
  - IIS-compliant devices
  - SPI devices
- A wide selection of data sizes: 8, 12, 16, 20, 24, and 32 bits
  Note: A value of the chosen data size is referred to as a *serial word* or *word* throughout the McBSP documentation. Elsewhere, *word* is used to describe a 16-bit value.
- The option of transmitting/receiving 8-bit data with the LSB or MSB first

The McBSP module in C28x devices is adapted from TI's family of McBSPs. Although it supports most of the McBSP applications, Table 1−1 lists some limitations to this implementation.

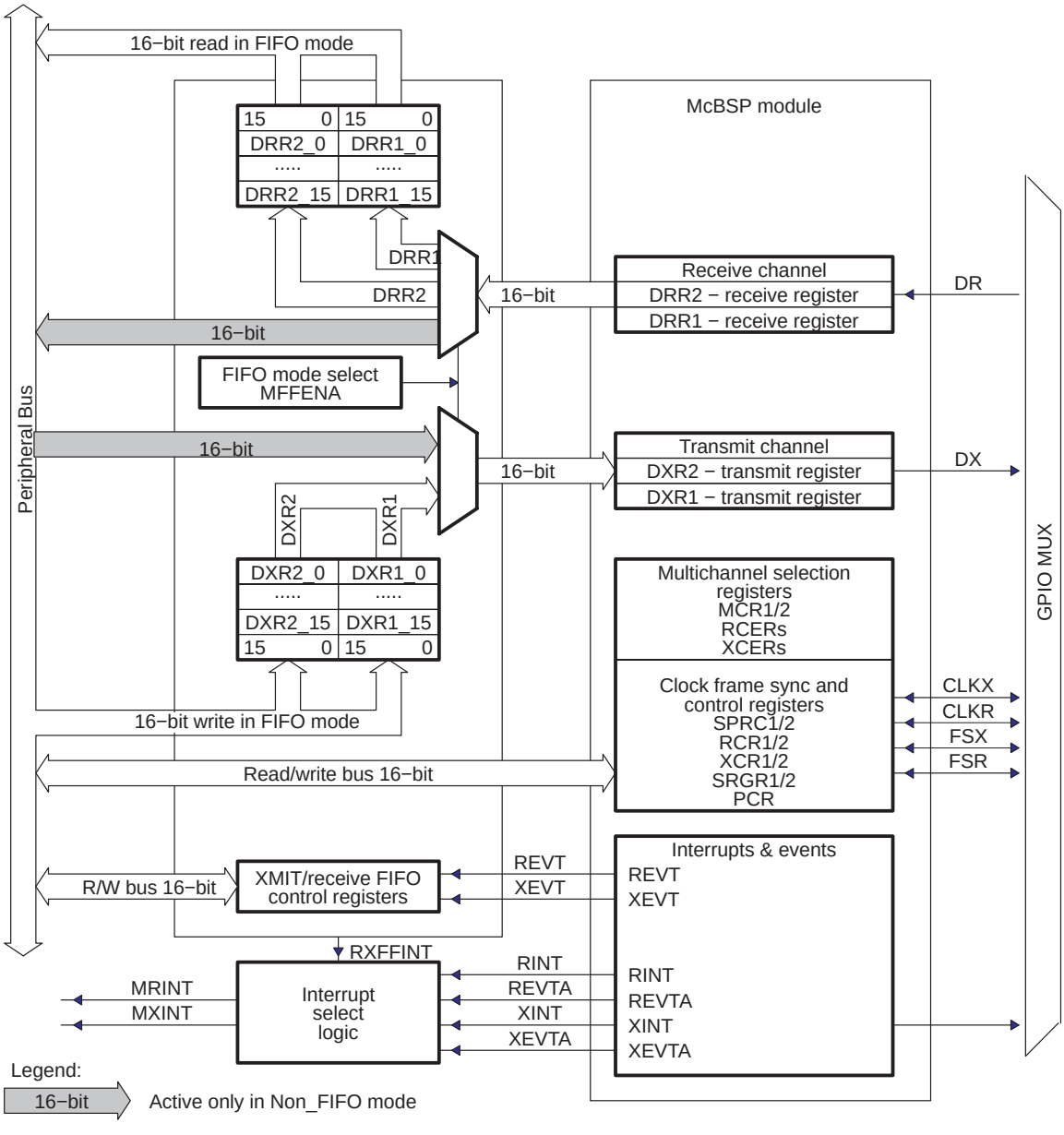
Table 1−1. 28x Implementation Changes

Features of TI's McBSP family	Implementation in 28x McBSP Module
DMA supports for data transfer	DMA replaced by two 32 X 16 level FIFOs
GPIO function on McBSP pins	Supported through the GPIO module available in 28x family.
Power down-mode using IDLE-EN bit in PCR register	The clock for the McBSP logic is controlled using bit 12 of the peripheral clock control register (PCLKCR).
CLKS as external shift clock	CLKS feature is not supported. CLKR/CLKX pin provide this feature.

1.1.2 Block Diagram of the McBSP Module With FIFO

The McBSP consists of a data-flow path and a control path connected to external devices by seven pins as shown in Figure 1−1.

Figure 1−1. Block Diagram With FIFO Interface



Data is communicated to devices interfaced with the McBSP via the data transmit (DX) pin for transmission and the data receive (DR) pin for reception. Control information in the form of clocking and frame synchronization is communicated via the following pins: CLKX (transmit clock), CLKR (receive clock), FSX (transmit frame sync), and FSR (receive frame sync).

DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted) if the serial word length is 8 bits, 12 bits, or 16 bits. For larger word lengths, these registers are needed to hold the most significant bits.

The remaining registers are registers for controlling McBSP operation. Details about these registers are available in section 1.2 (page 1-7).

1.1.3 McBSP Signals

Table 1−2 describes the McBSP interface signals. Information on using these signals for general-purpose input/output is in section 4.3 on page 4-11.

Table 1−2. McBSP Signal Summary

Signal Name	Type †	Reset Status	Description
External Signals			**External Signals Names**
CLKX	I/O/Z	Input	Transmit clock
CLKR	I/O/Z	Input	Receive clock
DR	I	Input	Received serial data
DX	O/Z	High-z	Transmitted serial data
FSR	I/O/Z	Input	Receive frame synchronization
FSX	I/O/Z	Input	Transmit frame synchronization
CPU-Interrupt Signals			
MRINT			Receive interrupt to CPU or FIFO
MXINT			Transmit interrupt to CPU or FIFO
FIFO Events			
REVT			Receive synchronization event to FIFO
XEVT			Transmit synchronization event to FIFO

† I = input, O = output, Z = high impedance

Notes: 1) CLKS signal is not supported in this implementation.

Table 1−2. McBSP Signal Summary (Continued)

Signal Name	Type †	Reset Status	Description
REVTA			Receive synchronization in A-bis mode in the FIFO
XEVTA			Transmit synchronization in A-bis mode in the FIFO

† I = input, O = output, Z = high impedance

Notes: 1) CLKS signal is not supported in this implementation.

1.2 Register Summary

For each McBSP, the DSP contains the following registers. For the address of each register, see the data sheet for your device.

Table 1−3. McBSP Register Summary

Address Offset 0x000 xxh	16-Bit Access	Type R/W	Reset ValueHex	McBSP Communications Interface Register Description
Data Registers, Receive, Transmit				
–	–	–		McBSP Receive Buffer Register
–	–	–		McBSP Receive Shift Register
–	–	–		McBSP Transmit Shift Register
00	DRR2	R	0x0000	McBSP Data Receive Register2 ❑ Read First if the word size is greater than 16 bit else ignore DRR2
01	DRR1	R	0x0000	McBSP Data Receive Register1 ❑ Read second if the word size is greater than 16 bit else read DRR1 only
02	DXR2	W	0x0000	McBSP Data Transmit Register2 ❑ Write First if the word size is greater than 16 bit else ignore DXR2
03	DXR1	W	0x0000	McBSP Data Transmit Register1 ❑ Write second if the word size is greater than 16 bit else Write to DXR1 only
McBSP Control Registers				
04	SPCR2	R/W	0x0000	McBSP Serial Port Control Register2
05	SPCR1	R/W	0x0000	McBSP Serial Port Control Register1
06	RCR2	R/W	0x0000	McBSP Receive Control Register2
07	RCR1	R/W	0x0000	McBSP Receive Control Register1
08	XCR2	R/W	0x0000	McBSP Transmit Control Register2
09	XCR1	R/W	0x0000	McBSP Transmit Control Register1

† MFFST will read 0x000A if FSX/FSR pins are left unconnected; else, it will assume pin status at reset.

Table 1−3. McBSP Register Summary (Continued)

Address Offset 0x000 xxh	16-Bit Access	Type R/W	Reset ValueHex	McBSP Communications Interface Register Description
Multichannel Control Registers (Continued)				
0A	SRGR2	R/W	0x2000	McBSP Sample Rate Generator Register2
0B	SRGR1	R/W	0x0001	McBSP Sample Rate Generator Register1
0C	MCR2	R/W	0x0000	McBSP Multi−Channel Register2
0D	MCR1	R/W	0x0000	McBSP Multi−Channel Register1
0E	RCERA	R/W	0x0000	McBSP Receive Channel Enable Register Partition A
0F	RCERB	R/W	0x0000	McBSP Receive Channel Enable Register Partition B
10	XCERA	R/W	0x0000	McBSP Transmit Channel Enable Register Partition A
11	XCERB	R/W	0x0000	McBSP Transmit Channel Enable Register Partition A
12	PCR1	R/W	0x0000	McBSP Pin Control Register
13	RCERC	R/W	0x0000	McBSP Receive Channel Enable Register Partition C
14	RCERD	R/W	0x0000	McBSP Receive Channel Enable Register Partition D
15	XCERC	R/W	0x0000	McBSP Transmit Channel Enable Register Partition C
16	XCERD	R/W	0x0000	McBSP Transmit Channel Enable Register Partition D
17	RCERE	R/W	0x0000	McBSP Receive Channel Enable Register Partition E
18	RCERF	R/W	0x0000	McBSP Receive Channel Enable Register Partition F
19	XCERE	R/W	0x0000	McBSP Transmit Channel Enable Register Partition E
1A	XCERF	R/W	0x0000	McBSP Transmit Channel Enable Register Partition F
1B	RCERG	R/W	0x0000	McBSP Receive Channel Enable Register Partition G
1C	RCERH	R/W	0x0000	McBSP Receive Channel Enable Register Partition H
1D	XCERG	R/W	0x0000	McBSP Transmit Channel Enable Register Partition G
1E	XCERH	R/W	0x0000	McBSP Transmit Channel Enable Register Partition H

† MFFST will read 0x000A if FSX/FSR pins are left unconnected; else, it will assume pin status at reset.

Table 1−3. McBSP Register Summary (Continued)

Address Offset 0x000 xxh	16-Bit Access	Type R/W	Reset ValueHex	McBSP Communications Interface Register Description
FIFO Mode Data Registers Applicable Only FIFO Mode				
00	DRR2	R	0x0000	McBSP Data Receive Register2 −Top of receive FIFO ❑ Read First FIFO pointers will not advance
01	DRR1	R	0x0000	McBSP Data Receive Register1−Top of receive FIFO ❑ Read second for FIFO pointers to advance
02	DXR2	W	0x0000	McBSP Data Transmit Register2−Top of transmit FIFO ❑ Write First FIFO pointers will not advance
03	DXR1	W	0x0000	McBSP Data Transmit Register1−Top of transmit FIFO ❑ Write second for FIFO pointers to advance
FIFO Control Registers				
20	MFFTX	R/W	0x2000	McBSP FIFO Transmit register
21	MFFRX	R/W	0x201F	McBSP FIFO Receive register
22	MFFCT	R/W	0x0000	McBSP FIFO Control register
23	MFFINT	R/W	0x0000	McBSP FIFO Interrupt register
24	MFFST	R/W	0x000x†	McBSP FIFO Status register

† MFFST will read 0x000A if FSX/FSR pins are left unconnected; else, it will assume pin status at reset.

The registers bits are summarized in Table 6−5.

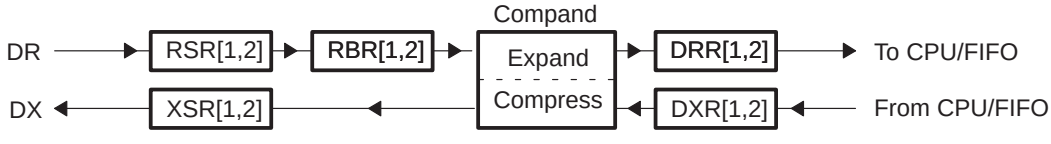
1.3 McBSP Operation

This section contains the following topics:

1.3.1 Data Transfer Process of a McBSP

Figure 1−2 shows a diagram of the McBSP data transfer paths. McBSP receive operation is triple buffered, and transmit operation is double buffered. The use of registers varies depending on whether the defined length of each serial word fits in 16 bits.

Figure 1−2. McBSP Data Transfer Paths



1.3.1.1 Data Transfer Process for Word Length of 8, 12, or 16 Bits

If the word length is 16 bits or smaller, only one 16-bit register is needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are not used (written, read, or shifted).

Receive data arrives on the DR pin and is shifted into receive shift register 1 (RSR1). Once a full word is received, the content of RSR1 is copied to receive buffer register 1 (RBR1), only if RBR1 is not full with previous data. RBR1 is then copied to data receive register 1 (DRR1), unless the previous content of DRR1 has not been read by the CPU. If the companding feature of the McBSP is implemented, the required word length is 8 bits and receive data is expanded into the appropriate format before being passed from RBR1 to DRR1. For more details about reception, see section 1.3.5 on page 1-21.

Transmit data is written by the CPU to data transmit register 1 (DXR1). If there is no previous data in transmit shift register (XSR1), the value in DXR1 is copied to XSR1; otherwise, DXR1 is copied to XSR1 when the last bit of the previous data is shifted out on the DX pin. If selected, the companding module compresses 16-bit data into the appropriate 8-bit format before passing it to XSR1. After transmit frame synchronization, the transmitter begins shifting bits from XSR1 to the DX pin. For more details about transmission, see section 1.3.6 on page 1-22.

1.3.1.2 Data Transfer Process for Word Length of 20, 24, or 32 Bits

If the word length is larger than 16 bits, two 16-bit registers are needed at each stage of the data transfer paths. The registers DRR2, RBR2, RSR2, DXR2, and XSR2 are needed to hold the most significant bits.

Receive data arrives on the DR pin and is shifted into RSR2 first and then into RSR1. Once the full word is received, the contents of RSR2 and RSR1 are copied to RBR2 and RBR1, respectively, only if RBR1 is not full. Then the contents of RBR2 and RBR1 are copied to DRR2 and DRR1, respectively, unless the previous content of DRR1 has not been read by the CPU. The CPU must read data from DRR2 first and then from DRR1. When DRR1 is read, the next RBR-to-DRR copy occurs. For more details about reception, see section 1.3.5 on page 1-21.

For transmission, the CPU must write data to DXR2 first and then to DXR1. When new data arrives in DXR1, if there is no previous data in XSR1, the contents of DXR2 and DXR1 are copied to XSR2 and XSR1, respectively; otherwise, the contents of the DXRs are copied to the XSRs when the last bit of the previous data is shifted out on the DX pin. After transmit frame synchronization, the transmitter begins shifting bits from the XSRs to the DX pin. For more details about transmission, see section 1.3.6 on page 1-22.

1.3.2 Companding (Compressing and Expanding) Data

Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

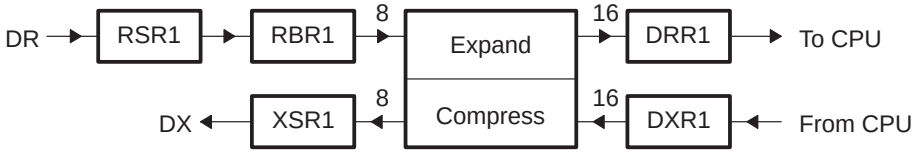
A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits

(RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 1−3 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to 2s-complement format.

Figure 1−3. Companding Processes

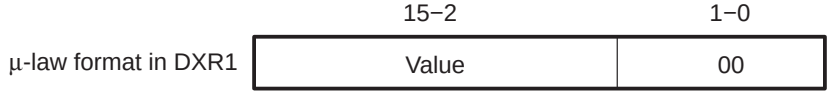


1.3.2.1 Companding Formats

For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. The receive sign-extension and justification mode specified in RJUST is ignored when companding is used.

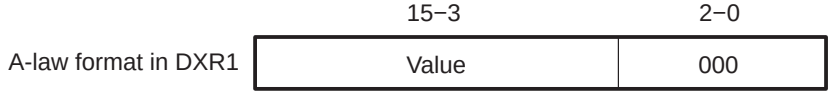
For transmission using μ-law compression, make sure the 14 data bits are left-justified in DXR1, with the remaining two low-order bits filled with 0s as shown in Figure 1−4.

Figure 1−4. μ-Law Transmit Data Companding Format



For transmission using A-law compression, make sure the 13 data bits are left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 1−5.

Figure 1−5. A-Law Transmit Data Companding Format



1.3.2.2 Capability to Compand Internal Data

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. This can be used to:

- Convert linear to the appropriate μ-law or A-law format.
- Convert μ-law or A-law to the linear format.
- Observe the quantization effects in companding by transmitting linear data, and compressing and re-expanding this data. This is useful only if both XCOMPAND and RCOMPAND enable the same companding format.

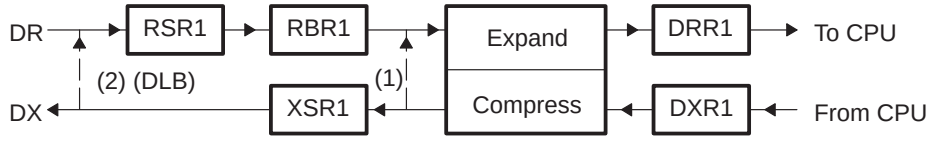
Figure 1−6 shows two methods by which the McBSP can compand internal data. Data paths for these two methods are used to indicate:

- When both the transmit and receive sections of the serial port are reset, DRR1 and DXR1 are connected internally through the companding logic. Values from DXR1 are compressed, as selected by XCOMPAND, and then expanded, as selected by RCOMPAND. Note that RRDY and XRDY bits are not set. However, data is available in DRR1 within four CPU clocks after being written to DXR1.

  The advantage of this method is its speed. The disadvantage is that there is no synchronization available to the CPU to control the flow. Note that DRR1 and DXR1 are internally connected if the (X/R)COMPAND bits are set to 10b or 11b (compand using μ-law or A-law).

- The McBSP is enabled in digital loopback mode with companding appropriately enabled by RCOMPAND and XCOMPAND. Receive and transmit interrupts (RINT when RINTM = 0 and XINT when XINTM = 0) or synchronization events (REVT and XEVT) allow synchronization of the CPU to these conversions. The time for this companding depends on the serial bit rate selected.

Figure 1−6. Two Methods by Which the McBSP Can Compand Internal Data



1.3.2.3 Reversing Bit Order: Option to Transfer LSB First

Normally, the McBSP transmit or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

1.3.3 Clocking and Framing Data

This section explains basic concepts and terminology important for understanding how McBSP data transfers are timed and delimited.

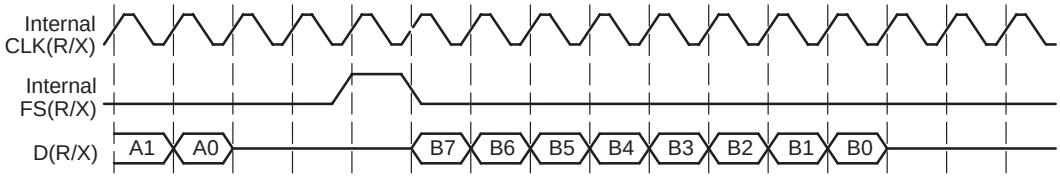
1.3.3.1 Clocking

Data is shifted one bit at a time from the DR pin to the RSR(s) or from the XSR(s) to the DX pin. The time for each bit transfer is controlled by the rising or falling edge of a clock signal.

The receive clock signal (CLKR) controls bit transfers from the DR pin to the RSR(s). The transmit clock signal (CLKX) controls bit transfers from the XSR(s) to the DX pin. CLKR or CLKX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP. The polarities of CLKR and CLKX are programmable.

In Figure 1−7, the clock signal controls the timing of each bit transfer on the pin.

Figure 1−7. Clock Signal Control Waveform



Note:

The McBSP cannot operate at a frequency faster than 1/2 the CPU clock frequency. When driving CLKX or CLKR at the pin, choose an appropriate input clock frequency. When using the internal sample rate generator for CLKX and/or CLKR, choose an appropriate input clock frequency and divide down value (CLKDV).

1.3.3.2 Serial Words

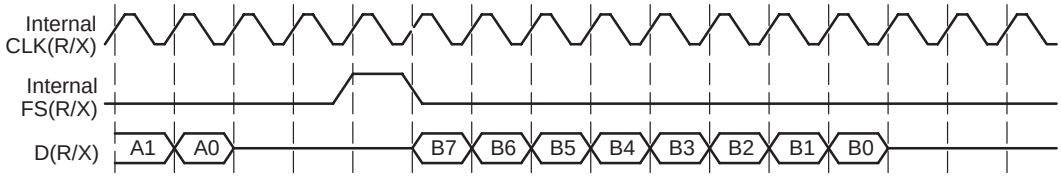
Bits traveling between a shift register (RSR or XSR) and a data pin (DR or DX) are transferred in a group called a **serial word**. You define how many bits are in a word.

Bits coming in on the DR pin are held in RSR until RSR holds a full serial word. Only then is the word passed to RBR (and ultimately to the DRR).

During transmission, XSR does not accept new data from DXR until a full serial word has been passed from XSR to the DX pin.

In Figure 1−8, an 8-bit word size was defined (see bits 7 through 0 of word B being transferred).

Figure 1−8. 8-Bit Word Size Defined Waveform



1.3.3.3 Frames and Frame Synchronization

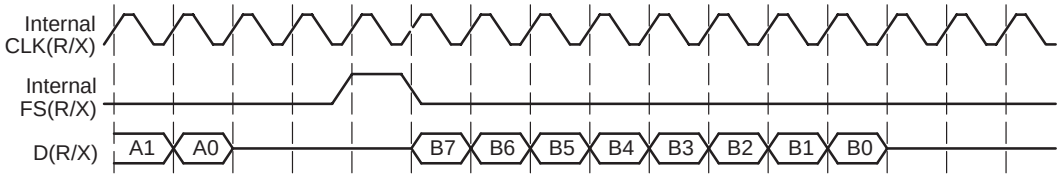
One or more words are transferred in a group called a **frame**. You define how many words are in a frame.

All of the words in a frame are sent in a continuous stream. However, there can be pauses between frame transfers. The McBSP uses frame-synchronization (frame-sync) signals to determine when each frame is received/transmitted. When a pulse occurs on a frame-sync signal, the McBSP begins receiving/transmitting a frame of data. When the next pulse occurs, the McBSP receives/transmits the next frame, and so on.

Pulses on the receive frame-sync signal (FSR) initiate frame transfers on DR. Pulses on the transmit frame-sync signal (FSX) initiate frame transfers on DX. FSR or FSX can be derived from a pin at the boundary of the McBSP or derived from inside the McBSP.

In Figure 1−9, a 1-word frame is transferred when a frame-sync pulse occurs.

Figure 1−9. One-word Frame Transfer



In McBSP operation, the inactive-to-active transition of the frame-synchronization signal indicates the start of the next frame. For this reason, the frame-sync signal may be high for an arbitrary number of clock cycles. Only after the signal is recognized to have gone inactive, and then active again, does the next frame synchronization occur.

1.3.3.4 Detecting Frame-Sync Pulses, Even in the Reset State

The McBSP can send receive and transmit interrupts to the CPU to indicate specific events in the McBSP. To facilitate detection of frame synchronization, these interrupts can be sent in response to frame-sync pulses. Set the appropriate interrupt mode bits to 10b (for reception, RINTM = 10b; for transmission, XINTM = 10b).

Unlike other serial port interrupt modes, this mode can operate while the associated portion of the serial port is in reset (such as activating RINT when the receiver is in reset). In this case, FSRM/FSXM and FSRP/FSXP still select the appropriate source and polarity of frame synchronization. Thus, even when the serial port is in the reset state, these signals are synchronized to the CPU clock and then sent to the CPU in the form of RINT and XINT at the point at which they feed the receiver and transmitter of the serial port. Consequently, a new frame-synchronization pulse can be detected, and after this occurs the CPU can take the serial port out of reset safely.

1.3.3.5 Ignoring Frame-Sync Pulses

The McBSP can be configured to ignore transmit and/or receive frame-synchronization pulses. To have the receiver or transmitter recognize frame-sync pulses, clear the appropriate frame-sync ignore bit (RFIG = 0 for the receiver, XFIG = 0 for the transmitter). To have the receiver or transmitter ignore frame-sync pulses until the desired frame length or number of words is reached, set the appropriate frame-sync ignore bit (RFIG = 1 for the receiver, XFIG = 1 for the transmitter). For more details on unexpected frame-sync pulses, see one of the following topics:

- ❏ *Unexpected Receive Frame-Sync Pulse* (page 1-40)
- ❏ *Unexpected Transmit Frame-Sync Pulse* (page 1-46)

You can also use the frame-sync ignore function for data packing (for more details, see section 4.2.2 on page 4-9).

1.3.3.6 Frame Frequency

The frame frequency is determined by the period between frame-synchronization pulses and is defined as shown by Equation 1−1.

Equation 1−1. Frame Frequency of a McBSP

$$\text{Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Clock Cycles Between Frame−Sync Pulses}}$$

The frame frequency may be increased by decreasing the time between frame-synchronization pulses (limited only by the number of bits per frame). As the frame transmit frequency increases, the inactivity period between the data packets for adjacent transfers decreases to zero.

1.3.3.7 Maximum Frame Frequency

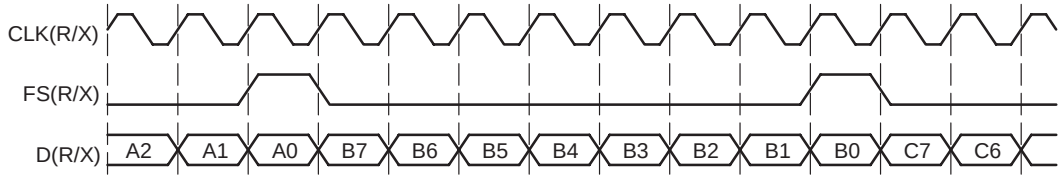
The minimum number of clock cycles between frame synchronization pulses is equal to the number of bits transferred per frame. The maximum frame frequency is defined as shown by Equation 1−2.

Equation 1−2. Maximum Frame Frequency of a McBSP

$$\text{Maximum Frame Frequency} = \frac{\text{Clock Frequency}}{\text{Number of Bits Per Frame}}$$

Figure 1−10 shows the McBSP operating at maximum packet frequency. At maximum packet frequency, the data bits in consecutive packets are transmitted contiguously with no inactivity between bits.

Figure 1−10. McBSP Operating at Maximum Packet Frequency



If there is a 1-bit data delay as shown in this figure, the frame-synchronization pulse overlaps the last bit transmitted in the previous frame. Effectively, this permits a continuous stream of data, making frame-synchronization pulses redundant. Theoretically, only an initial frame-synchronization pulse is required to initiate a multipacket transfer.

The McBSP supports operation of the serial port in this fashion by ignoring the successive frame-sync pulses. Data is clocked in to the receiver, or clocked out of the transmitter, during every clock cycle.

Note:

For XDATDLY = 0 (0-bit data delay), the first bit of data is transmitted asynchronously to the internal transmit clock signal (CLKX). For more details, see *Set the Transmit Data Delay* on page 3-36.

1.3.4 Frame Phases

The McBSP allows you to configure each frame to contain one or two phases. The number of words per frame, and the number of bits per word, can be specified differently for each of the two phases of a frame, allowing greater flexibility in structuring data transfers. For example, a user might define a frame as consisting of one phase containing two words of 16 bits each, followed by a second phase consisting of 10 words of 8 bits each. This configuration permits the user to compose frames for custom applications, or in general, to maximize the efficiency of data transfers.

1.3.4.1 Number of Phases, Words, and Bits Per Frame

Table 1−4 shows which bit fields in the receive control registers (RCR1 and RCR2) and in the transmit control registers (XCR1 and XCR2) determine the number of phases per frame, the number of words per frame, and number of bits per word for each phase, for the receiver and transmitter. The maximum number of words per frame is 128 for a single-phase frame and 256 for a dual-phase frame. The number of bits per word can be 8, 12, 16, 20, 24, or 32 bits.

Table 1−4. McBSP Register Bits That Determine the Number of Phases, Words, and Bits Per Frame

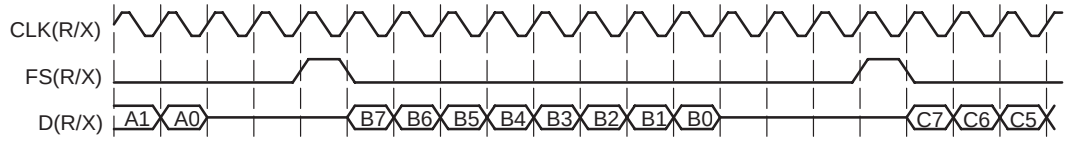
Operation	Number of Phases	Words Per Frame Set With ...	Bits Per Word Set With ...
Reception	1 (RPHASE = 0)	RFRLEN1	RWDLEN1
Reception	2 (RPHASE = 1)	RFRLEN1 and RFRLEN2	RWDLEN1 for phase 1 RWDLEN2 for phase 2
Transmission	1 (XPHASE = 0)	XFRLEN1	XWDLEN1
Transmission	2 (XPHASE = 1)	XFRLEN1 and XFRLEN2	XWDLEN1 for phase 1 XWDLEN2 for phase 2

1.3.4.2 Single-Phase Frame Example

Figure 1−11 shows an example of a single-phase data frame comprising one 8-bit word. Since the transfer is configured for one data bit delay, the data on the DX and DR pins are available one clock cycle after FS(R/X) goes active. The figure makes the following assumptions:

- ❑ (R/X)PHASE = 0: Single-phase frame
- ❑ (R/X)FRLEN1 = 0b: 1 word per frame
- ❑ (R/X)WDLEN1 = 000b: 8-bit word length
- ❑ (R/X)FRLEN2 and (R/X)WDLEN2 are ignored
- ❑ CLK(X/R)P = 0: Receive data clocked on falling edge; transmit data clocked on rising edge
- ❑ FS(R/X)P = 0: Active-high frame-sync signals
- ❑ (R/X)DATDLY = 01b: 1-bit data delay

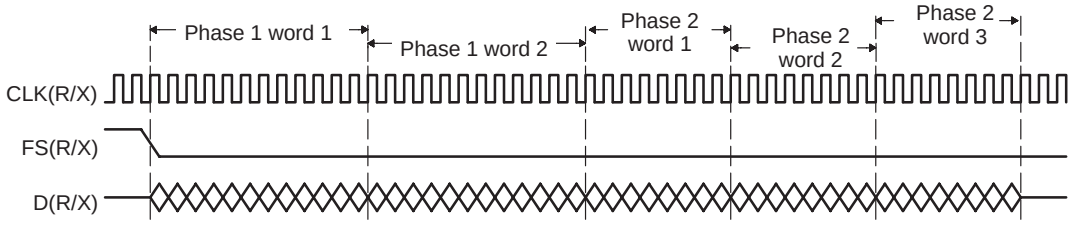
Figure 1−11. Single-Phase Frame for a McBSP Data Transfer



1.3.4.3 Dual-Phase Frame Example

Figure 1−12 shows an example of a frame where the first phase consists of 2 words of 12 bits each followed by a second phase of three words of 8 bits each. Note that the entire bit stream in the frame is contiguous. There are no gaps either between words or between phases.

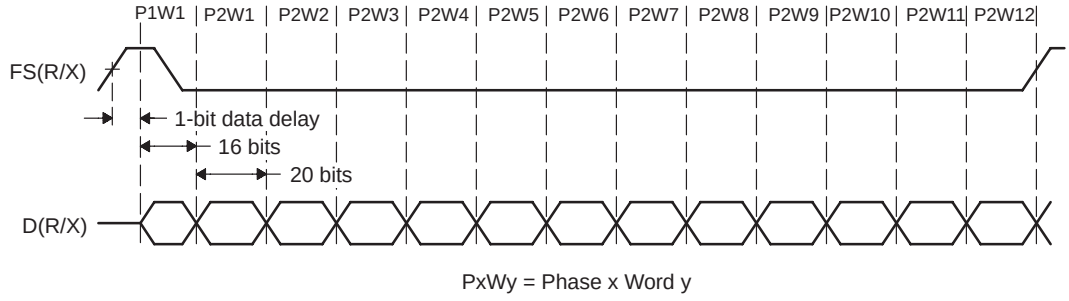
Figure 1−12. Dual-Phase Frame for a McBSP Data Transfer



1.3.4.4 Implementing the AC97 Standard With a Dual-Phase Frame

Figure 1−13 shows an example of the Audio Codec ‘97 (AC97) standard, which uses the dual-phase frame feature. Notice that words, not individual bits, are shown on the D(R/X) signal. The first phase (P1) consists of a single 16-bit word. The second phase (P2) consists of twelve 20-bit words. The phase configurations are listed after the figure.

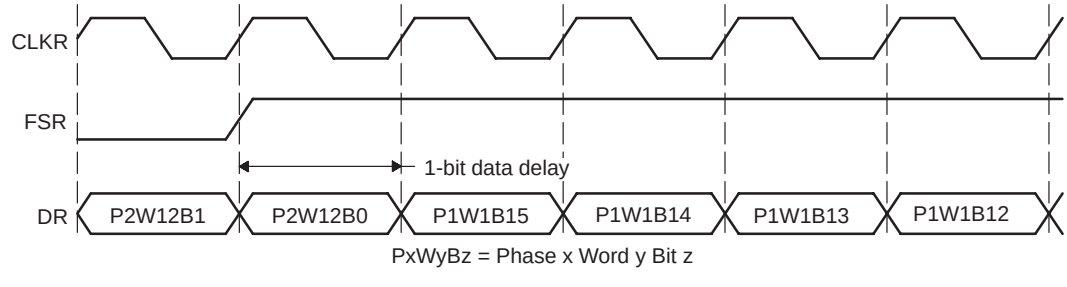
Figure 1−13. Implementing the AC97 Standard With a Dual-Phase Frame



- ❏ (R/X)PHASE = 1: Dual-phase frame
- ❏ (R/X)FRLEN1 = 0000000b: 1 word in phase 1
- ❏ (R/X)WDLEN1 = 010b: 16 bits per word in phase 1
- ❏ (R/X)FRLEN2 = 0001011b: 12 words in phase 2
- ❏ (R/X)WDLEN2 = 011b: 20 bits per word in phase 2
- ❏ CLKRP/CLKXP= 0: Receive data sampled on falling edge of internal CLKR / transmit data clocked on rising edge of internal CLKX
- ❏ FSRP/FSXP = 0: Active-high frame-sync signal
- ❏ (R/X)DATDLY = 01b: Data delay of 1 clock cycle (1-bit data delay)

Figure 1−14 shows the timing of an AC97-standard data transfer near frame synchronization. In this figure, individual bits are shown on D(R/X). Specifically, the figure shows the last two bits of phase 2 of one frame and the first four bits of phase 1 of the next frame. Regardless of the data delay, data transfers can occur without gaps. The first bit of the second frame (P1W1B15) immediately follows the last bit of the first frame (P2W12B0). Because a 1-bit data delay has been chosen, the transition on the frame-sync signal can occur when P2W12B0 is transferred.

Figure 1−14. Timing of an AC97-Standard Data Transfer Near Frame Synchronization

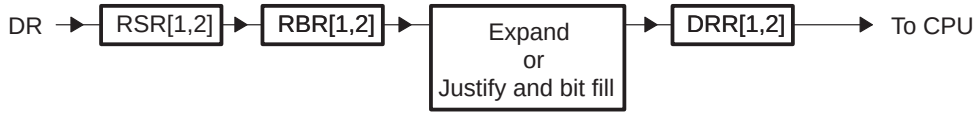


1.3.5 McBSP Reception

This section explains the fundamental process of reception in the McBSP. For details about how to program the McBSP receiver, see *Receiver Configuration* on page 3-2.

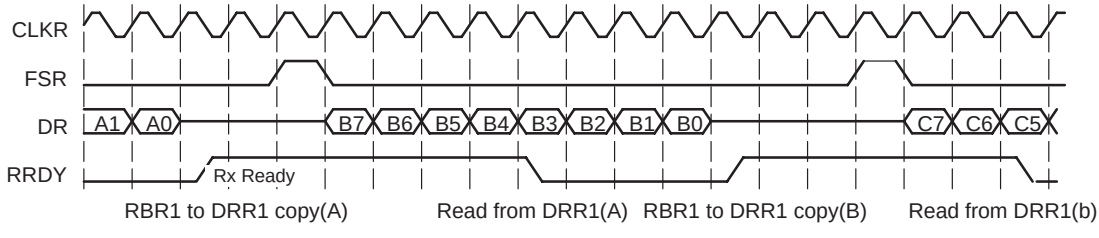
Figure 1−15 and Figure 1−16 show how reception occurs in the McBSP. Figure 1−15 shows the physical path for the data. Figure 1−16 is a timing diagram showing signal activity for one possible reception scenario. A description of the process follows the figures.

Figure 1−15. McBSP Reception Physical Data Path



RSR[1,2]: Receive shift registers 1 and 2
RBR[1,2]: Receive buffer registers 1 and 2
DRR[1,2]: Data receive registers 1 and 2

Figure 1−16. McBSP Reception Signal Activity



CLKR: Internal receive clock
FSR: Internal receive frame-sync signal
DR: Data on DR pin
RRDY: Status of receiver ready bit (high is 1)

The following process describes how data travels from the DR pin to the CPU:

1) The McBSP waits for a receive frame-sync pulse on internal FSR.

2) When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the RDATDLY bits of RCR2.

   In the preceding timing diagram (Figure 1−16), a 1-bit data delay is selected.

3) The McBSP accepts data bits on the DR pin and shifts them into the receive shift register(s).

   If the word length is 16 bits or smaller, only RSR1 is used. If the word length is larger than 16 bits, RSR2 and RSR1 are used, and RSR2 contains the

most significant bits. For details on choosing a word length, see *Set the Receive Word Length(s)*.

4) When a full word is received, the McBSP copies the contents of the receive shift register(s) to the receive buffer register(s), provided that RBR1 is not full with previous data.

   If the word length is 16 bits or smaller, only RBR1 is used. If the word length is larger than 16 bits, RBR2 and RBR1 are used, and RBR2 contains the most significant bits.

5) The McBSP copies the contents of the receive buffer register(s) into the data receive register(s), provided that DRR1 is not full with previous data. When DRR1 receives new data, the receiver ready bit (RRDY) is set in SPCR1. This indicates that receive data is ready to be read by the CPU.

   If the word length is 16 bits or smaller, only DRR1 is used. If the word length is larger than 16 bits, DRR2 and DRR1 are used, and DRR2 contains the most significant bits.

   If companding is used during the copy (RCOMPAND = 10b or 11b in RCR2), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.

6) The CPU reads the data from the data receive register(s). When DRR1 is read, RRDY is cleared and the next RBR-to-DRR copy is initiated.

Note:

If both DRRs are needed (word length larger than 16 bits), the CPU must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

When activity is not properly timed, errors can occur. See the following topics for more details:

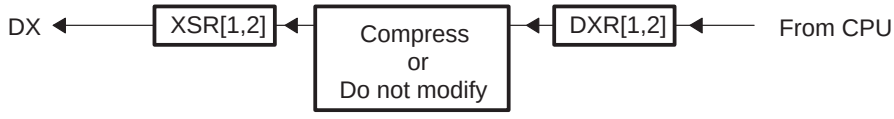
- *Overrun in the Receiver* (page 1-39)
- *Unexpected Receive Frame-Sync Pulse* (page 1-40)

1.3.6 McBSP Transmission

This section explains the fundamental process of transmission in the McBSP. For details about how to program the McBSP transmitter, see *Transmitter Configuration* on page 3-26.

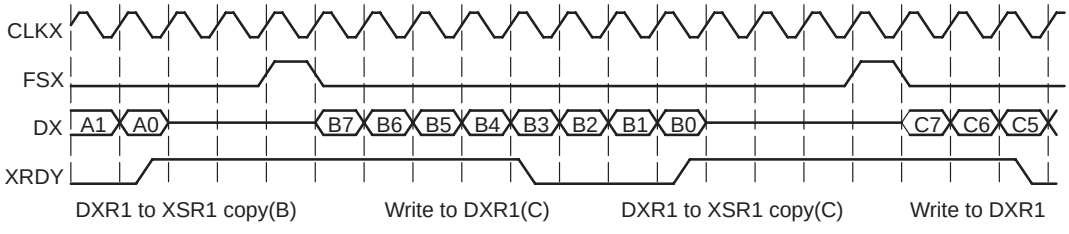
Figure 1−17 and Figure 1−18 show how transmission occurs in the McBSP. Figure 1−17 shows the physical path for the data. Figure 1−18 is a timing diagram showing signal activity for one possible transmission scenario. A description of the process follows the figures.

Figure 1−17. McBSP Transmission Physical Data Path



XSR[1,2]: Transmit shift registers 1 and 2 DXR[1,2]: Data transmit registers 1 and 2

Figure 1−18. McBSP Transmission Signal Activity



CLKX: Internal transmit clock
FSX: Internal transmit frame-sync signal
DX: Data on DX pin
XRDY: Status of transmitter ready bit (high is 1)

1) The CPU writes data to the data transmit register(s). When DXR1 is loaded, the transmitter ready bit (XRDY) is cleared in SPCR2 to indicate that the transmitter is not ready for new data.

   If the word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, DXR2 and DXR1 are used, and DXR2 contains the most significant bits. For details on choosing a word length, see *Set the Transmit Word Length(s)* in Chapter 3.

Note:

If both DXRs are needed (word length larger than 16 bits), the CPU must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs), as described in the next step. If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

2) When new data arrives in DXR1, the McBSP copies the content of the data transmit register(s) to the transmit shift register(s). In addition, the transmit

ready bit (XRDY) is set. This indicates that the transmitter is ready to accept new data from the CPU.

If the word length is 16 bits or smaller, only XSR1 is used. If the word length is larger than 16 bits, XSR2 and XSR1 are used, and XSR2 contains the most significant bits.

If companding is used during the transfer (XCOMPAND = 10b or 11b in XCR2), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

3) The McBSP waits for a transmit frame-sync pulse on internal FSX.

4) When the pulse arrives, the McBSP inserts the appropriate data delay that is selected with the XDATDLY bits of XCR2.

   In the preceding timing diagram (Figure 1−18), a 1-bit data delay is selected.

5) The McBSP shifts data bits from the transmit shift register(s) to the DX pin.

When activity is not properly timed, errors can occur. See the following topics for more details:

- *Overwrite in the Transmitter* (page 1-43)
- *Underflow in the Transmitter* (page 1-44)
- *Unexpected Transmit Frame-Sync Pulse* (page 1-46)

1.3.7 Interrupts and FIFO Events Generated by a McBSP

The McBSP sends notification of important events to the CPU and FIFO via the internal signals shown in Table 1−5.

Table 1−5. Interrupts and FIFO Events Generated by a McBSP

Internal Signal	Description
RINT	Receive interrupt The McBSP can send a receive interrupt request to CPU based upon a selected condition in the receiver of the McBSP (a condition selected by the RINTM bits of SPCR1).
XINT	Transmit interrupt The McBSP can send a transmit interrupt request to CPU based upon a selected condition in the transmitter of the McBSP (a condition selected by the XINTM bits of SPCR2).

Table 1−5. Interrupts and FIFO Events Generated by a McBSP (Continued)

Internal Signal	Description
REVT	Receive synchronization event An REVT signal is sent to the FIFO when data has been received in the data receive registers (DRRs).
XEVT	Transmit synchronization event An XEVT signal is sent to the FIFO when the data transmit registers (DXRs) are ready to accept the next serial word for transmission.
REVTA	A-bis mode receive synchronization event If ABIS = 1 (A-bis mode is enabled) an REVTA signal is sent to the FIFO every 16 cycles.
XEVTA	A-bis mode transmit synchronization event If ABIS = 1 (A-bis mode is enabled) an XEVTA signal is sent to the FIFO every 16 cycles.

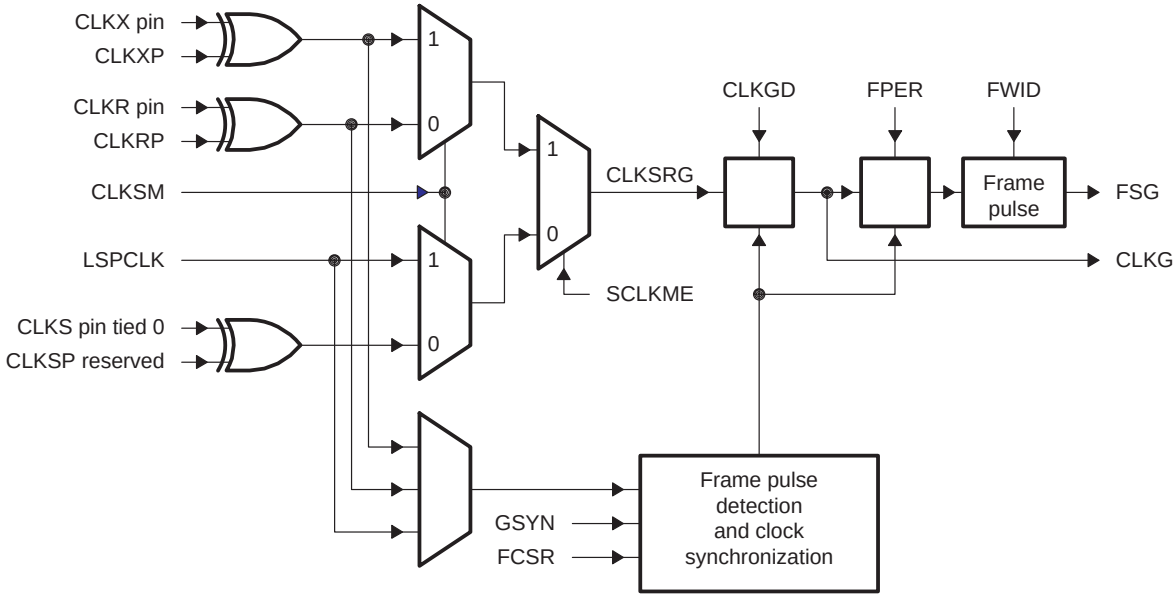
1.4 Sample Rate Generator of the McBSP

Note: General-purpose I/O Selection

Do not use the GPIO function using RIOEN/XIOEN bits 12 and 13 in the PCR register. This feature is not applicable to the 28x McBSP implementation; therefore, these bits are reserved on 28x devices.

The McBSP contains a sample rate generator module that can be programmed to generate internal data-clock (CLKG) and an internal frame-synchronization (FSG) signal. Figure 1−19 is a conceptual block diagram that shows the clock selection options available on the McBSP. The clock selection to the module is selectable using the SCLKME bit (PCR bit 7) and CLKSM bit (SRGR2 bit 13).

Figure 1−19. Sample Rate Generator Clock Selection



The McBSP contains a sample rate generator that can be used to generate an internal data clock (CLKG) and an internal frame-synchronization signal (FSG). CLKG can be used for bit shifting on the data receive (DR) pin and/or the data transmit (DX) pin. FSG can be used to initiate frame transfers on DR and/or DX.

The source clock for the sample rate generator (labeled CLKSRG in the diagram) can be supplied by the LSPCLK or by an external pin (CLKS, CLKX, or CLKR). The source is selected with the SCLKME bit of PCR and the CLKSM

bit of SRGR2. If a pin is used, the polarity of the incoming signal can be inverted with the appropriate polarity bit (CLKSP of SRGR2, CLKXP of PCR, or CLKRP of PCR).

The sample rate generator has a 3-stage clock divider that gives CLKG and FSG programmability. The three stages provide:

- Clock divide down. The source clock is divided according to the CLKGDV bits of SRGR1 to produce CLKG.
- Frame period divide down. CLKG is divided according to the FPER bits of SRGR2 to control the period from the start of a frame-sync pulse to the start of the next pulse.
- Frame-sync pulse width countdown. CLKG cycles are counted according to the FWID bits of SRGR1 to control the width of each frame-sync pulse.

In addition to the 3-stage clock divider, the sample rate generator has a frame-sync pulse detection and clock synchronization module that allows synchronization of the clock divide down with an incoming frame-sync pulse on the FSR pin. This feature is enabled or disabled with the GSYNC bit of SRGR2.

For details on getting the sample rate generator ready for operation, see the reset and initialization procedure on page 1-33.

1.4.1 Clock Generation in the Sample Rate Generator

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both. Use of the sample rate generator to drive clocking is controlled by the clock mode bits (CLKRM and CLKXM) in the pin control register (PCR). When a clock mode bit is set to 1 (CLKRM = 1 for reception, CLKXM = 1 for transmission), the corresponding data clock (CLKR for reception, CLKX for transmission) is driven by the internal sample rate generator output clock (CLKG).

Note that the effects of CLKRM = 1 and CLKXM = 1 on the McBSP are partially affected by the use of the digital loopback mode and the clock stop (SPI) mode, respectively, as described in Table 1−6. The digital loopback mode (described in Chapter 3) is selected with the DLB bit of SPCR1. The clock stop mode is selected with the CLKSTP bits of SPCR1.

When using the sample rate generator as a clock source, make sure the sample rate generator is enabled ($\overline{\text{GRST}}$ = 1).

Table 1−6. Effects of DLB and CLKSTP on Clock Modes

Mode Bit Settings		Effect
CLKRM = 1	DLB = 0 (Digital loopback mode disabled)	CLKR is an output pin driven by the sample rate generator output clock (CLKG).
	DLB = 1 (Digital loopback mode enabled)	CLKR is an output pin driven by internal CLKX. The source for CLKX depends on the CLKXM bit.
CLKXM = 1	CLKSTP = 00b or 01b (Clock stop (SPI) mode disabled)	CLKX is an output pin driven by the sample rate generator output clock (CLKG).
	CLKSTP = 10b or 11b (Clock stop (SPI) mode enabled)	The McBSP is a master in an SPI system. Internal CLKX drives internal CLKR and the shift clocks of any SPI-compliant slave devices in the system. CLKX is driven by the internal sample rate generator.

1.4.1.1 Choosing an Input Clock

The sample rate generator must be driven by an input clock signal from one of the four sources selectable with the SCLKME bit of PCR and the CLKSM bit of SRGR2 (see Table 1−7). When CLKSM = 1, the minimum divide down value in CLKGDV bits should be 1. CLKGDV is described in section 1.4.1.3.

Note:

The McBSP cannot operate at a frequency faster than one-half the LSPCLK. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to one-half the LSPCLK.

Table 1−7. Choosing an Input Clock for the Sample Rate Generator With the SCLKME and CLKSM Bits

SCLKME	CLKSM	Input Clock For Sample Rate Generator
0	0	Reserved
0	1	LSPCLK
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

1.4.1.2 Choosing a Polarity for the Input Clock

As shown in Figure 1−20, when the input clock is received from a pin, you can choose the polarity of the input clock. The rising edge of CLKSRG generates CLKG and FSG, but you can determine which edge of the input clock causes a rising edge on CLKSRG. The polarity options and their effects are described in Table 1−8.

Figure 1−20. Possible Inputs to the Sample Rate Generator and the Polarity Bits

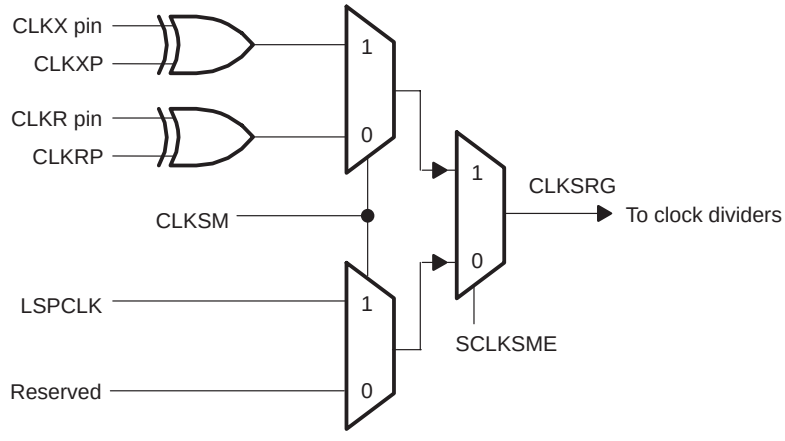


Table 1−8. Polarity Options for the Input to the Sample Rate Generator

Input Clock	Polarity Option	Effect
LSPCLK	Always positive polarity	Rising edge of CPU clock generates transitions on CLKG and FSG.
Signal on CLKR pin	CLKRP = 0 in PCR	Falling edge on CLKR pin generates transitions on CLKG and FSG.
	CLKRP = 1 in PCR	Rising edge on CLKR pin generates transitions on CLKG and FSG.
Signal on CLKX pin	CLKXP = 0 in PCR	Rising edge on CLKX pin generates transitions on CLKG and FSG.
	CLKXP = 1 in PCR	Falling edge on CLKX pin generates transitions on CLKG and FSG.

1.4.1.3 Choosing a Frequency for the Output Clock (CLKG)

The input clock (LSPCLK or external clock) can be divided down by a programmable value to drive CLKG. Regardless of the source to the sample rate generator, the rising edge of CLKSRG (see the sample rate generator diagram on page 1-26) generates CLKG and FSG.

The first divider stage of the sample rate generator creates the output clock from the input clock. This divider stage uses a counter that is preloaded with the divide down value in the CLKGDV bits of SRGR1. The output of this stage is the data clock (CLKG). CLKG has the frequency represented by the following equation.

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

Thus, the input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

Note:

The McBSP cannot operate at a frequency faster than one-half the LSPCLK. Choose an input clock frequency and a CLKDV value such that CLKG is less than or equal to one-half the LSPCLK.

1.4.1.4 Keeping CLKG Synchronized to an External Input Clock

When an external signal is selected to drive the sample rate generator (see section 1.4.1.1), the GSYNC bit in SRGR2 and the FSR pin can be used to configure the timing of the output clock (CLKG) relative to the input clock.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

For more details about the synchronization, see section 1.4.3 on page 1-31.

1.4.2 Frame Sync Generation in the Sample Rate Generator

The sample rate generator can produce a frame-sync signal (FSG) for use by the receiver, the transmitter, or both.

If you want the **receiver** to use FSG for frame synchronization, make sure FSRM = 1. (When FSRM = 0, receive frame synchronization is supplied via the FSR pin.)

If you want the **transmitter** to use FSG for frame synchronization, you must set:

- FSXM = 1 in PCR: This indicates that transmit frame synchronization is supplied by the McBSP itself rather than from the FSX pin.
- FSGM = 1 in SRGR2: This indicates that when FSXM = 1, transmit frame synchronization is supplied by the sample rate generator. (When FSGM = 0 and FSXM = 1, the transmitter uses frame-sync pulses generated every time data is transferred from DXR[1,2] to XSR[1,2].)

In either case, the sample rate generator must be enabled ($\overline{\text{GRST}}$ = 1) and the frame-sync logic in the sample rate generator must be enabled ($\overline{\text{FRST}}$ = 0).

1.4.2.1 Choosing the Width of the Frame-Sync Pulse on FSG

Each pulse on FSG has a programmable width. You program the FWID bits of SRGR1, and the resulting pulse width is (FWID + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

1.4.2.2 Controlling the Period Between the Starting Edges of Frame-Sync Pulses on FSG

You can control the amount of time from the starting edge of one FSG pulse to the starting edge of the next FSG pulse. This period is controlled in one of two ways, depending on the configuration of the sample rate generator:

- If the sample rate generator is using an external input clock and GSYNC = 1 in SRGR2, FSG pulses in response to an inactive-to-active transition on the FSR pin. Thus, the frame-sync period is controlled by an external device.
- Otherwise, you program the FPER bits of SRGR2, and the resulting frame-sync period is (FPER + 1) CLKG cycles, where CLKG is the output clock of the sample rate generator.

1.4.2.3 Keeping FSG Synchronized to an External Clock

When an external signal is selected to drive the sample rate generator (see section 1.4.1.1 on page 1-28), the GSYNC bit of SRGR2 and the FSR pin can be used to configure the timing of FSG pulses.

GSYNC = 1 ensures that the McBSP and an external device are dividing down the input clock with the same phase relationship. If GSYNC = 1, an inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and generation of FSG.

Section 1.4.3 has more details about the synchronization.

1.4.3 Synchronizing Sample Rate Generator Outputs to an External Clock

The sample rate generator can produce a clock signal (CLKG) and a frame-sync signal (FSG) based on an input clock signal that is either the CPU clock signal or a signal at the CLKS, CLKR, or CLKX pin. When an external clock is selected to drive the sample rate generator, the GSYNC bit of SRGR2 and the FSR pin can be used to control the timing of CLKG and the pulsing of FSG relative to the chosen input clock.

Make GSYNC = 1 when you want the McBSP and an external device to divide down the input clock with the same phase relationship. If GSYNC = 1:

- An inactive-to-active transition on the FSR pin triggers a resynchronization of CLKG and a pulsing of FSG.
- CLKG always begins with a high state after synchronization.
- FSR is always detected at the same edge of the input clock signal that generates CLKG, no matter how long the FSR pulse is.
- The FPER bits of SRGR2 are ignored because the frame-sync period on FSG is determined by the arrival of the next frame-sync pulse on the FSR pin.

If GSYNC = 0, CLKG runs freely and is not resynchronized, and the frame-sync period on FSG is determined by FPER.

1.4.3.1 Operating the Transmitter Synchronously With the Receiver

When GSYNC = 1, the transmitter can operate synchronously with the receiver, provided that:

- FSX is programmed to be driven by FSG (FSGM = 1 in SRGR2 and FSXM = 1 in PCR). If the input FSR has appropriate timing so that it can be sampled by the falling edge of CLKG, it can be used, instead, by setting FSXM = 0 and connecting FSR to FSX externally.
- The sample rate generator clock drives the transmit and receive clocking (CLKRM = CLKXM = 1 in PCR). Therefore, the CLK(R/X) pin should not be driven by any other driving source.

1.4.3.2 Synchronization Examples

Figure 1−21 and Figure 1−22 show the clock and frame-synchronization operation with various polarities of CLKS (the chosen input clock) and FSR. These figures assume FWID = 0 in SRGR1, for an FSG pulse that is 1 CLKG cycle wide. The FPER bits of SRGR2 are not programmed; the period from the start of a frame-sync pulse to the start of the next pulse is determined by the arrival of the next inactive-to-active transition on the FSR pin. Each of the figures shows what happens to CLKG when it is initially synchronized and GSYNC = 1, and when it is not initially synchronized and GSYNC = 1. The second figure has a slower CLKG frequency (it has a larger divide-down value in the CLKGDV bits of SRGR1).

Figure 1−21. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 1

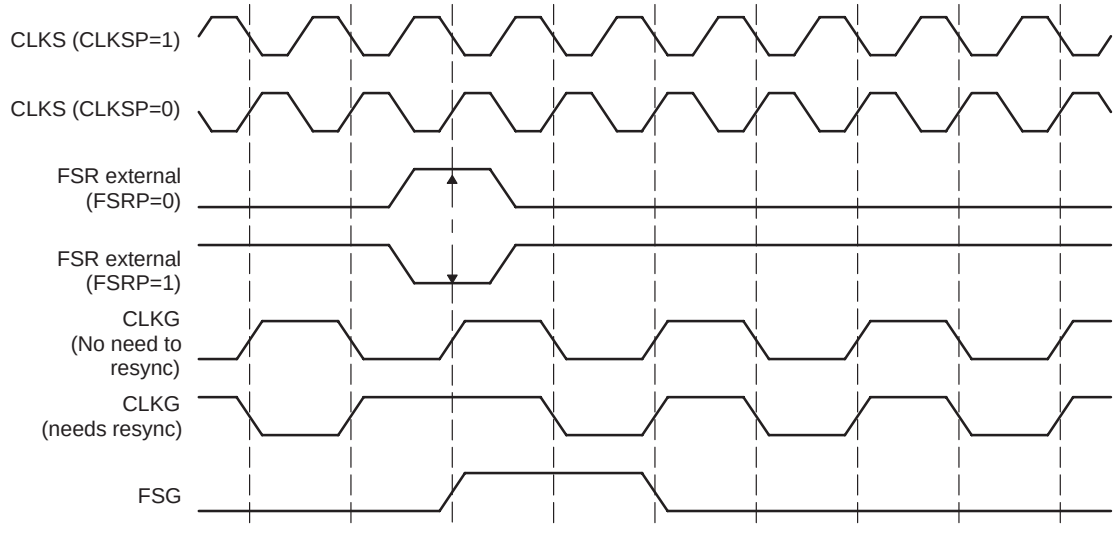
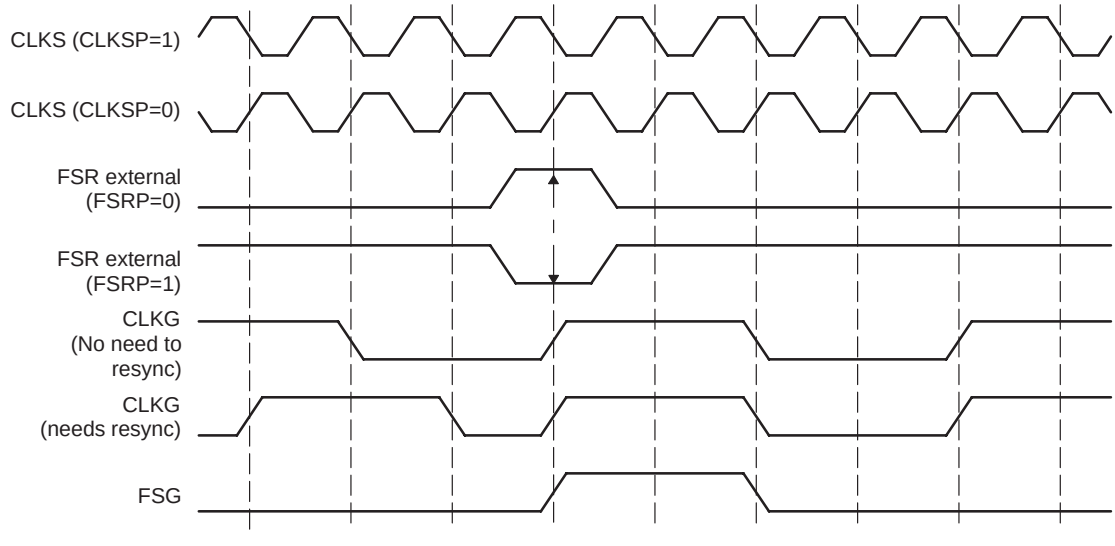


Figure 1−22. CLKG Synchronization and FSG Generation When GSYNC = 1 and CLKGDV = 3



1.4.4 Reset and Initialization Procedure for the Sample Rate Generator

To reset and initialize the sample rate generator:

1) Place the McBSP/sample rate generator in reset.

During a DSP reset, the sample rate generator, the receiver, and the transmitter reset bits ($\overline{\text{GRST}}$, $\overline{\text{RRST}}$, and $\overline{\text{XRST}}$) are automatically forced to 0. Otherwise, during normal operation, the sample rate generator can be reset by making $\overline{\text{GRST}}$ = 0 in SPCR2, provided that CLKG and/or FSG is not used by any portion of the McBSP. Depending on your system you may also want to reset the receiver ($\overline{\text{RRST}}$ = 0 in SPCR1) and reset the transmitter ($\overline{\text{XRST}}$ = 0 in SPCR2).

If $\overline{\text{GRST}}$ = 0 due to a DSP reset, CLKG is driven by the CPU clock divided by 2, and FSG is driven inactive-low. If $\overline{\text{GRST}}$ = 0 due to program code, CLKG and FSG are driven low (inactive).

2) Program registers that affect the sample rate generator.

Program the sample rate generator registers (SRGR1 and SRGR2) as required for your application. If necessary, other control registers can be loaded with desired values, provided the respective portion of the McBSP (the receiver or transmitter) is in reset.

After the sample rate generator registers are programmed, wait 2 CLKSRG cycles. This ensures proper synchronization internally.

3) Enable the sample rate generator (take it out of reset).

In SPCR2, make $\overline{\text{GRST}}$ = 1 to enable the sample rate generator.

After the sample rate generator is enabled, wait 2 CLKG cycles for the sample rate generator logic to stabilize.

On the next rising edge of CLKSRG, CLKG transitions to 1 and starts clocking with a frequency equal to

$$\text{CLKG frequency} = \frac{\text{Input clock frequency}}{(\text{CLKGDV} + 1)}$$

where the input clock is selected with the SCLKME bit of PCR and the CLKSM bit of SRGR2:

SCLKME	CLKSM	Input Clock For Sample Rate Generator
0	0	Reserved
0	1	LSPCLK
1	0	Signal on CLKR pin
1	1	Signal on CLKX pin

4) If necessary, enable the receiver and/or the transmitter.

If necessary, remove the receiver and/or transmitter from reset by setting $\overline{\text{RRST}}$ and/or XRST = 1.

5) If necessary, enable the frame-sync logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1/2] is loaded with data), set $\overline{\text{FRST}}$ = 1 in SPCR2 if an internally generated frame-sync pulse is required. FSG is generated with an active-high edge after the programmed number of CLKG clocks (FPER + 1) have elapsed.

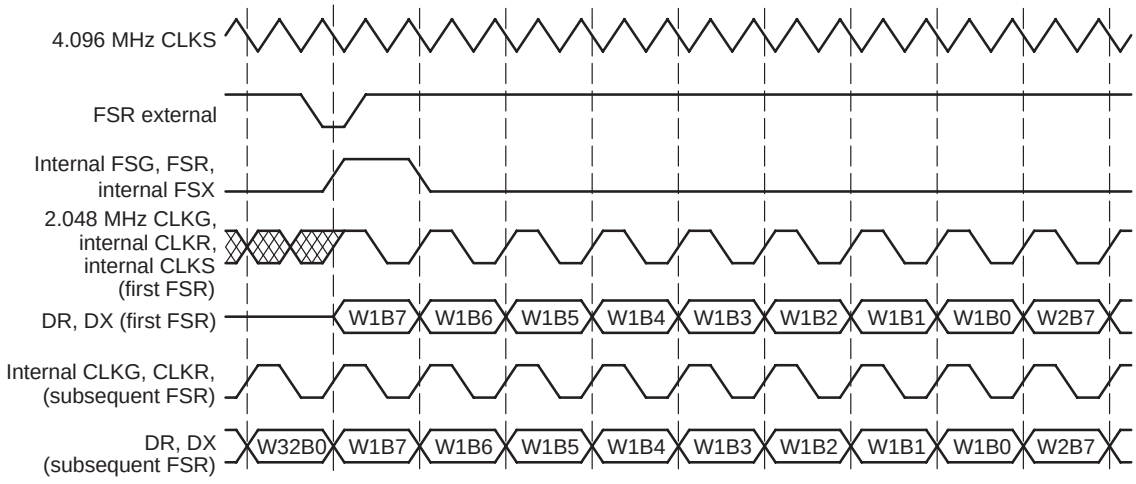
1.4.5 Sample Rate Generator Clocking Examples

This section shows three examples of using the sample rate generator to clock data during transmission and reception.

1.4.5.1 Double-Rate ST-Bus Clock

Figure 1−23 shows McBSP configuration to be compatible with the Mitel ST-Bus. Note that this operation is running at maximum frame frequency (described on page 1-17).

Figure 1−23. ST-BUS and MVIP Clocking Example



For this McBSP configuration:

- DLB = 0: Digital loopback mode off, CLKSTP = 00b: Clock stop mode off, and CLKRM/CLKXM = 1: Internal CLKR/CLKX generated internally by sample rate generator
- GSYNC = 1: Synchronize CLKG with external frame-sync signal input on FSR pin. CLKG is not synchronized until the frame-sync signal is active. FSR is regenerated internally to form a minimum pulse width.

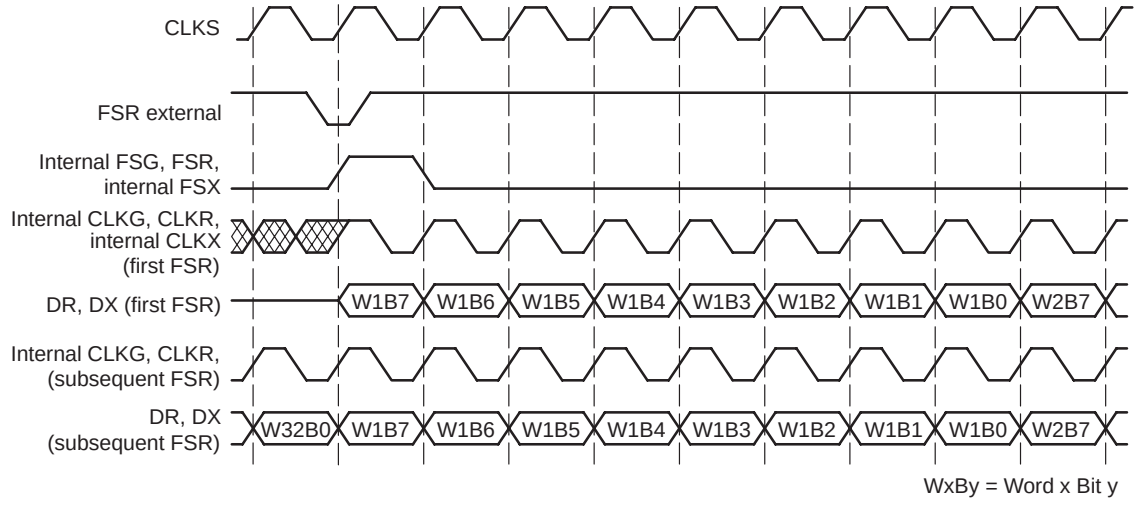
- ❑ SCLKME = 0 and CLKSM = 1: External clock signal at CLKS pin drives the sample rate generator
- ❑ CLKSP = 1: Falling edge of CLKS generates CLKG and thus internal CLK(R/X)
- ❑ CLKGDV = 1: Frequency of receive clock (shown as CLKR) is half CLKS frequency
- ❑ FSRP/FSXP = 1: Active-low frame-sync pulse
- ❑ RFRLEN1/XFRLEN1 = 11111b: 32 words per frame
- ❑ RWDLEN1/XWDLEN1 = 0: 8 bits per word
- ❑ RPHASE/XPHASE = 0: Single-phase frame and thus (R/X)FRLEN2 and (R/X)WDLEN2 are ignored
- ❑ RDATDLY/XDATDLY = 0: No data delay

1.4.5.2 Single-Rate ST-Bus Clock

The example in Figure 1−24 is the same as the double-rate ST-bus clock example in section 1.4.5.1 except that:

- ❑ CLKGDV = 0: CLKS drives internal CLK(R/X) without any divide down (single-rate clock).
- ❑ CLKSP = 0: Rising edge of CLKS generates CLKG and internal CLK(R/X)

Figure 1−24. Single-Rate Clock Example



The rising edge of CLKS is used to detect the external FSR pulse, which resynchronizes the internal McBSP clocks and generates a frame-sync pulse

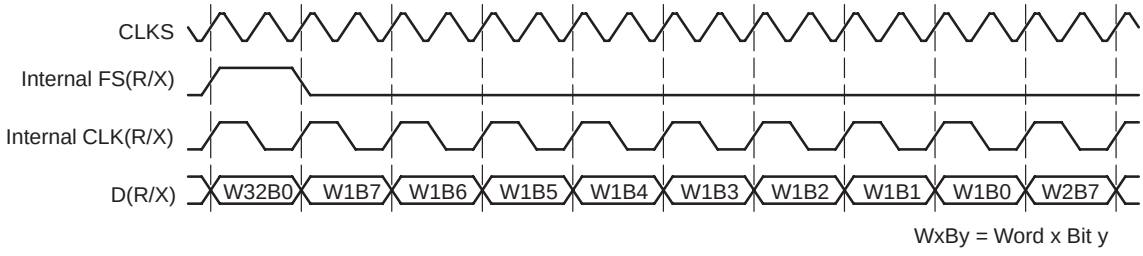
for internal use. The internal frame-sync pulse is generated so that it is wide enough to be detected on the falling edge of internal clocks.

1.4.5.3 Other Double-Rate Clock

The example in Figure 1−25 is the same as the double-rate ST-bus clock example in section 1.4.5.1 except that:

- ❑ CLKSP = 0: Rising edge of CLKS generates CLKG and thus CLK(R/X)
- ❑ CLKGDV = 1: Frequency of CLKG (and thus internal CLKR and internal CLKX) is half CLKS frequency
- ❑ FSRM/FSXM = 0: Frame synchronization is externally generated. The frame-sync pulse is wide enough to be detected.
- ❑ GSYNC = 0: CLKS drives CLKG. CLKG runs freely; it is not resynchronized by a pulse on the FSR pin.
- ❑ FSRP/FSXP = 0: Active-high input frame-sync signal
- ❑ RDATDLY/XDATDLY = 1: Data delay of one bit

Figure 1−25. Double-Rate Clock Example



1.5 McBSP Exception/Error Conditions

There are five serial port events that may constitute a system error:

- **Receiver Overrun (RFULL = 1)**. This occurs when DRR1 has not been read since the last RBR-to-DRR copy. Consequently, the receiver does not copy a new word from the RBR(s) to the DRR(s), and the RSR(s) are now full with another new word shifted in from DR. Therefore, RFULL = 1 indicates an error condition wherein any new data that may arrive at this time on DR will replace the contents of the RSR(s), and thus, the previous word is lost. The RSR(s) continue to be overwritten as long as new data arrives on DR and DRR1 is not read. For more details about overrun in the receiver, see page 1-39.

- **Unexpected Receive Frame-Sync Pulse (RSYNCERR = 1)**. This occurs during reception when RFIG = 0 and an unexpected frame-sync pulse occurs. An unexpected frame-sync pulse is one that begins the next frame transfer before all the bits of the current frame have been received. Such a pulse causes data reception to abort and restart. If new data has been copied into the RBR(s) from the RSR(s) since the last RBR-to-DRR copy, this new data in the RBR(s) is lost. This is because no RBR-to-DRR copy occurs; the reception has been restarted. For more details about receive frame-sync errors, see page 1-40.

- **Transmitter Data Overwrite**. This occurs when the CPU overwrites data in the DXR(s) before the data is copied to the XSR(s). The overwritten data never reaches the DX pin. For more details about overwrite in the transmitter, see page 1-43.

- **Transmitter Underflow ($\overline{\text{XEMPTY}}$ = 0)**. If a new frame-sync signal arrives before new data is loaded into DXR1, the previous data in the DXR(s) is sent again. This will continue for every new frame-sync pulse that arrives until DXR1 is loaded with new data. For more details about underflow in the transmitter, see page 1-44.

- **Unexpected Transmit Frame-Synch Pulse (XSYNCERR = 1)**. This occurs during transmission when XFIG = 0 and an unexpected frame-sync pulse occurs. An unexpected frame-sync pulse is one that begins the next frame transfer before all the bits of the current frame have been transferred. Such a pulse causes the current data transmission to abort and restart. If new data has been written to the DXR(s) since the last DXR-to-XSR copy, the current value in the XSR(s) is lost. For more details about transmit frame-sync errors, see page 1-46.

1.5.1 Overrun in the Receiver

RFULL = 1 in SPCR1 indicates that the receiver has experienced overrun and is in an error condition. RFULL is set when all of the following conditions are met:

1) DRR1 has not been read since the last RBR-to-DRR copy (RRDY = 1).

2) RBR1 is full and an RBR-to-DRR copy has not occurred.

3) RSR1 is full and an RSR1-to-RBR copy has not occurred.

As described in the section on McBSP reception (page 1-21), data arriving on DR is continuously shifted into RSR1 (for word length of 16 bits or smaller) or RSR2 and RSR1 (for word length larger than 16 bits). Once a complete word is shifted into the RSR(s), an RSR-to-RBR copy can occur only if the previous data in RBR1 has been copied to DRR1. The RRDY bit is set when new data arrives in DRR1 and is cleared when that data is read from DRR1. Until RRDY = 0, the next RBR-to-DRR copy will not take place, and the data is held in the RSR(s). New data arriving on the DR pin is shifted into RSR(s), and the previous content of the RSR(s) is lost.

You can prevent the loss of data if DRR1 is read no later than 2.5 cycles before the end of the third word is shifted into the RSR1.

Important: If both DRRs are needed (word length larger than 16 bits), the CPU must read from DRR2 first and then from DRR1. As soon as DRR1 is read, the next RBR-to-DRR copy is initiated. If DRR2 is not read first, the data in DRR2 is lost.

Note that after the receiver starts running from reset, a minimum of three words must be received before RFULL is set. Either of the following events clears the RFULL bit and allows subsequent transfers to be read properly:

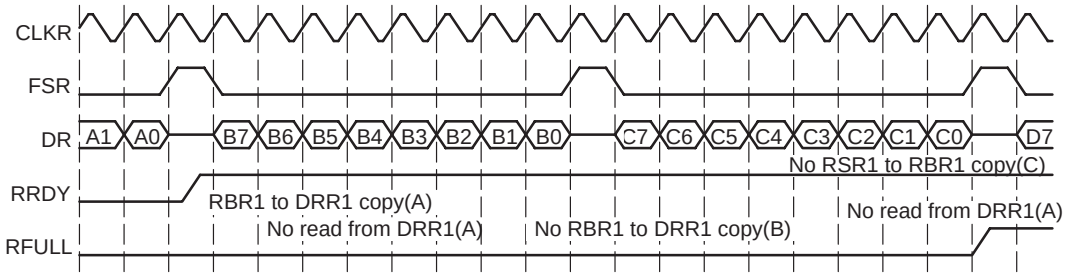
- The CPU reads DRR1.
- The receiver is reset individually ($\overline{\text{RRST}}$ = 0) or as part of a DSP reset.

Another frame-sync pulse is required to restart the receiver.

1.5.1.1 Example of the Overrun Condition

Figure 1−26 shows the receive overrun condition. Because serial word A is not read from DRR1 before serial word B arrives in RBR1, B is not transferred to DRR1 yet. Another new word (C) arrives and RSR1 is full with this data. DRR1 is finally read, but not earlier than 2.5 cycles before the end of word C. Therefore, new data (D) overwrites word C in RSR1. If DRR1 is not read in time, the next word can overwrite D.

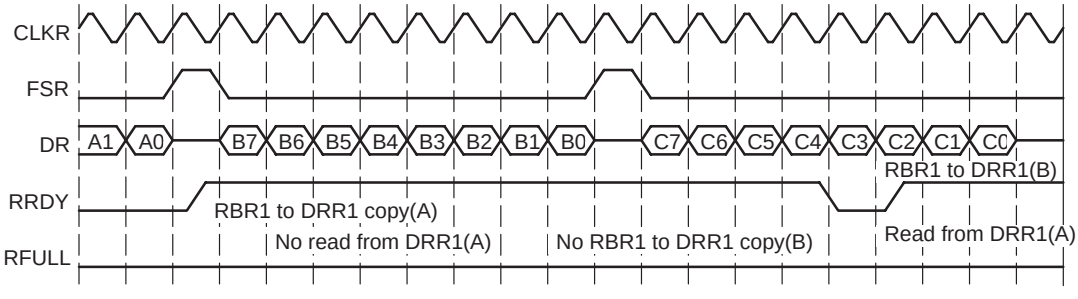
Figure 1−26. Overrun in the McBSP Receiver



1.5.1.2 Example of Preventing the Overrun Condition

Figure 1−27 shows the case where RFULL is set, but the overrun condition is prevented by a read from DRR1 at least 2.5 cycles before the next serial word (C) is completely shifted into RSR1. This ensures that an RBR1-to-DRR1 copy of word B occurs before receiver attempts to transfer word C from RSR1 to RBR1.

Figure 1−27. Overrun Prevented in the McBSP Receiver



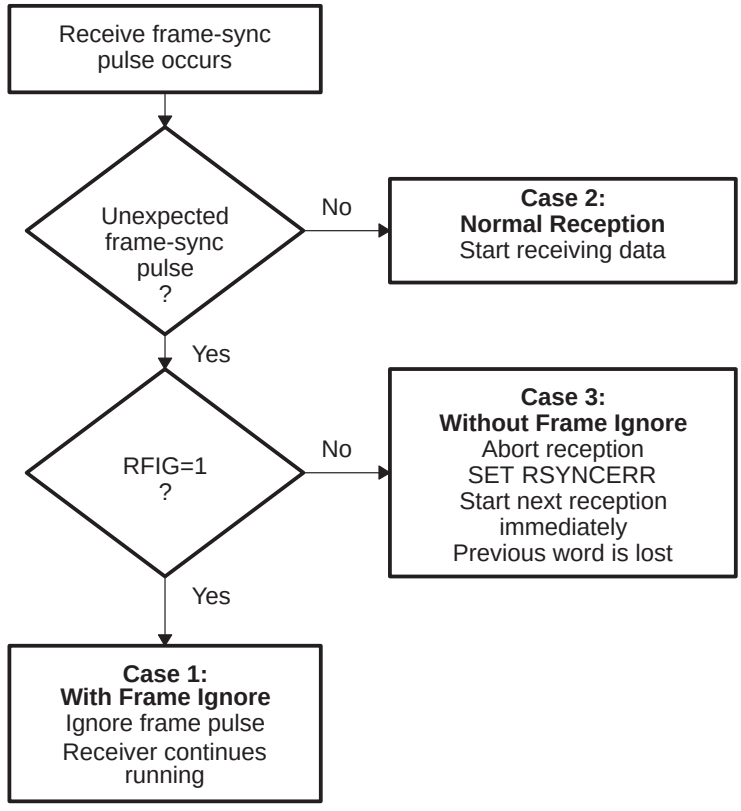
1.5.2 Unexpected Receive Frame-Sync Pulse

Section 1.5.2.1 shows how the McBSP responds to any receive frame-sync pulses, including an unexpected pulse. Sections 1.5.2.2 and 1.5.2.3 show an examples of a frame-sync error and an example of how to prevent such an error, respectively.

1.5.2.1 Possible Responses to Receive Frame-Sync Pulses

Figure 1−28 shows the decision tree that the receiver uses to handle all incoming frame-sync pulses. The figure assumes that the receiver has been started ($\overline{\text{RRST}}$ = 1 in SPCR1). Case 3 in the figure is the case in which an error occurs.

Figure 1−28. Possible Responses to Receive Frame-Sync Pulses



Any one of three cases can occur:

- **Case 1:** Unexpected internal FSR pulses with RFIG = 1 in RCR2. Receive frame-sync pulses are ignored, and the reception continues.

- **Case 2:** Normal serial port reception. Reception continues normally because the frame-sync pulse is not unexpected. There are three possible reasons why a receive operation might *not* be in progress when the pulse occurs:

  - The FSR pulse is the first after the receiver is enabled ($\overline{\text{RRST}}$ = 1 in SPCR1).

  - The FSR pulse is the first after DRR[1,2] is read, clearing a receiver full (RFULL = 1 in SPCR1) condition.

- The serial port is in the interpacket intervals. The programmed data delay for reception (programmed with the RDATDLY bits in RCR2) may start during these interpacket intervals for the first bit of the next word to be received. Thus, at maximum frame frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.

❑ **Case 3:** Unexpected receive frame synchronization with RFIG = 0 (frame-sync pulses not ignored). Unexpected frame-sync pulses can originate from an external source or from the internal sample rate generator.

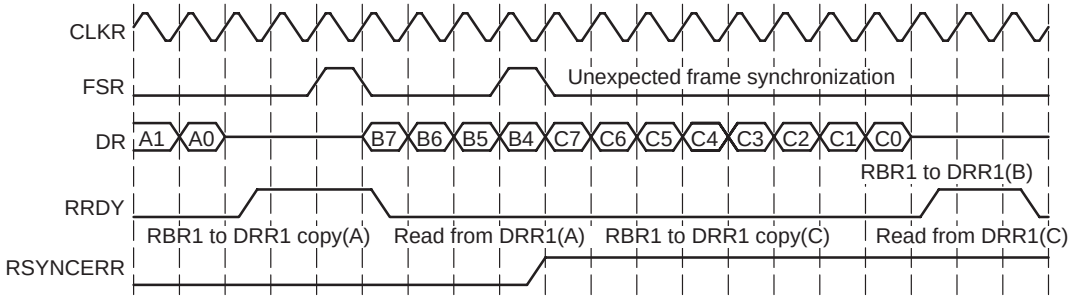
If a frame-sync pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-sync pulse, and the receiver sets the receive frame-sync error bit (RSYNCERR) in SPCR1. RSYNCERR can be cleared only by a receiver reset or by a write of 0 to this bit.

If you want the McBSP to notify the CPU of receive frame-sync errors, you can set a special receive interrupt mode with the RINTM bits of SPCR1. When RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU each time that RSYNCERR is set.

1.5.2.2 Example of an Unexpected Receive Frame-Sync Pulse

Figure 1−29 shows an unexpected receive frame-sync pulse during normal operation of the serial port, with time intervals between data packets. When the unexpected frame-sync pulse occurs, the RSYNCERR bit is set, the reception of data B is aborted, and the reception of data C begins. In addition, if RINTM = 11b, the McBSP sends a receive interrupt (RINT) request to the CPU.

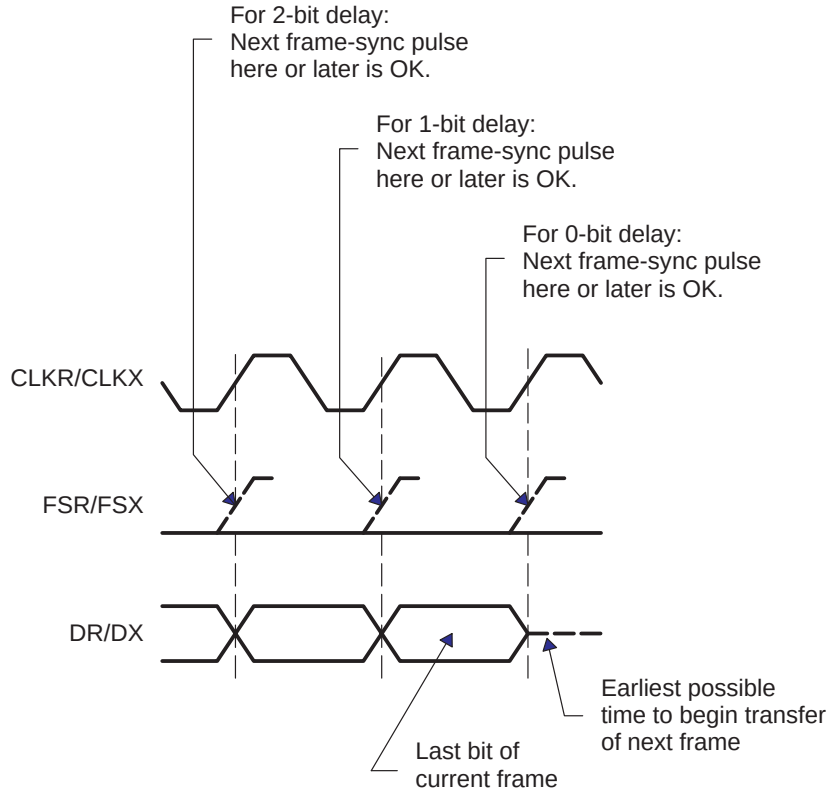
Figure 1−29. An Unexpected Frame-Sync Pulse During a McBSP Reception



1.5.2.3 Preventing Unexpected Receive Frame-Sync Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKR cycles, depending on the value in the RDATDLY bits of RCR2. For each possible data delay, Figure 1−30 shows when a new frame-sync pulse on FSR can safely occur relative to the last bit of the current frame.

Figure 1−30. Proper Positioning of Frame-Sync Pulses



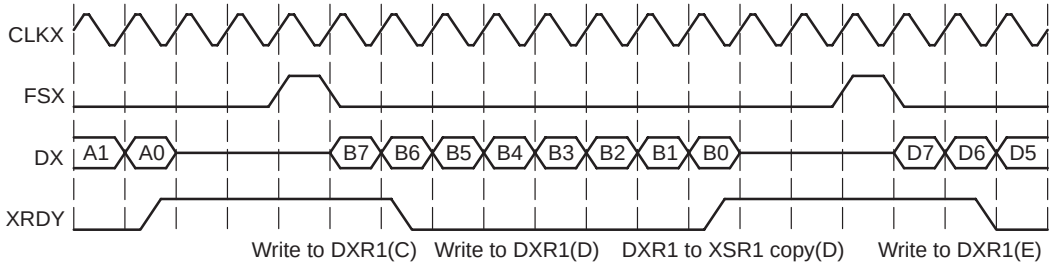
1.5.3 Overwrite in the Transmitter

As described in the section on McBSP transmission (page 1-22), after the CPU writes data to the DXR(s), the transmitter must then copy that data to the XSR(s) and then shift each bit from the XSR(s) to the DX pin. If new data is written to the DXR(s) before the previous data is copied to the XSR(s), the previous data in the DXR(s) is overwritten and thus lost.

1.5.3.1 Example of the Overwrite Condition

Figure 1−31 shows what happens if the data in DXR1 is overwritten before being transmitted. Initially, DXR1 is loaded with data C. A subsequent write to DXR1 overwrites C with D before C is copied to XSR1. Thus, C is never transmitted on DX.

Figure 1−31. Data in the McBSP Transmitter Overwritten and Thus Not Transmitted



1.5.3.2 Preventing Overwrites

You can prevent CPU overwrites by making the CPU:

- ❏ Poll for XRDY = 1 in SPCR2 before writing to the DXR(s). XRDY is set when data is copied from DXR1 to XSR1 and is cleared when new data is written to DXR1.
- ❏ Wait for a transmit interrupt (XINT) before writing to the DXR(s). When XINTM = 00b in SPCR2, the transmitter sends XINT to the CPU each time XRDY is set.

1.5.4 Underflow in the Transmitter

The McBSP indicates a transmitter empty (or underflow) condition by clearing the $\overline{\text{XEMPTY}}$ bit in SPCR2. Either of the following events activates $\overline{\text{XEMPTY}}$ ($\overline{\text{XEMPTY}}$ = 0):

- ❏ DXR1 has not been loaded since the last DXR-to-XSR copy, and all bits of the data word in the XSR(s) have been shifted out on the DX pin.
- ❏ The transmitter is reset (by forcing $\overline{\text{XRST}}$ = 0 in SPCR2, or by a DSP reset) and is then restarted.

In the underflow condition, the transmitter continues to transmit the old data that is in the DXR(s) for every new transmit frame-sync signal until a new value is loaded into DXR1 by the CPU.

Note:

If both DXRs are needed (word length larger than 16 bits), the CPU must load DXR2 first and then load DXR1. As soon as DXR1 is loaded, the contents of both DXRs are copied to the transmit shift registers (XSRs). If DXR2 is not loaded first, the previous content of DXR2 is passed to the XSR2.

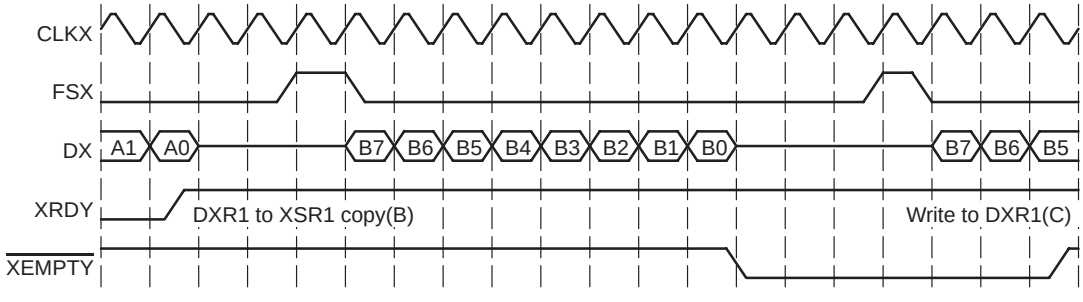
$\overline{\text{XEMPTY}}$ is deactivated ($\overline{\text{XEMPTY}}$ = 1) when a new word in DXR1 is transferred to XSR1. If FSXM = 1 in PCR and FSGM = 0 in SRGR2, the transmitter generates a single internal FSX pulse in response to a DXR-to-XSR copy. Otherwise, the transmitter waits for the next frame-sync pulse before sending out the next frame on DX.

When the transmitter is taken out of reset ($\overline{\text{XRST}}$ = 1), it is in a transmitter ready (XRDY = 1 in SPCR2) and transmitter empty ($\overline{\text{XEMPTY}}$ = 0) state. If DXR1 is loaded by the CPU before internal FSX goes active high, a valid DXR-to-XSR transfer occurs. This allows for the first word of the first frame to be valid even before the transmit frame-sync pulse is generated or detected. Alternatively, if a transmit frame-sync pulse is detected before DXR1 is loaded, zeros will be output on DX.

1.5.4.1 Example of the Underflow Condition

Figure 1−32 shows an underflow condition. After B is transmitted, DXR1 is not reloaded before the subsequent frame-sync pulse. Thus, B is again transmitted on DX.

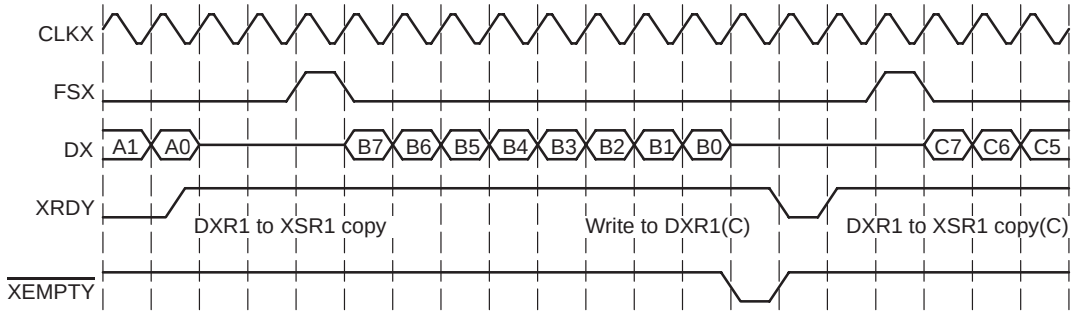
Figure 1−32. Underflow During McBSP Transmission



1.5.4.2 Example of Preventing the Underflow Condition

Figure 1−33 shows the case of writing to DXR1 just before an underflow condition would otherwise occur. After B is transmitted, C is written to DXR1 before the next frame-sync pulse. As a result, there is no underflow; B is not transmitted twice.

Figure 1−33. Underflow Prevented in the McBSP Transmitter



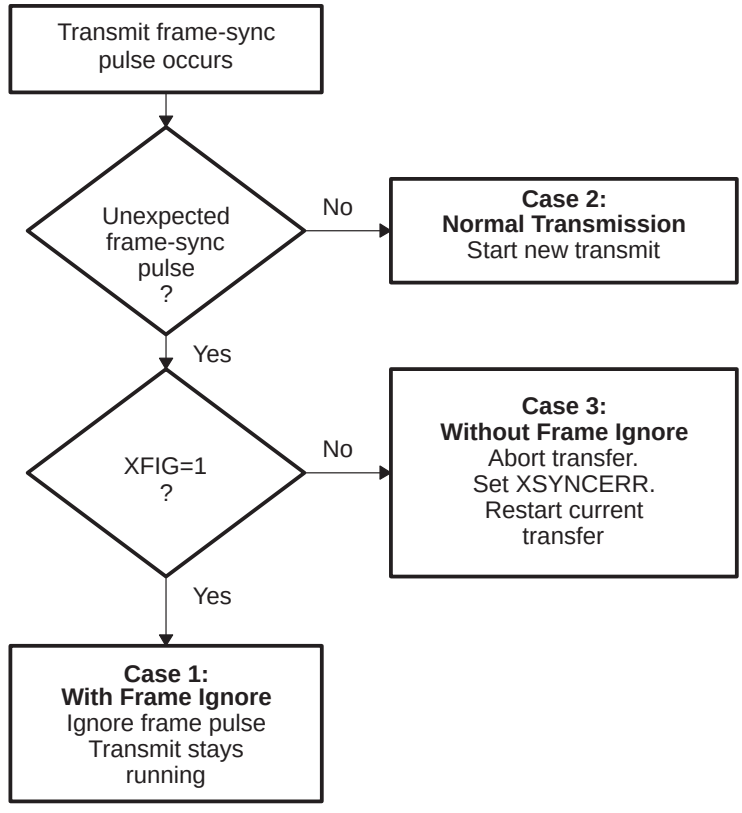
1.5.5 Unexpected Transmit Frame-Sync Pulse

Section 1.5.5.1 shows how the McBSP responds to any transmit frame-sync pulses, including an unexpected pulse. Sections 1.5.5.2 and 1.5.5.3 show an examples of a frame-sync error and an example of how to prevent such an error, respectively.

1.5.5.1 Possible Responses to Transmit Frame-Sync Pulses

Figure 1−34 shows the decision tree that the transmitter uses to handle all incoming frame-sync pulses. The figure assumes that the transmitter has been started ($\overline{\text{XRST}}$ = 1 in SPCR2). Case 3 in the figure is the case in which an error occurs.

Figure 1−34. Possible Responses to Transmit Frame-Sync Pulses



Any one of three cases can occur:

- **Case 1:** Unexpected internal FSX pulses with XFIG = 1 in XCR2. Transmit frame-sync pulses are ignored, and the transmission continues.

- **Case 2:** Normal serial port transmission. Transmission continues normally because the frame-sync pulse is not unexpected. There are two possible reasons why a transmit operations might *not* be in progress when the pulse occurs:

  This FSX pulse is the first after the transmitter is enabled ($\overline{\text{XRST}}$ = 1).

  The serial port is in the interpacket intervals. The programmed data delay for transmission (programmed with the XDATDLY bits of XCR2) may start during these interpacket intervals before the first bit of the previous word is transmitted. Thus, at maximum packet frequency, frame synchronization can still be received 0 to 2 clock cycles before the first bit of the synchronized frame.

- **Case 3:** Unexpected transmit frame synchronization with XFIG = 0 (frame-sync pulses not ignored). Unexpected frame-sync pulses can originate from an external source or from the internal sample rate generator.

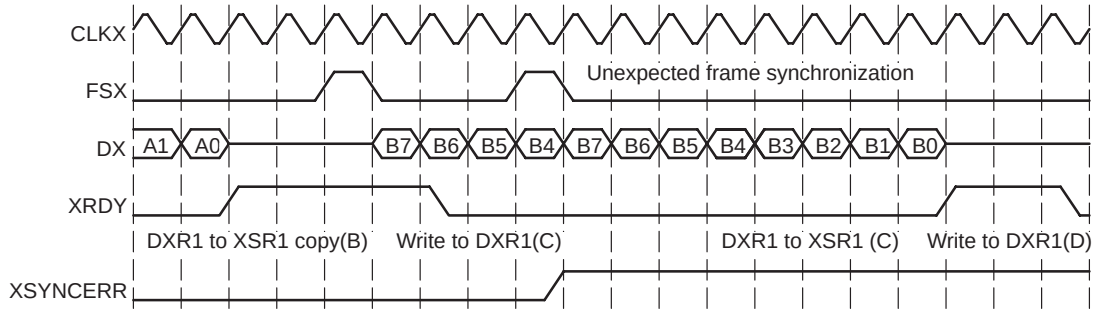
  If a frame-sync pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-sync pulse, and the transmitter sets the transmit frame-sync error bit (XSYNCERR) in SPCR2. XSYNCERR can be cleared only by a transmitter reset or by a write of 0 to this bit.

  If you want the McBSP to notify the CPU of frame-sync errors, you can set a special transmit interrupt mode with the XINTM bits of SPCR2. When XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU each time that XSYNCERR is set.

1.5.5.2 Example of an Unexpected Transmit Frame-Sync Pulse

Figure 1−35 shows an unexpected transmit frame-sync pulse during normal operation of the serial port, with intervals between the data packets. When the unexpected frame-sync pulse occurs, the XSYNCERR bit is set and because no new data has been passed to XSR1 yet, the transmission of data B is restarted. In addition, if XINTM = 11b, the McBSP sends a transmit interrupt (XINT) request to the CPU.

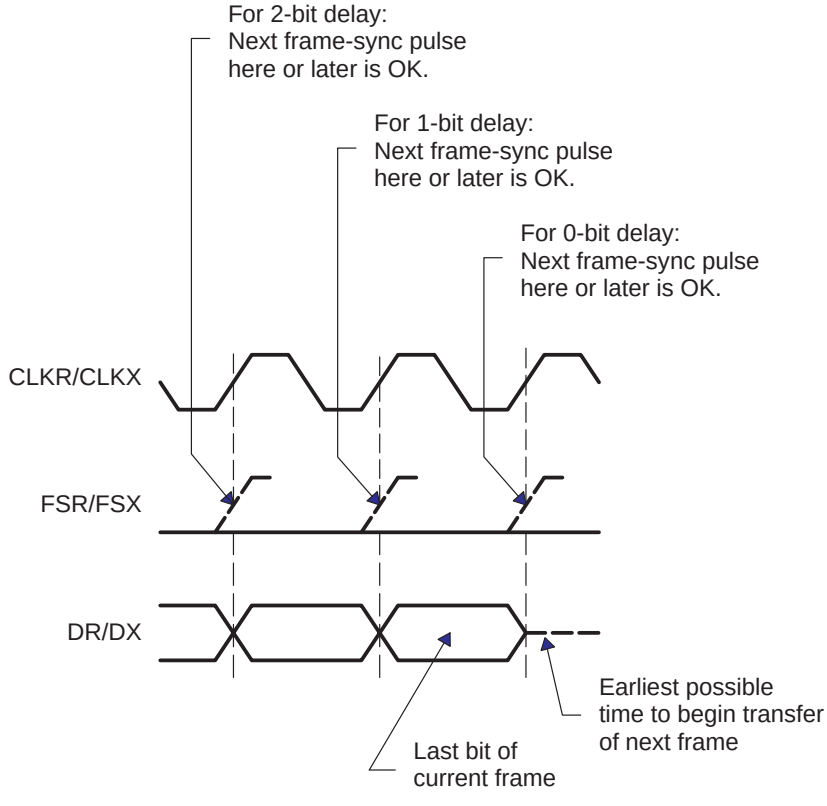
Figure 1−35. An Unexpected Frame-Sync Pulse During a McBSP Transmission



1.5.5.3 Preventing Unexpected Transmit Frame-Sync Pulses

Each frame transfer can be delayed by 0, 1, or 2 CLKX cycles, depending on the value in the XDATDLY bits of XCR2. For each possible data delay, Figure 1−36 shows when a new frame-sync pulse on FSX can safely occur relative to the last bit of the current frame.

Figure 1−36. Proper Positioning of Frame-Sync Pulses



Chapter 2

Multichannel Selection Modes

This chapter explains how to assign blocks to partitions for multichannel selection.

Topic **Page**

2.1 Channels, Blocks, and Partitions

A McBSP **channel** is a time slot for shifting in/out the bits of one serial word. Each McBSP supports up to 128 channels for reception and 128 channels for transmission.

In the receiver and in the transmitter, the 128 available channels are divided into eight **blocks** that each contain 16 contiguous channels:

Block 0: Channels 0–15	Block 4: Channels 64–79
Block 1: Channels 16–31	Block 5: Channels 80–95
Block 2: Channels 32–47	Block 6: Channels 96–111
Block 3: Channels 48–63	Block 7: Channels 112–127

The blocks are assigned to **partitions** according to the selected partition mode. In the 2-partition mode, you assign one even-numbered block (0, 2, 4, or 6) to partition A and one odd-numbered block (1, 3, 5, or 7) to partition B. In the 8-partition mode, blocks 0 through 7 are automatically assigned to partitions, A through H, respectively.

The number of partitions for reception and the number of partitions for transmission are independent. For example, it is possible to use 2 receive partitions (A and B) and 8 transmit partitions (A–H).

2.1.1 Multichannel Selection

When a McBSP uses a time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices, the McBSP may need to receive and/or transmit on only a few channels. To save memory and bus bandwidth, you can use a multichannel selection mode to prevent data flow in some of the channels.

Each channel partition has a dedicated channel-enable register. If the appropriate multichannel selection mode is on, each bit in the register controls whether data flow is allowed or prevented in one of the channels that is assigned to that partition.

The McBSP has one receive multichannel selection mode (described in section 2.1.5) and three transmit multichannel selection modes (described in section 2.1.6).

2.1.2 Configuring a Frame for Multichannel Selection

Before you enable a multichannel selection mode, make sure you properly configure the data frame:

- ❑ Select a single-phase frame (RPHASE/XPHASE = 0). Each frame represents a TDM data stream.

- Set a frame length (in RFRLEN1/XFRLEN1) that includes the highest-numbered channel that will be used. For example, if you plan to use channels 0, 15, and 39 for reception, the receive frame length must be at least 40 (RFRLEN1 = 39). If XFRLEN1 = 39 in this case, the receiver creates 40 time slots per frame but only receives data during time slots 0, 15, and 39 of each frame.

2.1.3 Using Two Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use two partitions or eight partitions. If you choose the 2-partition mode (RMCME = 0 for reception, XMCME = 0 for transmission), McBSP channels are activated using an alternating scheme. In response to a frame-sync pulse, the receiver or transmitter begins with the channels in partition A and then alternates between partitions B and A until the complete frame has been transferred. When the next frame-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A.

2.1.3.1 Assigning Blocks to Partitions A and B

For reception, any two of the eight receive-channel blocks can be assigned to receive partitions A and B, which means up to 32 receive channels can be enabled at any given point in time. Similarly, any two of the eight transmit-channel blocks (up 32 enabled transmit channels) can be assigned to transmit partitions A and B.

For reception:

- Assign an even-numbered channel block (0, 2, 4, or 6) to receive partition A by writing to the RPABLK bits. In the receive multichannel selection mode (described in section 2.1.5), the channels in this partition are controlled by receive channel enable register A (RCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to receive partition B with the RPBBLK bits. In the receive multichannel selection mode, the channels in this partition are controlled by receive channel enable register B (RCERB).

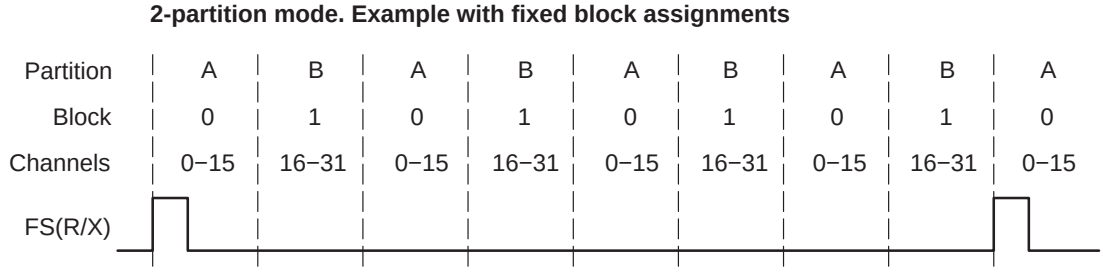
For transmission:

- Assign an even-numbered channel block (0, 2, 4, or 6) to transmit partition A by writing to the XPABLK bits. In one of the transmit multichannel selection modes (described in section 2.1.6), the channels in this partition are controlled by transmit channel enable register A (XCERA).
- Assign an odd-numbered block (1, 3, 5, or 7) to transmit partition B with the XPBBLK bits. In one of the transmit multichannel selection modes, the

channels in this partition are controlled by transmit channel enable register B (XCERB).

Figure 2−1 shows an example of alternating between the channels of partition A and the channels of partition B. Channels 0–15 have been assigned to partition A, and channels 16–31 have been assigned to partition B. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then alternates between partitions B and A until the complete frame is transferred.

Figure 2−1. Alternating Between the Channels of Partition A and the Channels of Partition B



As explained next, you can dynamically change which blocks of channels are assigned to the partitions.

2.1.3.2 Reassigning Blocks During Reception/Transmission

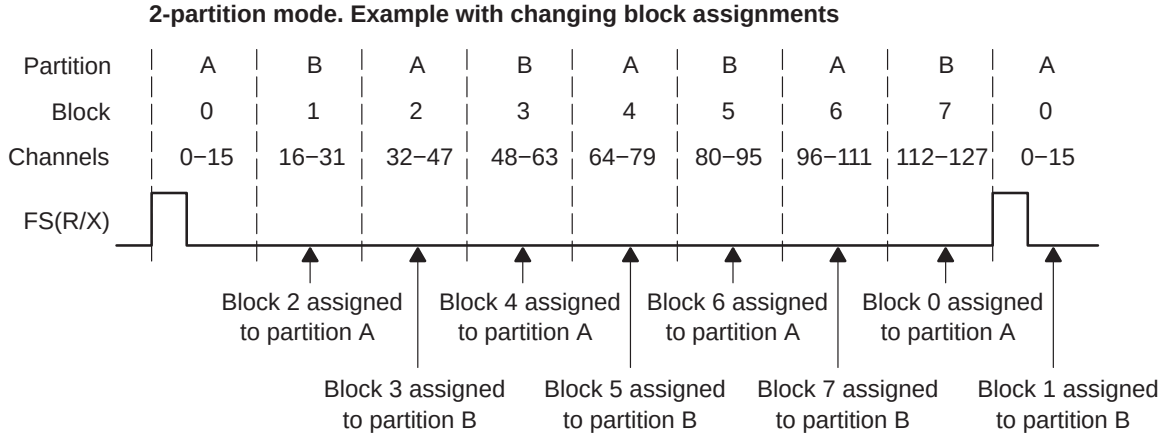
If you want to use more than 32 channels, you can change which channel blocks are assigned to partitions A and B during the course of a data transfer. However, these changes must be carefully timed. While a partition is being transferred, its the associated block assignment bits cannot be modified, and its associated channel enable register cannot be modified. For example, if block 3 is being transferred and block 3 is assigned to partition A, you cannot modify (R/X)PABLK to assign different channels to partition A, and you cannot modify (R/X)CERA to change the channel configuration for partition A. Several features of the McBSP help you time the reassignment:

- The block of channels currently involved in reception/transmission (the current block) is reflected in the RCBLK/XCBLK bits. Your program can poll these bits to determine which partition is active. When a partition is not active, it is safe to change its block assignment and channel configuration.
- At the end of every block (at the boundary of two partitions), an interrupt can be sent to the CPU. In response to the interrupt, the CPU can then check the RCBLK/XCBLK bits and update the inactive partition.

Figure 2−2 shows an example of reassigning channels throughout a data transfer. In response to a frame-sync pulse, the McBSP alternates between

partitions A and B. Whenever partition B is active, the CPU changes the block assignment for partition A. Whenever, partition A is active, the CPU changes the block assignment for partition B.

Figure 2−2. Reassigning Channel Blocks Throughout a McBSP Data Transfer



2.1.4 Using Eight Partitions

For multichannel selection operation in the receiver and/or the transmitter, you can use eight partitions or two partitions. If you choose the 8-partition mode (RMCME = 1 for reception, XMCME = 1 for transmission), McBSP channels are activated in the following order: A, B, C, D, E, F, G, H. In response to a frame-sync pulse, the receiver or transmitter begins with the channels in partition A and then continues with the other partitions in order until the complete frame has been transferred. When the next frame-sync pulse occurs, the next frame is transferred, beginning with the channels in partition A.

In the 8-partition mode, the (R/X)PABLK and (R/X)PBBLK bits are ignored and the 16-channel blocks are assigned to the partitions as shown in Table 2−1 and Table 2−2. These assignments cannot be changed. The tables also show the registers used to control the channels in the partitions.

Table 2−1. Receive Channel Assignment and Control When Eight Receive Partitions Are Used

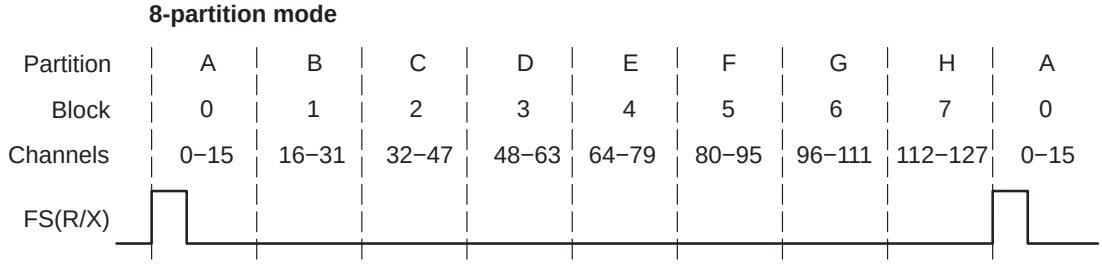
Receive Partition	Assigned Block of Receive Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	RCERA
B	Block 1: channels 16 through 31	RCERB
C	Block 2: channels 32 through 47	RCERC
D	Block 3: channels 48 through 63	RCERD
E	Block 4: channels 64 through 79	RCERE
F	Block 5: channels 80 through 95	RCERF
G	Block 6: channels 96 through 111	RCERG
H	Block 7: channels 112 through 127	RCERH

Table 2−2. Transmit Channel Assignment and Control When Eight Transmit Partitions Are Used

Transmit Partition	Assigned Block of Transmit Channels	Register Used For Channel Control
A	Block 0: channels 0 through 15	XCERA
B	Block 1: channels 16 through 31	XCERB
C	Block 2: channels 32 through 47	XCERC
D	Block 3: channels 48 through 63	XCERD
E	Block 4: channels 64 through 79	XCERE
F	Block 5: channels 80 through 95	XCERF
G	Block 6: channels 96 through 111	XCERG
H	Block 7: channels 112 through 127	XCERH

Figure 2−3 shows an example of the McBSP using the 8-partition mode. In response to a frame-sync pulse, the McBSP begins a frame transfer with partition A and then activates B, C, D, E, F, G, and H to complete a 128-word frame.

Figure 2−3. McBSP Data Transfer in the 8-Partition Mode



2.1.5 Receive Multichannel Selection Mode

The RMCM bit of MCR1 determines whether all channels or only selected channels are enabled for reception. When RMCM = 0, all 128 receive channels are enabled and cannot be disabled. When **RMCM = 1**, the receive multichannel selection mode is enabled. In this mode:

- Channels can be individually enabled or disabled. The only channels enabled are those selected in the appropriate receive channel enable registers (RCERs). The way channels are assigned to the RCERs depends on the number of receive channel partitions (2 or 8), as defined by the RMCME bit of MCR1.
- If a receive channel is disabled, any bits received in that channel are passed only as far as the receive buffer register(s) (RBR(s)). The receiver does not copy the content of the RBR(s) to the DRR(s), and as a result, does not set the receiver ready bit (RRDY). Therefore, no receive FIFO event (REVT) is generated, and if the receiver interrupt mode depends on RRDY (RINTM = 00b), no interrupt is generated.

As an example of how the McBSP behaves in the receive multichannel selection mode, suppose you enable only channels 0, 15, and 39 and that the frame length is 40. The McBSP:

1) Accepts bits shifted in from the DR pin in channel 0
2) Ignores bits received in channels 1–14
3) Accepts bits shifted in from the DR pin in channel 15
4) Ignores bits received in channels 16–38
5) Accepts bits shifted in from the DR pin in channel 39

2.1.6 Transmit Multichannel Selection Modes

The XMCM bits of XCR2 determine whether all channels or only selected channels are enabled and unmasked for transmission. More details on enabling and masking are in section 2.1.7. The McBSP has three transmit multichannel selection modes (XMCM = 01b, XMCM = 10b, and XMCM = 11b), which are described in the following table:

Table 2−3. Selecting a Transmit Multichannel Selection Mode With the XMCM Bits

XMCM	Transmit Multichannel Selection Mode
00b	No transmit multichannel selection mode is on. All channels are enabled and unmasked. No channels can be disabled or masked.
01b	All channels are disabled unless they are selected in the appropriate transmit channel enable registers (XCERs) If enabled, a channel in this mode is also unmasked. The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
10b	All channels are enabled, but they are masked unless they are selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in XCERs.
11b	This mode is used for symmetric transmission and reception. All channels are disabled for transmission unless they are enabled for reception in the appropriate receive channel enable registers (RCERs). Once enabled, they are masked unless they are also selected in the appropriate transmit channel enable registers (XCERs). The XMCME bit of MCR2 determines whether 32 channels or 128 channels are selectable in RCERs and XCERs.

As an example of how the McBSP behaves in a transmit multichannel selection mode, suppose that XMCM = 01b (all channels disabled unless individually enabled) and that you have enabled only channels 0, 15, and 39. Suppose also that the frame length is 40. The McBSP …

1) Shifts data to the DX pin in channel 0
2) Places the DX pin in the high impedance state in channels 1–14
3) Shifts data to the DX pin in channel 15
4) Places the DX pin in the high impedance state in channels 16–38
5) Shifts data to the DX pin in channel 39

2.1.7 Disabling/Enabling Versus Masking/Unmasking

For transmission, a channel can be:

- ❑ Enabled and unmasked (transmission can begin and can be completed)
- ❑ Enabled but masked (transmission can begin but cannot be completed)
- ❑ Disabled (transmission cannot occur)

The following definitions explain the channel control options:

Enabled channel — A channel that can begin transmission by passing data from the data transmit register(s) (DXR(s)) to the transmit shift registers (XSR(s)).

Masked channel — A channel that cannot complete transmission. The DX pin is held in the high impedance state; data cannot be shifted out on the DX pin.

In systems where symmetric transmit and receive provides software benefits, this feature allows transmit channels to be disabled on a shared serial bus. A similar feature is not needed for reception because multiple receptions cannot cause serial bus contention.

Disabled channel — A channel that is not enabled. A disabled channel is also masked.

Because no DXR-to-XSR copy occurs, the XRDY bit of SPCR2 is not set. Therefore, no transmit FIFO event (XEVT) is generated, and if the transmit interrupt mode depends on XRDY (XINTM = 00b in SPCR2), no interrupt is generated.

The $\overline{\text{XEMPTY}}$ bit of SPCR2 is not affected.

Unmasked channel — A channel that is not masked. Data in the XSR(s) is shifted out on the DX pin.

2.1.8 Activity on McBSP Pins for Different Values of XMCM

Figure 2−4 shows the activity on the McBSP pins for the various XMCM values. In all cases, the transmit frame is configured as follows:

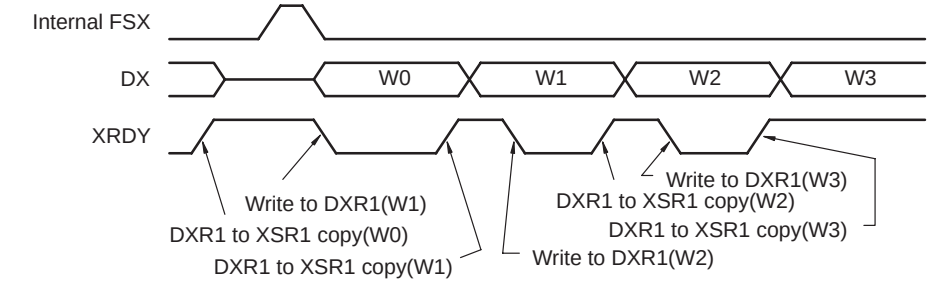
- XPHASE = 0: Single-phase frame (required for multichannel selection modes)
- XFRLEN1 = 0000011b: 4 words per frame
- XWDLEN1 = 000b: 8 bits per word
- XMCME = 0: 2-partition mode (only partitions A and B used)

In the case where XMCM = 11b, transmission and reception are symmetric, which means the corresponding bits for the receiver (RPHASE, RFRLEN1, RWDLEN1, and RMCME) must have the same values as XPHASE, XFRLEN1, and XWDLEN1, respectively.

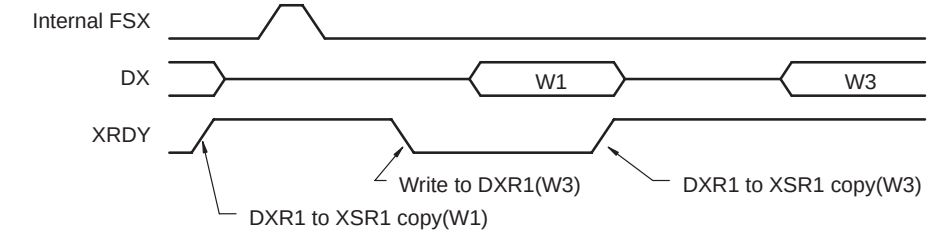
In the figure, the arrows showing where the various events occur are only sample indications. Wherever possible, there is a time window in which these events can occur.

Figure 2−4. Activity on McBSP Pins for the Possible Values of XMCM

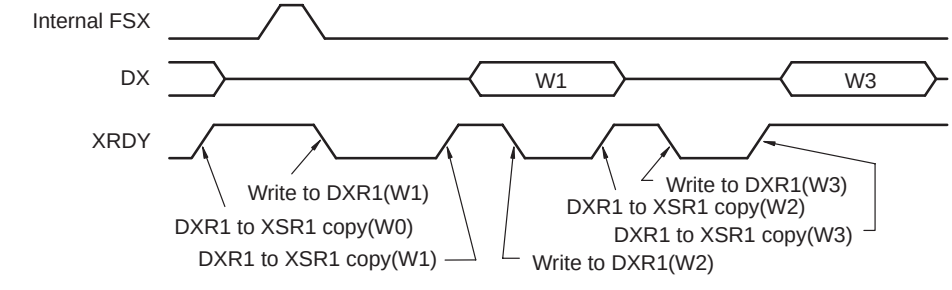
(a) XMCM = 00b: All channels enabled and unmasked



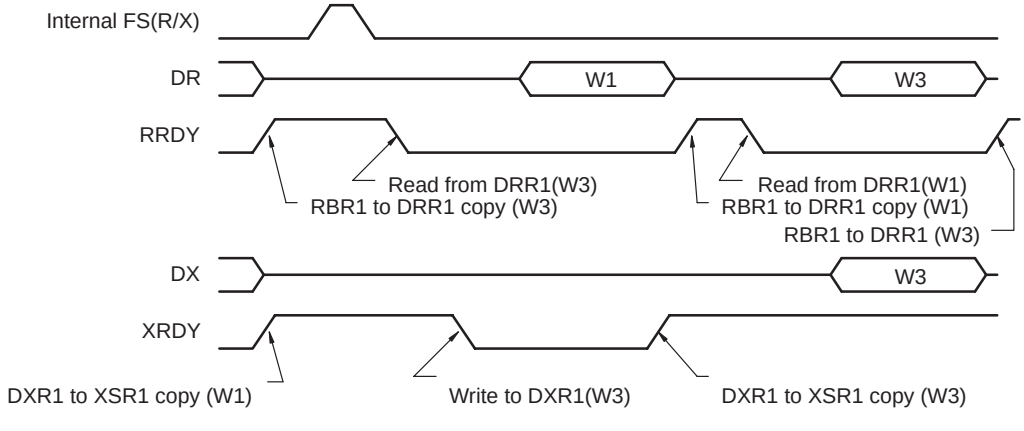
(b) XMCM = 01b, XPABLK = 00b, XCERA = 1010b: Only channels 1 and 3 enabled and unmasked



(c) XMCM = 10b, XPABLK = 00b, XCERA = 1010b: All channels enabled, only 1 and 3 unmasked



(d) XMCM = 11b, RPABLK = 00b, XPABLK = X, RCERA = 1010b, XCERA = 1000b:
Receive channels: 1 and 3 enabled; transmit channels: 1 and 3 enabled, but only 3 unmasked



2.1.9 Using Interrupts Between Block Transfers

When a multichannel selection mode is used, an interrupt request can be sent to the CPU at the end of every 16-channel block (at the boundary between partitions and at the end of the frame). In the receive multichannel selection mode, a receive interrupt (RINT) request is generated at the end of each block transfer if RINTM = 01b. In any of the transmit multichannel selection modes, a transmit interrupt (XINT) request is generated at the end of each block transfer if XINTM = 01b. When RINTM/XINTM = 1b, no interrupt is generated unless a multichannel selection mode is on.

This type of interrupt is especially helpful if you are using the 2-partition mode and you want to know when you can assign a different block of channels to partition A or B.

2.2 A-bis Mode

In the A-bis mode (ABIS = 1 in SPCR1), the McBSP can receive and transmit up to 1024 bits on a PCM link. The receive section can extract all 1024 bits from a 1024-bit PCM frame according to a given bit-enable pattern, and generate an interrupt to the CPU when 16 enabled bits have been compacted into a word in DRR1, or when a receive frame is complete. In addition, the transmit section can expand up to 1024 bits into a 1024-bit PCM frame at a specific position, according to a given bit-enable pattern, and generate an interrupt when 16 enabled bits have been transmitted or a transmit frame is complete.

The bit-enable patterns are specified with channel enable registers A and B (RCERA and RCERB for reception, XCERA and XCERB for transmission). These registers have a different function than in the multichannel selection modes (described in section 2.1). Instead of indicating which channels will be enabled, these registers indicate which bits in the data stream will be enabled. A 1 in a given position in the (R/X)CER(A/B) register enables a corresponding bit in the receive/transmit data stream.

The A-bis mode requires a word length of 16 bits (for reception: RWDLEN1 = 010b in RCR1, for transmission: XWDLEN1 = 010b in XCR1). Otherwise, operation in the A-bis mode is undetermined.

2.2.1 A-bis Mode Receive Operation

In the A-bis mode, bits that are not enabled in the RCERA and RCERB registers are ignored and are not compacted in the receiver. Bits that are enabled are received and compacted. When 16 enabled bits have been received, the received word is copied from RSR1 to DRR1 and the McBSP generates an interrupt to the CPU. RCERA and RCERB alternate specifying the receive masking pattern for each of the 16 receive clocks. Figure 2−5 shows an example bit sequence for the receiver (in the figure, − indicates that the bit on the DR pin is ignored and thus is not passed to DRR1).

Figure 2−5. A-bis Mode Receive Operation

RCERA	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1																	
RCERB																	0	0	1	0	0	1	1	1	0	0	1	0	0	1	1	1	
DR pin	1	0	1	1	0	1	0	1	0	0	1	1	0	1	0	1	0	1	1	1	0	1	1	0	1	1	0	1	1	0	1	1	
DRR1	−	0	−	1	−	1	−	1	−	0	−	1	0	−	−	1	−	−	1	−	−	1	1	0	−	−	0	−	−	0	1	1	75E3h

Note: − indicates that the bit on the DR pin is ignored and thus is not passed to DRR1.

2.2.2 A-bis Mode Transmit Operation

In the A-bis mode, only bits that are enabled in the XCERA and XCERB registers are transmitted out from the DX pin. Bits that are not enabled are not transmitted, and the DX pin is in the high-impedance state during that clock cycle. XCERA and XCERB alternate specifying the bit-enable pattern for each 16 clock cycles. When 16 enabled have been shifted out, the McBSP generates an interrupt to the CPU. Figure 2−6 shows an example bit sequence for the transmitter (in the figure, z indicates the high-impedance state).

Figure 2−6. A-bis Mode Transmit Operation

XCERA	0	1	1	1	1	1	0	0	0	1	1	0	0	1	1	1																
XCERB																	0	0	1	1	0	0	0	1	1	0	0	0	1	1	0	0
DXR1	1	0	1	1	0	1	0	1	0	0	0	1	1	1	1	1	1	0	1	1	0	1	1	0	1	1	0	0	0	0	0	0
DX pin	z	0	1	1	0	1	z	z	z	0	0	z	z	1	1	1	z	z	1	1	z	z	z	0	1	z	z	z	0	0	z	z

Note: z indicates the high-impedance state.

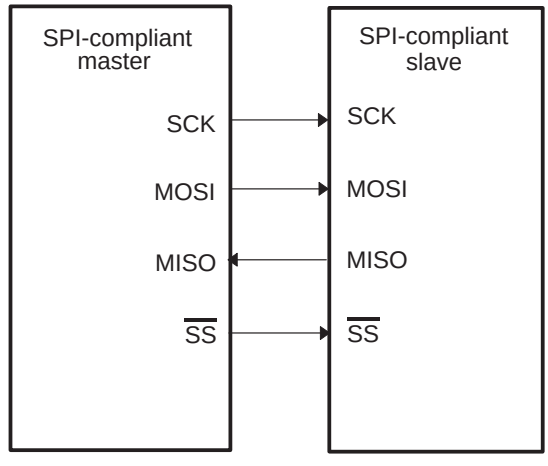
2.3 SPI Protocol

The SPI protocol is a master-slave configuration with one master device and one or more slave devices. The interface consists of the following four signals:

- Serial data input (also referred to as Master In − Slave Out, or MISO)
- Serial data output (also referred to as Master Out − Slave In, or MOSI)
- Shift-clock (also referred to as SCK)
- Slave-enable signal (also referred to as SS)

A typical SPI interface with a single slave device is shown in Figure 2−7.

Figure 2−7. Typical SPI Interface



The master device controls the flow of communication by providing shift-clock and slave-enable signals. The slave-enable signal is an optional active-low signal that enables the serial data input and output of the slave device (device not sending out the clock).

In the absence of a dedicated slave-enable signal, communication between the master and slave is determined by the presence or absence of an active shift-clock. In such a configuration, the slave device must remain enabled at all times, and multiple slaves cannot be used.

2.3.1 Clock Stop Mode

The clock stop mode of the McBSP provides compatibility with the SPI protocol. When the McBSP is configured in clock stop mode, the transmitter and receiver are internally synchronized, so that the McBSP functions as an SPI master or slave device. The transmit clock signal (CLKX) corresponds to the serial clock signal (SCK) of the SPI protocol, while the transmit frame-synchronization signal (FSX) is used as the slave-enable signal ($\overline{SS}$).

The receive clock signal (CLKR) and receive frame-synchronization signal (FSR) are not used in the clock stop mode because these signals are internally connected to their transmit counterparts, CLKX and FSX.

2.3.2 Bits Used to Enable and Configure the Clock Stop Mode

The bits required to configure the McBSP as an SPI device are introduced in Table 2−4. Table 2−5 shows how the various combinations of the CLKSTP bit and the polarity bits CLKXP and CLKRP create four possible clock stop mode configurations. The timing diagrams in section 2.3.3 show the effects of CLKSTP, CLKXP, and CLKRP.

Table 2−4. Bits Used to Enable and Configure the Clock Stop Mode

Bit Field	Description
CLKSTP bits of SPCR1	Use these bits to enable the clock stop mode and to select one of two timing variations. (See also Table 2−5.)
CLKXP bit of PCR	This bit determines the polarity of the CLKX signal. (See also Table 2−5.)
CLKRP bit of PCR	This bit determines the polarity of the CLKR signal. (See also Table 2−5.)
CLKXM bit of PCR	This bit determines whether CLKX is an input signal (McBSP as slave) or an output signal (McBSP as master).
XPHASE bit of XCR2	You must use a single-phase transmit frame (XPHASE = 0).
RPHASE bit of RCR2	You must use a single-phase receive frame (RPHASE = 0).
XFRLEN1 bits of XCR1	You must use a transmit frame length of 1 serial word (XFRLEN1 = 0).
RFRLEN1 bits of RCR1	You must use a receive frame length of 1 serial word (RFRLEN1 = 0).
XWDLEN1 bits of XCR1	The XWDLEN1 bits determine the transmit packet length. XWDLEN1 must be equal to RWDLEN1 because in the clock stop mode, the McBSP transmit and receive circuits are synchronized to a single clock.
RWDLEN1 bits of RCR1	The RWDLEN1 bits determine the receive packet length. RWDLEN1 must be equal to XWDLEN1 because in the clock stop mode, the McBSP transmit and receive circuits are synchronized to a single clock.

Table 2−5. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

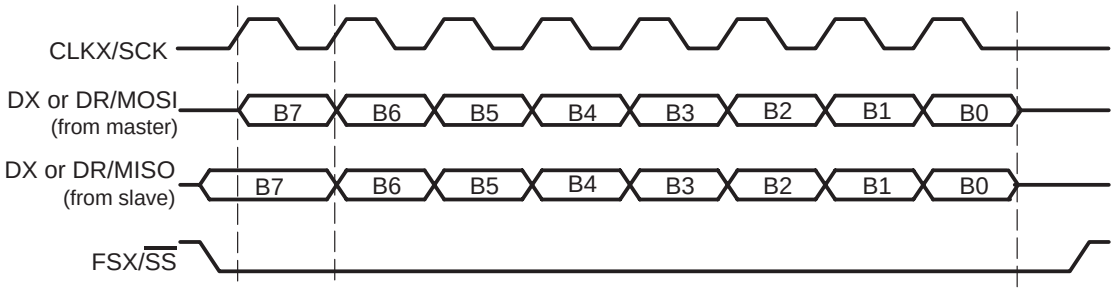
2.3.3 Clock Stop Mode Timing Diagrams

The timing diagrams for the four possible clock stop mode configurations are shown here. Notice that the frame-synchronization signal used in clock stop mode is active throughout the entire transmission as a slave-enable signal. Although the timing diagrams show 8-bit transfers, the packet length can be set to 8, 12, 16, 20, 24, or 32 bits per packet. The receive packet length is selected with the RWDLEN1 bits of RCR1, and the transmit packet length is selected with the XWDLEN1 bits of XCR1. For clock stop mode, the values of RWDLEN1 and XWDLEN1 must be the same because the McBSP transmit and receive circuits are synchronized to a single clock.

Note:

Even if multiple words are consecutively transferred, the CLKX signal is always stopped and the FSX signal returns to the inactive state after a packet transfer. When consecutive packet transfers are performed, this leads to a minimum idle time of two bit-periods between each packet transfer.

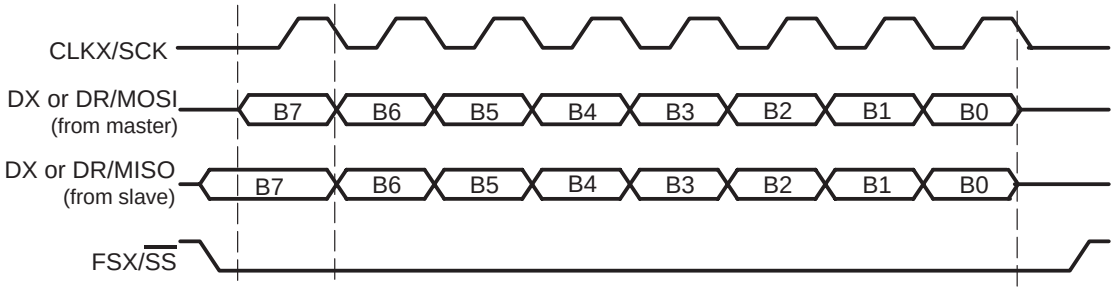
Figure 2−8. SPI Transfer With CLKSTP = 10b (no clock delay), CLKXP = 0, CLKRP = 0



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

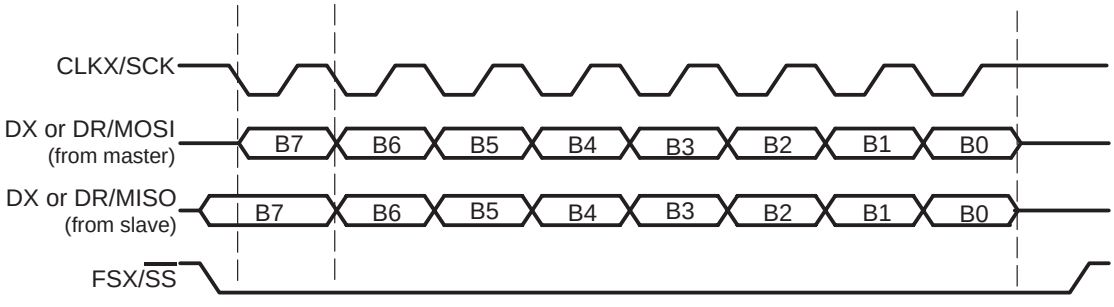
Figure 2−9. SPI Transfer With CLKSTP = 11b (clock delay), CLKXP = 0, CLKRP = 1



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

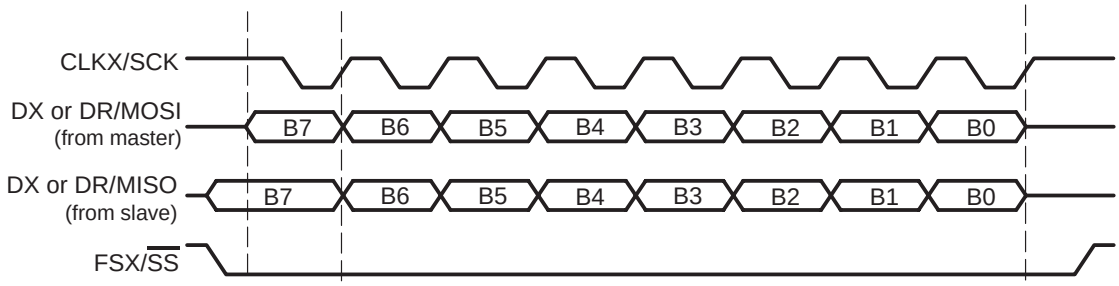
Figure 2−10. SPI Transfer With CLKSTP = 10b (no clock delay), CLKXP = 1, CLKRP = 0



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI = DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO = DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

Figure 2−11. SPI Transfer With CLKSTP = 11b (clock delay), CLKXP = 1, CLKRP = 1



Notes: 1) If the McBSP is the SPI master (CLKXM = 1), MOSI=DX. If the McBSP is the SPI slave (CLKXM = 0), MOSI = DR.

2) If the McBSP is the SPI master (CLKXM = 1), MISO=DR. If the McBSP is the SPI slave (CLKXM = 0), MISO = DX.

2.3.4 Procedure for Configuring a McBSP for SPI Operation

To configure the McBSP for SPI master or slave operation:

1) Place the transmitter and receiver in reset.

   Clear the transmitter reset bit ($\overline{\text{XRST}}$ = 0) in SPCR2, to reset the transmitter. Clear the receiver reset bit ($\overline{\text{RRST}}$ = 0) in SPCR1, to reset the receiver.

2) Place the sample rate generator in reset.

   Clear the sample rate generator reset bit ($\overline{\text{GRST}}$ = 0) in SPCR2, to reset the sample rate generator.

3) Program registers that affect SPI operation.

   Program the appropriate McBSP registers to configure the McBSP for proper operation as an SPI master or an SPI slave. For a list of important bits settings, see one of the following topics:

   - *McBSP as the SPI Master* (page 2-20)
   - *McBSP as an SPI Slave* (page 2-22)

4) Enable the sample rate generator.

   To release the sample rate generator from reset, set the sample rate generator reset bit ($\overline{\text{GRST}}$ = 1) in SPCR2.

   Make sure that during the write to SPCR2, you only modify $\overline{\text{GRST}}$. Otherwise, you will modify the McBSP configuration you selected in the previous step.

5) Enable the transmitter and receiver.

After the sample rate generator is released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

If the CPU services the McBSP transmit and receive buffers, then you can immediately enable the transmitter ($\overline{\text{XRST}}$ = 1 in SPCR2) and enable the receiver ($\overline{\text{RRST}}$ = 1 in SPCR1).

The FIFO has to be configured first if it must write into the McBSP transmit buffer or read from the McBSP receive buffer. After the FIFO has been configured, make XRST = 1 and $\overline{\text{RRST}}$ = 1.

Note: In either case, make sure you only change $\overline{\text{XRST}}$ and $\overline{\text{RRST}}$ when you write to SPCR2 and SPCR1. Otherwise, you will modify the bit settings you selected earlier in this procedure.

After the transmitter and receiver are released from reset, wait two sample rate generator clock periods for the McBSP logic to stabilize.

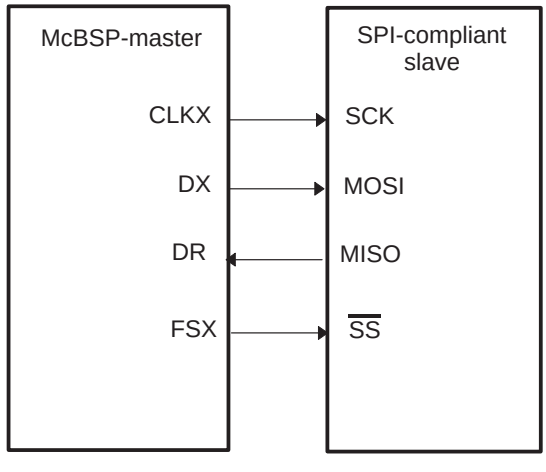
6) If necessary, enable the frame-sync logic of the sample rate generator.

After the required data acquisition setup is done (DXR[1/2] is loaded with data), set $\overline{\text{FRST}}$ = 1 if an internally generated frame-sync pulse is required (that is, if the McBSP is the SPI master).

2.3.5 McBSP as the SPI Master

An SPI interface with the McBSP used as the master is shown in Figure 2−12. When the McBSP is configured as a master, the transmit output signal (DX) is used as the MOSI signal of the SPI protocol, and the receive input signal (DR) is used as the MISO signal.

Figure 2−12. SPI Interface With McBSP as Master



The register bit values required to configure the McBSP as a master are listed in the following table. After the table are more details about the configuration requirements.

Table 2−6. Bit Values Required to Configure the McBSP as an SPI Master

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 1	The CLKX pin is an output pin driven by the internal sample rate generator. Because CLKSTP is equal to 10b or 11b, CLKR is driven internally by CLKX.
SCLKME = 0 CLKSM = 1	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock.
CLKGDV is a value from 0 to 255	CLKGDV defines the divide down value for CLKG.
FSXM = 1	The FSX pin is an output pin driven according to the FSGM bit.
FSGM = 0	The transmitter drives a frame-sync pulse on the FSX pin every time data is transferred from DXR1 to XSR1.
FSXP = 1	The FSX pin is active low.
XDATDLY = 01b RDATDLY = 01b	This setting provides the correct setup time on the FSX signal.

When the McBSP functions as the SPI master, it controls the transmission of data by producing the serial clock signal. The clock signal on the CLKX pin is enabled only during packet transfers. When packets are not being transferred, the CLKX pin remains high or low depending on the polarity used.

For SPI master operation, the CLKX pin must be configured as an output. The sample rate generator is then used to derive the CLKX signal from the CPU clock. The clock stop mode internally connects the CLKX pin to the CLKR signal so that no external signal connection is required on the CLKR pin, and both the transmit and receive circuits are clocked by the master clock (CLKX).

The data delay parameters of the McBSP (XDATDLY and RDATDLY) must be set to 1 for proper SPI master operation. A data delay value of 0 or 2 is undefined in the clock stop mode.

The McBSP can also provide a slave-enable signal ($\overline{SS}$) on the FSX pin. If a slave-enable signal is required, the FSX pin must be configured as an output,

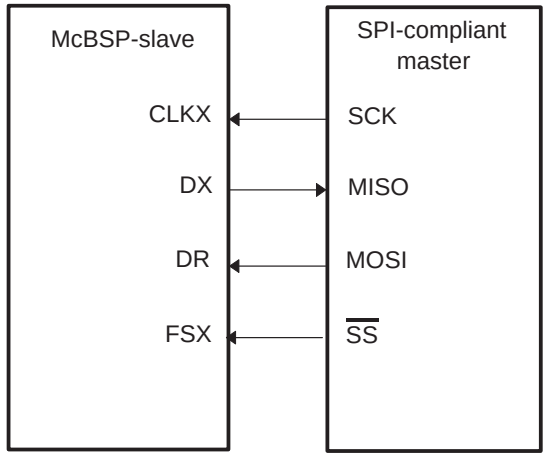
and the transmitter must be configured so that a frame-sync pulse is generated automatically each time a packet is transmitted (FSGM = 0). The polarity of the FSX pin is programmable high or low; however, in most cases the pin should be configured active-low.

When the McBSP is configured as described for SPI-master operation, the bit fields for frame-sync pulse width (FWID) and frame-sync period (FPER) are overridden, and custom frame-sync waveforms are not allowed. To see the resulting waveform produced on the FSX pin, see the timing diagrams in section 2.3.3. The signal becomes active before the first bit of a packet transfer, and remains active until the last bit of the packet is transferred. After the packet transfer is complete, the FSX signal returns to the inactive state.

2.3.6 McBSP as an SPI Slave

An SPI interface with the McBSP used as a slave is shown in Figure 2−13. When the McBSP is configured as a slave, DX is used as the MISO signal, and DR is used as the MOSI signal.

Figure 2−13. SPI With McBSP Configured as Slave



The register bit values required to configure the McBSP as a slave are listed in the following table. After the table are more details about the configuration requirements.

Table 2−7. Bit Values Required to Configure the McBSP as an SPI Slave

Required Bit Setting	Description
CLKSTP = 10b or 11b	The clock stop mode (without or with a clock delay) is selected.
CLKXP = 0 or 1	The polarity of CLKX as seen on the CLKX pin is positive (CLKXP = 0) or negative (CLKXP = 1).
CLKRP = 0 or 1	The polarity of CLKR as seen on the CLKR pin is positive (CLKRP = 0) or negative (CLKRP = 1).
CLKXM = 0	The CLKX pin is an input pin, so that it can be driven by the SPI master. Because CLKSTP = 10b or 11b, CLKR is driven internally by CLKX.
SCLKME = 0 CLKSM = 1	The clock generated by the sample rate generator (CLKG) is derived from the CPU clock. (The sample rate generator is used to synchronize the McBSP logic with the externally-generated master clock.)
CLKGDV = 1	The sample rate generator divides the CPU clock by 2 before generating CLKG.
FSXM = 0	The FSX pin is an input pin, so that it can be driven by the SPI master.
FSXP = 1	The FSX pin is active low.
XDATDLY = 00b RDATDLY = 00b	These bits must be 0s for SPI slave operation.

When the McBSP is used as an SPI slave, the master clock and slave-enable signals are generated externally by a master device. Accordingly, the CLKX and FSX pins must be configured as inputs. The CLKX pin is internally connected to the CLKR signal, so that both the transmit and receive circuits of the McBSP are clocked by the external master clock. The FSX pin is also internally connected to the FSR signal, and no external signal connections are required on the CLKR and FSR pins.

Although the CLKX signal is generated externally by the master and is asynchronous to the McBSP, the sample rate generator of the McBSP must be enabled for proper SPI slave operation. The sample rate generator should be programmed to its maximum rate of half the CPU clock rate. The internal sample rate clock is then used to synchronize the McBSP logic to the external master clock and slave-enable signals.

The McBSP requires an active edge of the slave-enable signal on the FSX input for each transfer. This means that the master device must assert the slave-enable signal at the beginning of each transfer, and deassert the signal after the completion of each packet transfer; the slave-enable signal cannot remain active between transfers.

The data delay parameters of the McBSP must be set to 0 for proper SPI slave operation. A value of 1 or 2 is undefined in the clock stop mode.

Chapter 3

Configure the Receiver and Transmitter

This chapter explains how to configure the receiver/transmitter and program the registers.

Topic **Page**

3.1 Receiver Configuration

To configure the McBSP receiver, perform the following procedure:

1) Place the McBSP receiver in reset (see section 3.1.2).
2) Program the McBSP registers for the desired receiver operation (see section 3.1.1).
3) Take the receiver out of reset (see section 3.1.2).

3.1.1 Programming the McBSP Registers for the Desired Receiver Operation

The following is a list of important tasks to be performed when you are configuring the McBSP receiver. Each task corresponds to one or more McBSP register bit fields. Note that in the list, SRG is an abbreviation for sample rate generator.

- **Global behavior:**
  - Set the receiver pins to operate as McBSP pins
  - Enable/disable the digital loopback mode
  - Enable/disable the clock stop mode
  - Enable/disable the receive multichannel selection mode
  - Enable/disable the A-bis mode
- **Data behavior:**
  - Choose 1 or 2 phases for the receive frame
  - Set the receive word length(s)
  - Set the receive frame length
  - Enable/disable the receive frame-sync ignore function
  - Set the receive companding mode
  - Set the receive data delay
  - Set the receive sign-extension and justification mode
  - Set the receive interrupt mode
- **Frame-sync behavior:**
  - Set the receive frame-sync mode
  - Set the receive frame-sync polarity
  - Set the SRG frame-sync period and pulse width
- **Clock behavior:**
  - Set the receive clock mode
  - Set the receive clock polarity
  - Set the SRG clock divide-down value
  - Set the SRG clock synchronization mode
  - Set the SRG clock mode (choose an input clock)
  - Set the SRG input clock polarity

3.1.2 Resetting and Enabling the Receiver

The first step of the receiver configuration procedure is to reset the receiver, and the last step is to enable the receiver (to take it out of reset).

The serial port can be reset in the following two ways:

1) A DSP reset ($\overline{\text{RESET}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{RESET}}$ signal released), GRST = FRST = RRST = XRST = 0, keeping the entire serial port in the reset state.

2) The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.

To get this result ...	Use this bit ...
Enable the receiver	RRST (bit 0 of the SPCR1 register) set to 1
Disable the receiver (the reset state)	RRST set to 0
Reset the sample-rate generator	GRST (bit 6 of the SPCR2 register) set to 0
Enable the sample-rate generator	GRST set to 1
Reset the frame-sync logic	FRST (bit 7 of the SPCR2 register) = 0
Enable the frame-sync logic	FRST = 1

For details on the Serial Port Control Register 1 (SPCR1) and SPCR2 see Figure 6−3 (page 6-4) and Figure 6−4 (page 6-6). Table 3−1 shows the state of McBSP pins when the serial port is reset due to a DSP reset and a direct receiver/transmitter reset.

Table 3−1. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced By DSP Reset	State Forced By Receiver/Transmitter Reset
			Receiver Reset ($\overline{RRST}$ = 0 and $\overline{GRST}$ = 1)
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if Input; CLKR running if output
FSR	I/O/Z	Input	Known state if Input; FSRP inactive state if output
			Transmitter Reset ($\overline{XRST}$ = 0 and $\overline{GRST}$ = 1)
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if Input; CLKX running if output
FSX	I/O/Z	Input	Known state if Input; FSXP inactive state if output

Note: In Possible State(s) column, I = Input, O = Output, Z = High impedance

For more details about McBSP reset conditions and effects, see *Resetting and Initializing a McBSP* on page 4-3.

To get this result…	**Set this bit...**
Set the receiver pins to operate as McBSP pins in the reset state.	RIOEN (bit 12 of the PCR register) set to 0 0 is the only possible setting for this bit. 1 is a reserved function and must not be used.
Disable the digital-loopback mode	DLB (bit 15 of the SPCR1 register)
Enable the digital-loopback mode	DLB (bit 15 of the SPCR1 register)

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 3−2. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 3−2. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This receive signal …	**Is fed internally by this transmit signal** …
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
CLKR (receive clock)	CLKX (transmit clock)

3.1.3 Clock Stop Mode

The CLKSTP bits in the serial port control registers determine whether the clock stop mode is on.

The clock stop mode supports the SPI master-slave protocol. If you will not be using the SPI protocol, you can clear the CLKSTP bits (12:11 in SPCR1) to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the CLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the CLKR pin.

Table 3−3 summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. Note that in the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-sync signal is tied internally to the transmit frame-sync signal.

Table 3−3. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

3.1.4 Receive Multichannel Selection and A-bis Modes

The RMCM bit determines whether the receive multichannel selection mode is on. For more details on this mode, see *Receive Multichannel Selection Mode* on page 2-7. For more details on the multichannel control registers, see Section 6.6 (page 6-17).

For more details on A-bis mode, see page 2-13. If A-bis mode is enabled, individual bits can be enabled or disabled during reception and transmission. For transmission, the bits are controlled by transmit channel enable registers A and B (XCERA and XCERB). For reception, the bits are controlled by receive channel enable registers A and B (RCERA and RCERB).

To get this result…	**Use this bit** …
Enable multichannel selection mode	RMCM bit in MCR1 to 1 (individual channels can be enabled or disabled with this setting)
Disable multichannel selection mode	Set RMCM to 0 so that all 128 channels are enabled
Enable A-bis mode	ABIS bit (number 6 in SPCR1) set to 1
Disable A-bis mode	ABIS set to 0

3.1.5 Choose 1 or 2 Phases for the Receive Frame

The RPHASE bit in the Receive Control Register 2 (RCR2) determines whether the receive data frame has one or two phases. See Section 6.3 for details of RCR1 and RCR2.

To get this result …	**Use this bit**…
Set receive frame to single phase	RPHASE bit in Receive FRCR2 set to 0
Set receive frame to dual-phase frame	RPHASE bit in RCR2 set to 1

3.1.5.1 Set the Receive Word Length(s)

The RWDLEN1 and RWDLEN2 bit fields determine how many bits are in each serial word in phase 1 and in phase 2, respectively, of the receive data frame.

To get this result …	Use this bit…	
Specify the length of every serial word in phase 1 of the receive frame	Set RWDLEN1 (bits 7:5 of RCR1) to one of the following:	
	000	8 bits
	001	12 bits
	010	16 bits
	011	20 bits
	100	24 bits
	101	32 bits
	11x	Reserved

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, RWDLEN1 selects the length for every serial word received in the frame. If a dual-phase frame is selected, RWDLEN1 determines the length of the serial words in phase 1 of the frame, and RWDLEN2 determines the word length in phase 2 of the frame with the same bit settings. For details on the receive control registers, see Section 6.3 (page 6-8).

3.1.5.2 Set the Receive Frame Length

The RFRLEN1 and RFRLEN2 bit fields determine how many serial words are in phase 1 and in phase 2, respectively, of the receive data frame.

To get this result …	Use this bit…	
Set receive frame length	Set RFRLEN1 (bits 14:8 of RCR1) to one of the following:	
	000 0000	1 word in phase 1
	000 0001	2 words in phase 1
	.	
	.	
	.	
	111 1111	128 words in phase 1

The receive frame length is the number of serial words in the receive frame. Each frame can have one or two phases, depending on value that you load into the RPHASE bit.

If a single-phase frame is selected (RPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (RPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2:

The 7-bit RFRLEN fields allow up to 128 words per phase. See Table 3−4 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Note: Program the RFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into RFRLEN1.

Table 3−4. How to Calculate the Length of the Receive Frame

RPHASE	RFRLEN1	RFRLEN2	Frame Length
0	0 ≤ RFRLEN1 ≤ 127	Don't care	(RFRLEN1 + 1) words
1	0 ≤ RFRLEN1 ≤ 127	0 ≤ RFRLEN2 ≤ 127	(RFRLEN1 + 1) + (RFRLEN2 + 1) words

3.1.5.3 Enable/Disable the Receive Frame-Sync Ignore Function

The RFIG bit controls the receive frame-sync ignore function.

If there is an unexpected receive frame-sync pulse …	**Use this bit**…
Restart the frame transfer	Set RFIG (bit 2 in RCR2) to 0
Ignore the unexpected pulses	Set RFIG to 1

If a frame-synchronization (frame-sync) pulse starts the transfer of a new frame before the current frame is fully received, this pulse is treated as an unexpected frame-sync pulse.

When RFIG = 1, reception continues, ignoring the unexpected frame-sync pulses.

When RFIG = 0, an unexpected FSR pulse causes the McBSP to discard the contents of RSR[1,2] in favor of the new incoming data. Therefore, if RFIG = 0 and an unexpected frame-sync pulse occurs, the serial port:

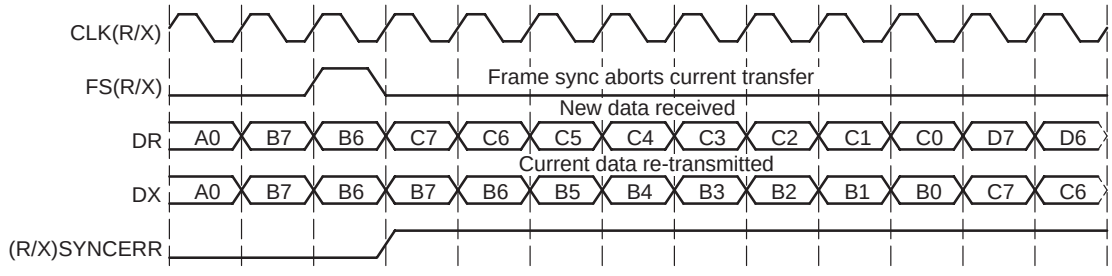
1) Aborts the current data transfer
2) Sets RSYNCERR in SPCR1 to 1
3) Begins the transfer of a new data word

For more details about the frame-sync error condition, see *Unexpected Receive Frame-Sync Pulse* on page 1-40.

3.1.5.4 Examples Showing the Effects of RFIG

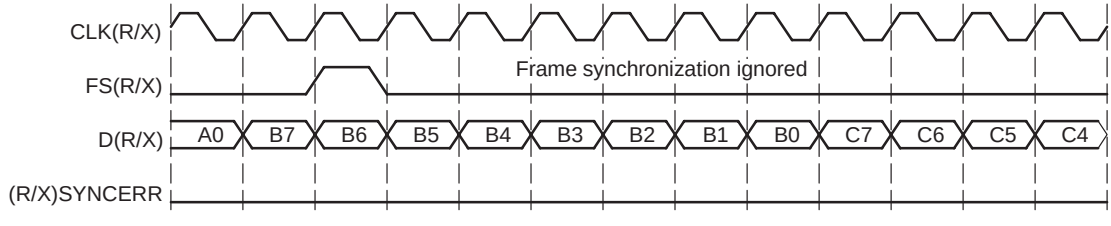
Figure 3−1 shows an example in which word B is interrupted by an unexpected frame-sync pulse when (R/X)FIG = 0. In the case of reception, the reception of B is aborted (B is lost), and a new data word (C in this example) is received after the appropriate data delay. This condition is a receive synchronization error, and thus sets the RSYNCERR bit.

Figure 3−1. Unexpected Frame-Sync Pulse With (R/X)FIG = 0



In contrast with Figure 3−1, Figure 3−2 shows McBSP operation when unexpected frame-sync signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected pulse.

Figure 3−2. Unexpected Frame-Sync Pulse With (R/X)FIG = 1



3.1.6 Set the Receive Companding Mode

Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

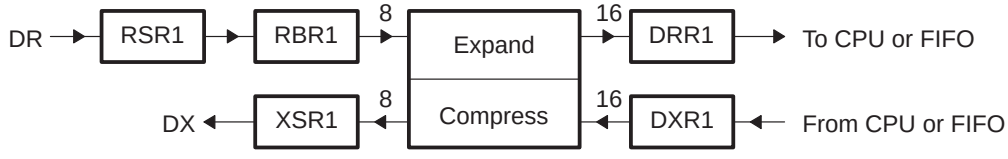
A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or FIFO must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits

(RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

Figure 3−3 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to 2s-complement format.

Figure 3−3. Companding Processes for Reception and for Transmission



For reception, the 8-bit compressed data in RBR1 is expanded to left-justified 16-bit data in DRR1. Note that the RJUST bit of SPCR1 is ignored when companding is used.

If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See *Capability to Compand Internal Data* on page 1-13.

Normally, the McBSP transmit or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set RCOMPAND = 01b in RCR2, the bit ordering of 8-bit words is reversed during reception. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

The RCOMPAND bits determine whether companding or another data transfer option is chosen for McBSP reception. Modes other than 00b are enabled only when the appropriate RWDLEN is 000b, indicating 8-bit data.

To get this result …	**Use this bit**…
Receive companding mode	Set RCOMPAND (bits 4:3 in RCR2) to one of the following
	00 No companding, any size data, MSB received first

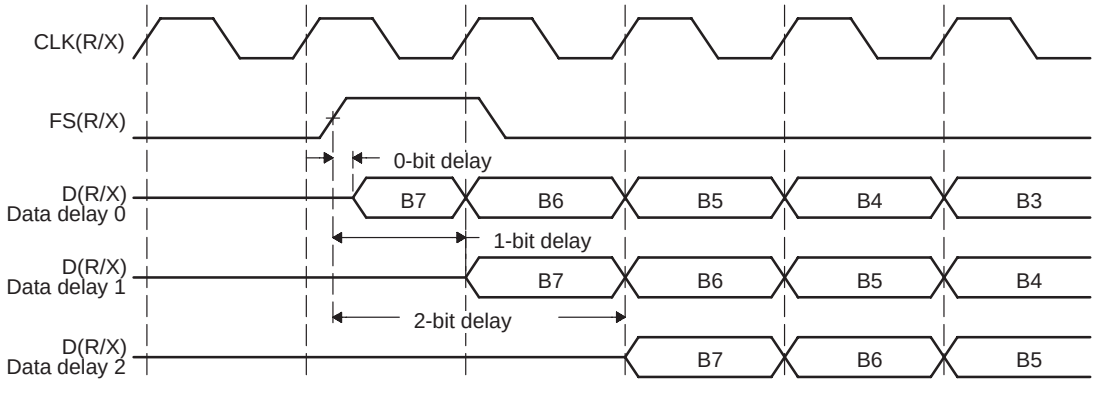
To get this result …	Use this bit…	
	01	No companding, 8-bit data, LSB received first (for details, scroll down to Option to Receive LSB First)
	10	μ-law companding, 8-bit data, MSB received first
	11	A-law companding, 8-bit data, MSB received first

3.1.7 Set the Receive Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

RDATDLY specifies the data delay for reception. The range of programmable data delay is zero to two bit-clocks (RDATDLY = 00b–10b), as shown in Figure 3−4. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-sync pulse.

Figure 3−4. Range of Programmable Data Delay



3.1.7.1 0-Bit Data Delay

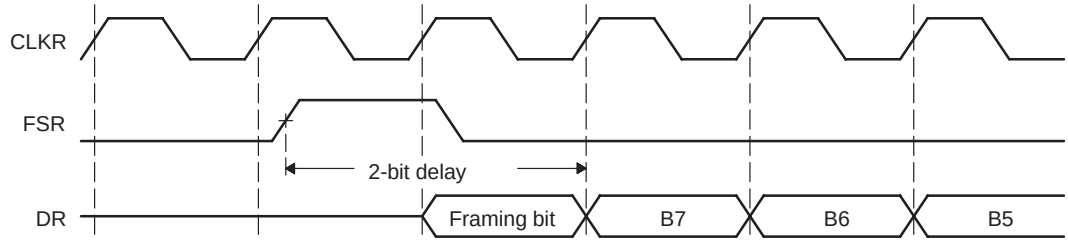
Normally, a frame-sync pulse is detected or sampled with respect to an edge of internal serial clock CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

For reception this problem is solved because receive data is sampled on the first falling edge of CLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus on DX. The transmitter then asynchronously detects the frame-sync signal (FSX) going active high and immediately starts driving the first bit to be transmitted on the DX pin.

3.1.7.2 2-Bit Data Delay

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in Figure 3−5. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 3−5. 2-Bit Data Delay Used to Skip a Framing Bit



The RDATDLY bits determine the length of the data delay for the receive frame.

To get this result …	**Use this bit**…	
Set the receive data delay	Set RDATDLY (bits 1:0 in RCR2 to:	
	00	0-bit data delay
	01	1-bit data delay
	10	2-bit data delay
	11	Reserved

3.1.8 Set the Receive Sign-Extension and Justification Mode

RJUST in SPCR1 selects whether data in RBR[1,2] is right- or left-justified (with respect to the MSB) in DRR[1,2] and how unused bits in DRR[1,2] are filled—with zeros or with sign bits.

Table 3−5 and Table 3−6 show the effects of various RJUST values. The first table shows the effect on an example 12-bit receive-data value 0xABC. The second table shows the effect on an example 20-bit receive-data value 0xABCDE.

Table 3−5. Example: Use of RJUST Field With 12-Bit Data Value 0xABC

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0x0000	0x0ABC
01b	Right	Sign extend data into MSBs	0xFFFF	0xFABC
10b	Left	Zero fill LSBs	0x0000	0xABC0
11b	Reserved	Reserved	Reserved	Reserved

Table 3−6. Example: Use of RJUST Field With 20-Bit Data Value 0xABCDE

RJUST	Justification	Extension	Value in DRR2	Value in DRR1
00b	Right	Zero fill MSBs	0x000A	0xBCDE
01b	Right	Sign extend data into MSBs	0xFFFA	0xBCDE
10b	Left	Zero fill LSBs	0xABCD	0xE000
11b	Reserved	Reserved	Reserved	Reserved

The RJUST bits determine whether data received by the McBSP is sign extended and how it is justified.

To get this result …	**Use this bit**…	
Set receive sign-extension and justification mode	Set RJUST (bits 14:13 in SPCR1 to:	
	00	Right justify data and zero fill MSBs in DRR[1,2]
	01	Right justify data and sign extend it into the MSBs in DRR[1,2]
	10	Left justify data and zero fill LSBs in DRR[1,2]
	11	Reserved

3.1.9 Set the Receive Interrupt Mode

The receive interrupt (RINT) signals the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the receive interrupt mode bits, RINTM, in SPCR1.

- ❑ RINTM = 00b. Interrupt on every serial word by tracking the RRDY bit in SPCR1. Note that regardless of the value of RINTM, RRDY can be read to detect the RRDY = 1 condition.

- ❑ RINTM = 01b. In the multichannel selection mode, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.

- ❑ RINTM = 10b. Interrupt on detection of receive frame-sync pulses. This generates an interrupt even when the receiver is in its reset state. This is done by synchronizing the incoming frame-sync pulse to the CPU clock and sending it to the CPU via RINT.

- ❑ RINTM = 11b. Interrupt on frame-synchronization error. Note that regardless of the value of RINTM, RSYNCERR can be read to detect this condition. For information on using RSYNCERR, see *Unexpected Receive Frame-Sync Pulse* on page 1-40. See section 5.4 for RINTM interrupt selection using FIFO logic.

The RINTM bits determine which event generates a receive interrupt request to the CPU.

To get this result …	**Use this bit**…	
Receive interrupt mode	Set RINTM (bits 5:4 in SPCR1 to:	
	00	RINT generated when RRDY changes from 0 to 1
	01	RINT generated by an end-of-block or end-of-frame condition in the receive multichannel selection mode
	10	RINT generated by a new receive frame-sync pulse
	11	RINT generated when RSYNCERR is set

3.1.10 Set the Receive Frame-Sync Mode

Table 3−7 shows how you may select various sources to provide the receive frame-synchronization signal and the effect on the FSR pin. The polarity of the signal on the FSR pin is determined by the FSRP bit.

Note that in the digital loop back mode (DLB = 1), the transmit frame-sync signal is used as the receive frame-sync signal.

Also, in the clock stop mode, the internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 3−7. Select Sources to Provide the Receive Frame-Synchronization Signal and the Effect on the FSR Pin

DLB	FSRM	GSYNC	Source of Receive Frame Synchronization	FSR Pin Status
0	0	0 or 1	An external frame-sync signal enters the McBSP through the FSR pin. The signal is then inverted as determined by FSRP before being used as internal FSR.	Input
0	1	0	Internal FSR is driven by the sample rate generator frame-sync signal (FSG).	Output. FSG is inverted as determined by FSRP before being driven out on the FSR pin.
0	1	1	Internal FSR is driven by the sample rate generator frame-sync signal (FSG).	Input. The external frame-sync input on the FSR pin is used to synchronize CLKG and generate FSG pulses.
1	0	0	Internal FSX drives internal FSR.	High impedance
1	0 or 1	1	Internal FSX drives internal FSR.	Input. If the sample rate generator is running, external FSR is used to synchronize CLKG and generate FSG pulses.
1	1	0	Internal FSX drives internal FSR.	Output. Receive (same as transmit) frame synchronization is inverted as determined by FSRP before being driven out on the FSR pin.

3.1.11 Set the Receive Frame-Sync Polarity

The FSRP bit determines whether frame-synchronization (frame-sync) pulses are active high or active low on the FSR pin.

Receive frame-sync pulses can be either generated internally by the sample rate generator (see section 1.4.2 on page 1-30) or driven by an external source. The source of frame sync is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see *Set the Receive Frame-Sync Mode* on page 3-15. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR.

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. Note that these internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

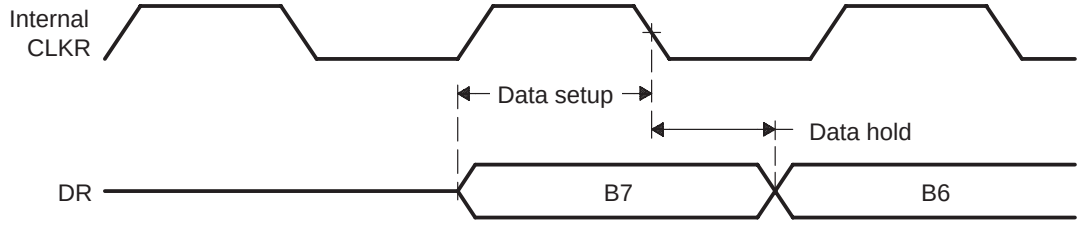
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-sync signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-sync signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-sync signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin. The figure in the topic Clock and Frame Generation shows this inversion using XOR gates.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Note that data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1, and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. Note that the receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1, and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 3−6 shows how data clocked by an external serial device using a rising edge, may be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 3−6. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



The FSRM (in PCR), GSYNC (in SRGR2), DLB and CLKSTP (both in SPCR1) determine the source for receive frame synchronization and the function of the FSR pin.

To get this result …	**Use this bit**…
Frame synchronization Supply receive frame synchronization by an external source via the FSR pin.	Set FSRM (bit 10 in PCR) to 0.
Receive frame synchronization is supplied by the sample rate generator. FSR is an output pin reflecting internal FSR, except when GSYNC = 1 in SRGR2.	Set FSRM (bit 10 in PCR) to 1
Set sample rate generator clock synchronization mode Use no clock synchronization. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.	Set GSYNC (bit 15 in SRGR2) to 0.

To get this result …	Use this bit…
Use clock synchronization. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKS, CLKR, or CLKX pin. ❑ FSG pulses. FSG **only** pulses in response to a pulse on the FSR pin. The frame-sync period defined in FPER is ignored.	Set GSYNC to 1. For more details, see *Synchronizing Sample Rate Generator Outputs to an External Clock* on page 1-31.
Set digital loopback (DLB) mode. Disable DLB mode	Set DLB (bit 15 in SPCR1) to 0.
Enable digital loopback mode. The receive signals, including the receive frame-sync signal, are connected internally through multiplexers to the corresponding transmit signals.	Set DLB to 1.
Set clock stop mode Disable clock stop mode − normal clocking for non-SPI mode.	Set CLKSTP (bits 12:11 in SPCR1) to 0Xb
Enable clock stop mode without clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.	Set CLKSTP (bits 12:11 in SPCR1) to 10b.
Clock stop mode enabled, with clock delay. The internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.	Set CLKSTP (bits 12:11 in SPCR1) to 11b.

3.1.11.1 Set the SRG Frame-Sync Period and Pulse Width

The sample rate generator can produce a clock signal, CLKG, and a frame-sync signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

On FSG, the period from the start of a frame-sync pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-sync

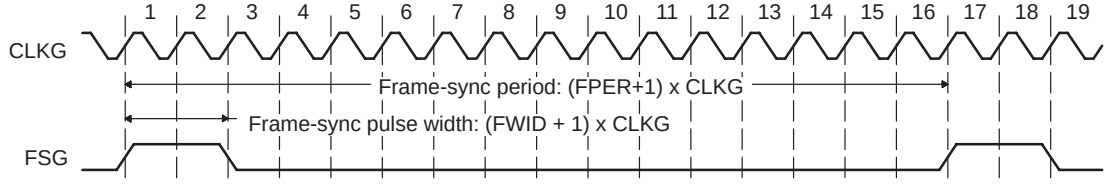
period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don't care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 3−7 shows a frame-sync period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-sync pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 3−7. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when $\overline{\text{FRST}}$ = 1 and FSGM = 1, a frame-sync pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

To get this result …	**Use this bit**…
Set the SRG frame-sync period and pulse width	FPER (bits 11:0 in SRGR2) set to range FPER + 1 (1 to 4096 CLKG cycles) For the frame-sync signal FSG, (FPER + 1) determines the period from the start of a frame-sync pulse to the start of the next frame-sync pulse.
Set the sample rate generator frame-sync pulse width	FWID (bits 15:8) in SRGR1 This field plus 1 determines the width of each frame-sync pulse on FSG.

3.1.11.2 Set the Receive Clock Mode

To get this result …	Use this bit…
Enable receive clock mode with digital loop-back (DLB) mode not set (DLB − 1)	Set CLKRM (bit 7 in PCR) to 0.
Enable receive clock mode with DLB mode not set (DLB − 1)	Set CLKRM (bit 7 in PCR) to 0.
Enable DLB mode	Set DLB (bit 15 in SPCR1) to 1
Disable DLB	Set DLB (bit 15 in SPCR1) to 0
Enable clock stop mode with clock delay	Set CLKSTP (bits 12:11 in SPCR1) to 11b.
Enable clock stop mode without clock delay	Set CLKSTP (bits 12:11 in SPCR1) to 10b.
Disable clock stop mode (normal clocking for non-SPI mode)	Set CLKSTP (bits 12:11 in SPCR1) to 0Xb.

3.1.11.3 Selecting a Source for the Receive Clock and a Data Direction for the CLKR Pin

Table 3−8 shows how you may select various sources to provide the receive clock signal and the effect on the CLKR pin. The polarity of the signal on the CLKR pin is determined by the CLKRP bit.

Note that in the digital loop back mode (DLB = 1), the transmit clock signal is used as the receive clock signal.

Also, in the clock stop mode, the internal receive clock signal (CLKR) and the internal receive frame-synchronization signal (FSR) are internally connected to their transmit counterparts, CLKX and FSX.

Table 3−8. Select Sources to Provide the Receive Clock Signal and the Effect on the CLKR Pin

DLB in SPCR1	CLKRM in PCR	Source of Receive Clock	CLKR Pin Status
0	0	The CLKR pin is an input driven by an external clock. The external clock signal is inverted as determined by CLKRP before being used.	Input
0	1	The sample rate generator clock (CLKG) drives internal CLKR.	Output. CLKG, inverted as determined by CLKRP, is driven out on the CLKR pin.

Table 3−8. Select Sources to Provide the Receive Clock Signal and the Effect on the CLKR Pin (Continued)

DLB in SPCR1	CLKRM in PCR	Source of Receive Clock	CLKR Pin Status
1	0	Internal CLKX drives internal CLKR.	High impedance
1	1	Internal CLKX drives internal CLKR.	Output. Internal CLKR (same as internal CLKX) is inverted as determined by CLKRP before being driven out on the CLKR pin.

Receive frame-sync pulses can be either generated internally by the sample rate generator (see section 1.4.2 on page 1-30) or driven by an external source. The source of frame sync is selected by programming the mode bit, FSRM, in PCR. FSR is also affected by the GSYNC bit in SRGR2. For information about the effects of FSRM and GSYNC, see *Set the Receive Frame-Sync Mode* on page 3-15. Similarly, receive clocks can be selected to be inputs or outputs by programming the mode bit, CLKRM, in the PCR.

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. Note that these internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-sync signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-sync signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-sync signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin. The figure in the topic Clock and Frame Generation shows this inversion using XOR gates.

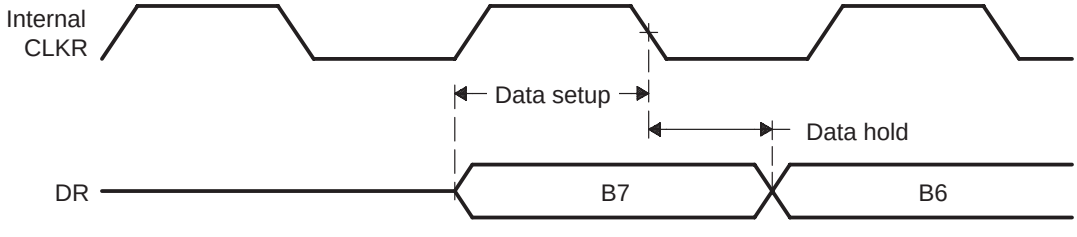
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Note that data is always transmitted on the

rising edge of internal CLKX. If CLKXP = 1, and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. Note that the receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1, and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 3−8 shows how data clocked by an external serial device using a rising edge, may be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 3−8. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



3.1.11.4 Set the Receive Clock Polarity

To get this result …	**Use this bit**…
Sample receive data on falling edge of CLKR	Set CLRP (bit 0 in PCR) to 0
Sample receive data on rising edge of CLKR	Set CLRP (bit 0 in PCR) to 1

3.1.11.5 Set the SRG Clock Divide-Down Value

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

To get this result …	**Use this bit**…
Set the divide down value of the sample rate generator clock	Set CLKGDV (bits 7−0 of SRGR1) to generate the required SRG clock frequency. The default value is 1 (divide input clock by 2).
Set the SRG clock synchronization mode The sample rate generator clock (CLKG) is free running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.	Set GYSYNC (bit 15 of SRGR2) to 0
Clock synchronization is performed. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKR or CLKX pin. ❑ FSG pulses. FSG **only** pulses in response to a pulse on the FSR pin. The frame-sync period defined in FPER is ignored.	Set GYSYNC (bit 15 of SRGR2) to 1

GSYNC is used only when the input clock source for the sample rate generator is external—on the CLKR or CLKX pin.

For more details on using the clock synchronization feature, see *Synchronizing Sample Rate Generator Outputs to an External Clock* on page 1-31.

3.1.11.6 Set the SRG Clock Mode (Choose an Input Clock)

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. The preceding table shows the four possible sources of the input clock. For more details on generating CLKG, see *Clock Generation in the Sample Rate Generator* on page 1-27.

To get this result …	**Use this bit**…
Choose sample rate generator clock derived from CLKS pin	Set SCLKME (bit 7 of PCR) and CLKSM (bit 13 of SRGR2) both to 0.
Choose sample rate generator clock derived from CPU clock. (This is the condition forced by a DSP reset.)	Set SCLKME = 0 and CLKSM = 1.
Choose sample rate generator clock derived from CLKR pin	Set SCLKME = 1 and CLKSM = 0.
Choose sample rate generator clock derived from CLKX pin	Set SCLKME = 1 and CLKSM = 1.

3.1.11.7 Set the SRG Input Clock Polarity

To get this result …	**Use this bit**…
CLKXP determines the input clock polarity when the CLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1).	Set CLKXP (bit 1 in the PCR register) to 0.
Rising edge on CLKX pin generates transitions on CLKG and FSG.	
Falling edge on CLKX pin generates transitions on CLKG and FSG.	Set CLKXP to 1.

To get this result …	Use this bit…
CLKR Pin Polarity CLKRP determines the input clock polarity when the CLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0). Falling edge on CLKR pin generates transitions on CLKG and FSG.	Set CLKRP to 0.
Rising edge on CLKR pin generates transitions on CLKG and FSG.	Set CLKRP to 1.

3.1.11.8 Using CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-sync signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKS, CLKX, or CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKXP for the CLKX pin and CLKRP for the CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

3.2 Transmitter Configuration

To configure the McBSP transmitter, perform the following procedure:

1) Place the McBSP/transmitter in reset (see section 3.2.2).
2) Program the McBSP registers for the desired transmitter operation (see 3.2.1).
3) Take the transmitter out of reset (see section 3.2.2).

3.2.1 Programming the McBSP Registers for the Desired Transmitter Operation

The following is a list of important tasks to be performed when you are configuring the McBSP transmitter. Each task corresponds to one or more McBSP register bit fields. Note that in the list, SRG is an abbreviation for sample rate generator.

- **Global behavior:**
  - Set the transmitter pins to operate as McBSP pins
  - Enable/disable the digital loopback mode
  - Enable/disable the clock stop mode
  - Enable/disable transmit multichannel selection
  - Enable/disable the A-bis mode
- **Data behavior:**
  - Choose 1 or 2 phases for the transmit frame
  - Set the transmit word length(s)
  - Set the transmit frame length
  - Enable/disable the transmit frame-sync ignore function
  - Set the transmit companding mode
  - Set the transmit data delay
  - Set the transmit DXENA mode
  - Set the transmit interrupt mode
- **Frame-sync behavior:**
  - Set the transmit frame-sync mode
  - Set the transmit frame-sync polarity
  - Set the SRG frame-sync period and pulse width
- **Clock behavior:**
  - Set the transmit clock mode
  - Set the transmit clock polarity
  - Set the SRG clock divide-down value
  - Set the SRG clock synchronization mode
  - Set the SRG clock mode (choose an input clock)
  - Set the SRG input clock polarity

3.2.2 Resetting and Enabling the Transmitter

The first step of the transmitter configuration procedure is to reset the transmitter, and the last step is to enable the transmitter (to take it out of reset).

To get this result ...	Use this bit ...
Enable the serial port transmitter	XRST (bit 0 of the SPCR2 register) set to 1
Disable the serial port transmitter (the reset state)	XRST set to 0
Reset the sample-rate generator	GRST (bit 6 of the SPCR2 register) set to 0
Enable the sample-rate generator	GRST set to 1
Reset the frame-sync logic	FRST (bit 7 of the SPCR2 register) = 0
Enable the frame-sync logic	FRST = 1

3.2.2.1 Reset Considerations

The serial port can be reset in the following two ways:

1) A DSP reset ($\overline{\text{RESET}}$ signal driven low) places the receiver, transmitter, and sample rate generator in reset. When the device reset is removed ($\overline{\text{RESET}}$ signal released), GRST = FRST= RRST = XRST = 0, keeping the entire serial port in the reset state.

2) The serial port transmitter and receiver can be reset directly using the RRST and XRST bits in the serial port control registers. The sample rate generator can be reset directly using the GRST bit in SPCR2.

Table 3−9 shows the state of McBSP pins when the serial port is reset due to a DSP reset and a direct receiver/transmitter reset.

Table 3−9. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced By DSP Reset	State Forced By Receiver/Transmitter Reset
			Receiver Reset (RRST = 0 and GRST = 1)
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if Input; CLKR running if output
FSR	I/O/Z	Input	Known state if Input; FSRP inactive state if output
			Transmitter Reset (XRST = 0 and GRST = 1)
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if Input; CLKX running if output
FSX	I/O/Z	Input	Known state if Input; FSXP inactive state if output

For more details about McBSP reset conditions and effects, see *Resetting and Initializing a McBSP* on page 4-3.

3.2.3 Set the Transmitter Pins to Operate as McBSP Pins

To get this result…	Set this bit...
Set the receiver pins to operate as McBSP pins in the reset state.	RIOEN (bit 12 of the PCR register) set to 0 0 is the only possible setting for this bit. 1 is a reserved function and must not be used.

3.2.4 Enable/Disable the Digital Loopback Mode

In the digital loopback mode, the receive signals are connected internally through multiplexers to the corresponding transmit signals, as shown in Table 3−10. This mode allows testing of serial port code with a single DSP device; the McBSP receives the data it transmits.

Table 3−10. Receive Signals Connected to Transmit Signals in Digital Loopback Mode

This receive signal …	Is fed internally by this transmit signal …
DR (receive data)	DX (transmit data)
FSR (receive frame synchronization)	FSX (transmit frame synchronization)
CLKR (receive clock)	CLKX (transmit clock)

The DLB bit determines whether the digital loopback mode is on.

To get this result…	Set this bit...
Disable the digital-loopback mode	DLB (bit 15 of the SPCR1 register)
Enable the digital-loopback mode	DLB (bit 15 of the SPCR1 register)

3.2.5 Enable/Disable the Clock Stop Mode

The CLKSTP bits determine whether the clock stop mode is on.

To get this result…	Set this bit...
Disable clock stop mode (normal clocking for nonSPI mode)	CLKSTP (bits 11 and 12 of the SPCR1 register) set to 0Xb
Enable clock stop mode without clock delay	CLKSTP (bits 11 and 12 of the SPCR1 register)set to 10b
Enable clock stop mode with clock delay	CLKSTP (bits 11 and 12 of the SPCR1 register) set to 11b

The clock stop mode supports the SPI master-slave protocol. If you will not be using the SPI protocol, you can clear CLKSTP to disable the clock stop mode.

In the clock stop mode, the clock stops at the end of each data transfer. At the beginning of each data transfer, the clock starts immediately (CLKSTP = 10b) or after a half-cycle delay (CLKSTP = 11b). The CLKXP bit determines whether the starting edge of the clock on the CLKX pin is rising or falling. The CLKRP bit determines whether receive data is sampled on the rising or falling edge of the clock shown on the CLKR pin.

Table 3−11 summarizes the impact of CLKSTP, CLKXP, and CLKRP on serial port operation. Note that in the clock stop mode, the receive clock is tied internally to the transmit clock, and the receive frame-sync signal is tied internally to the transmit frame-sync signal.

Table 3−11. Effects of CLKSTP, CLKXP, and CLKRP on the Clock Scheme

Bit Settings	Clock Scheme
CLKSTP = 00b or 01b CLKXP = 0 or 1 CLKRP = 0 or 1	Clock stop mode disabled. Clock enabled for non-SPI mode.
CLKSTP = 10b CLKXP = 0 CLKRP = 0	Low inactive state without delay: The McBSP transmits data on the rising edge of CLKX and receives data on the falling edge of CLKR.
CLKSTP = 11b CLKXP = 0 CLKRP = 1	Low inactive state with delay: The McBSP transmits data one-half cycle ahead of the rising edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 10b CLKXP = 1 CLKRP = 0	High inactive state without delay: The McBSP transmits data on the falling edge of CLKX and receives data on the rising edge of CLKR.
CLKSTP = 11b CLKXP = 1 CLKRP = 1	High inactive state with delay: The McBSP transmits data one-half cycle ahead of the falling edge of CLKX and receives data on the falling edge of CLKR.

3.2.6 Enable/Disable Transmit Multichannel Selection

To get this result…	**Use this bit** …
Enable all channels	XMCM (bits 0 and 1 of the MCR2 register) to 00 (No channels can be disabled or masked with this setting.)
Disable all channels unless they are selected in the appropriate transit channel enable registers (XCERs).	Set XMCM to 01b. A channel is in this mode is also unmasked.
Enable all channels as masked unless they are selected in the appropriate transmit channel enable registers (XCERs).	Set XMCM to 10b.
Enable channels for symmetric transmissions and reception.	Set XMCM to 11b.

For more details, see *Transmit Multichannel Selection Modes* on page 2-8.

3.2.7 Enable/Disable the A-bis Mode

The ABIS bit determines whether the A-bis mode is on.

To get this result…	**Use this bit** …
Enable A-bis mode	ABIS (bit 6 in SPCR1) set to 1
Disable A-bis mode	ABIS set to 0

For more details, see *A-bis Mode* on page 2-13.

3.2.8 Choose 1 or 2 Phases for the Transmit Frame

To get this result …	**Use this bit**…
Set transmit frame to single phase	XPHASE (bit 15 in XCR2 register set to 0.
Set transmit frame to dual-phase frame	XPHASE bit in XCR2 set to 1.

3.2.9 Set the Transmit Word Length(s)

To get this result …	**Use this bit**…	
Specify the length of every transmit word in Frame Phase 1	Set XWDLEN1 and XWDLEN2 (bits 7:5 of XCR1 XCR2 registers, respectively) to one of the following:	
Transmit Word Length of Frame Phase 1	000b	8 bits
	001b	12 bits
	010b	16 bits
	011b	20 bits
	100b	24 bits
	101b	32 bits
	11xb	Reserved

Each frame can have one or two phases, depending on the value that you load into the RPHASE bit. If a single-phase frame is selected, XWDLEN1 selects the length for every serial word transmitted in the frame. If a dual-phase frame is selected, XWDLEN1 determines the length of the serial words in phase 1 of the frame, and XWDLEN2 determines the word length in phase 2 of the frame.

3.2.10 Set the Transmit Frame Length

The transmit frame length is the number of serial words in the transmit frame. Each frame can have one or two phases, depending on value that you load into the XPHASE bit.

If a single-phase frame is selected (XPHASE = 0), the frame length is equal to the length of phase 1. If a dual-phase frame is selected (XPHASE = 1), the frame length is the length of phase 1 plus the length of phase 2.

The 7-bit XFRLEN fields allow up to 128 words per phase. See Table 3−12 for a summary of how to calculate the frame length. This length corresponds to the number of words or logical time slots or channels per frame-synchronization pulse.

Table 3−12. How to Calculate Frame Length

XPHASE	XFRLEN1	XFRLEN2	Frame Length
0	0 ≤ XFRLEN1 ≤ 127	Don't care	(XFRLEN1 + 1) words
1	0 ≤ XFRLEN1 ≤ 127	0 ≤ XFRLEN2 ≤ 127	(XFRLEN1 + 1) + (XFRLEN2 + 1) words

Note: Program the XFRLEN fields with [*w minus 1*], where *w* represents the number of words per phase. For the example, if you want a phase length of 128 words in phase 1, load 127 into XFRLEN1.

To get this result …	**Use this bit**…
Set transmit frame length 1	Set XFRLEN1 (bits 14:8 of XRCR1) to one of the following: 000 0000 1 word in phase 1 000 0001 2 words in phase 1 . . . 111 1111 128 words in phase 1
Set transmit frame length 2	Set XFRLEN2 (bits 14:8 of XRCR2) to one of the following: 000 0000 1 word in phase 1 000 0001 2 words in phase 1 . . . 111 1111 128 words in phase 1

3.2.10.1 Enable/Disable the Transmit Frame-Sync Ignore Function

If a frame-synchronization (frame-sync) pulse starts the transfer of a new frame before the current frame is fully transmitted, this pulse is treated as an unexpected frame-sync pulse.

When XFIG = 1, normal transmission continues with unexpected frame-sync signals ignored.

When XFIG = 0 and an unexpected frame-sync pulse occurs, the serial port:

1) Aborts the present transmission

2) Sets XSYNCERR to 1 in SPCR2

3) Re-initiates transmission of the current word that was aborted

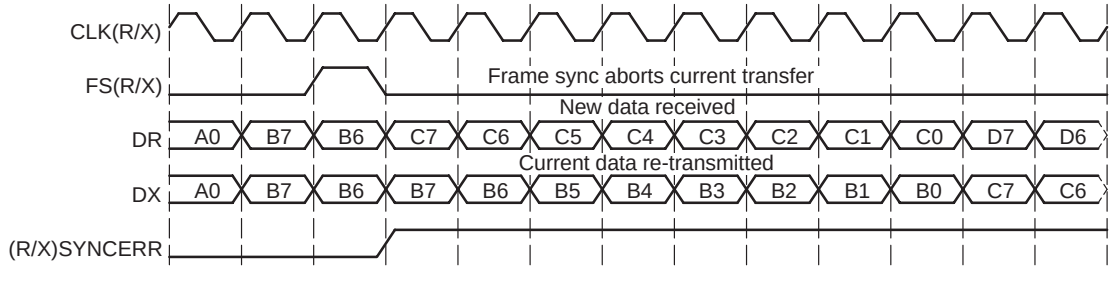
See section 5.5 for XINTM interrupt selection into FIFO logic.

If there is an unexpected receive frame-sync pulse …	**Use this bit**…
Allow an unexpected transmit frame sync pulse to restart the frame transfer	Set XFIG (bit 2 in XCR2) to 0
Ignore the unexpected pulses	Set XFIG to 1

3.2.10.2 Examples Showing the Effects of XFIG

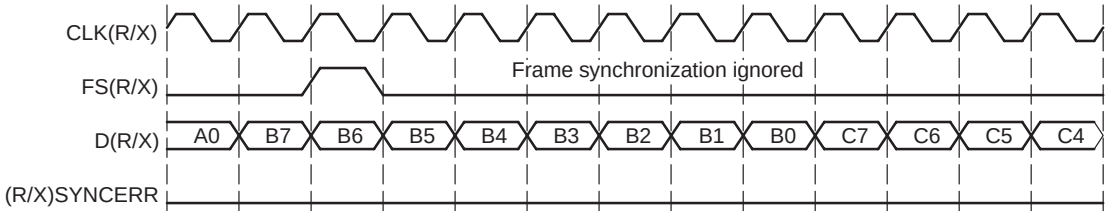
Figure 3−9 shows an example in which word B is interrupted by an unexpected frame-sync pulse when (R/X)FIG = 0. In the case of transmission, the transmission of B is aborted (B is lost). This condition is a transmit synchronization error, and thus sets the XSYNCERR bit. No new data has been written to DXR[1,2], and therefore, the McBSP transmits B again.

Figure 3−9. Unexpected Frame-Sync Pulse With (R/X)FIG = 0



In contrast with Figure 3−9, Figure 3−10 shows McBSP operation when unexpected frame-sync signals are ignored (when (R/X)FIG = 1). Here, the transfer of word B is not affected by an unexpected frame-sync pulse.

Figure 3−10. Unexpected Frame-Sync Pulse With (R/X)FIG = 1



3.2.11 Set the Transmit Companding Mode

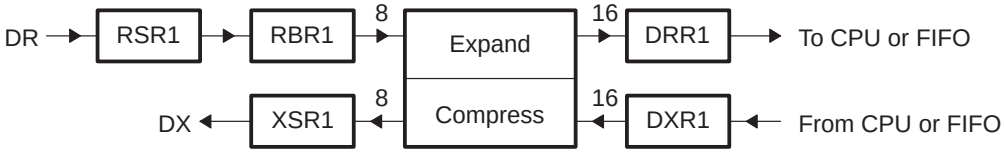
Companding (COMpressing and exPANDing) hardware allows compression and expansion of data in either μ-law or A-law format. The companding standard employed in the United States and Japan is μ-law. The European companding standard is referred to as A-law. The specifications for μ-law and A-law log PCM are part of the CCITT G.711 recommendation.

A-law and μ-law allow 13 bits and 14 bits of dynamic range, respectively. Any values outside this range are set to the most positive or most negative value. Thus, for companding to work best, the data transferred to and from the McBSP via the CPU or FIFO must be at least 16 bits wide.

The μ-law and A-law formats both encode data into 8-bit code words. Companded data is always 8 bits wide; the appropriate word length bits (RWDLEN1, RWDLEN2, XWDLEN1, XWDLEN2) must therefore be set to 0, indicating an 8-bit wide serial data stream. If companding is enabled and either of the frame phases does not have an 8-bit word length, companding continues as if the word length is 8 bits.

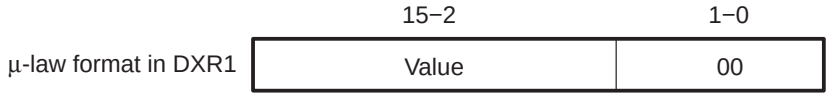
Figure 3−11 illustrates the companding processes. When companding is chosen for the transmitter, compression occurs during the process of copying data from DXR1 to XSR1. The transmit data is encoded according to the specified companding law (A-law or μ-law). When companding is chosen for the receiver, expansion occurs during the process of copying data from RBR1 to DRR1. The receive data is decoded to 2s-complement format.

Figure 3−11. Companding Processes for Reception and for Transmission



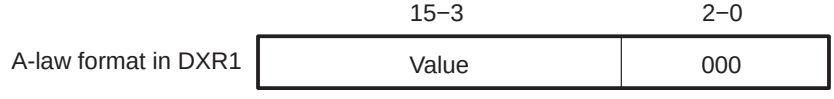
For transmission using μ-law compression, make sure the 14 data bits are left-justified in DXR1, with the remaining two low-order bits filled with 0s as shown in Figure 3−12.

Figure 3−12. μ-Law Transmit Data Companding Format



For transmission using A-law compression, make sure the 13 data bits are left-justified in DXR1, with the remaining three low-order bits filled with 0s as shown in Figure 3−13.

Figure 3−13. A-Law Transmit Data Companding Format



If the McBSP is otherwise unused (the serial port transmit and receive sections are reset), the companding hardware can compand internal data. See section 1.3.2.2 on page 1-13.

Normally, the McBSP transmit or receives all data with the most significant bit (MSB) first. However, certain 8-bit data protocols (that do not use companded data) require the least significant bit (LSB) to be transferred first. If you set XCOMPAND = 01b in XCR2, the bit ordering of 8-bit words is reversed (LSB first) before being sent from the serial port. Similar to companding, this feature is enabled only if the appropriate word length bits are set to 0, indicating that 8-bit words are to be transferred serially. If either phase of the frame does not have an 8-bit word length, the McBSP assumes the word length is eight bits, and LSB-first ordering is done.

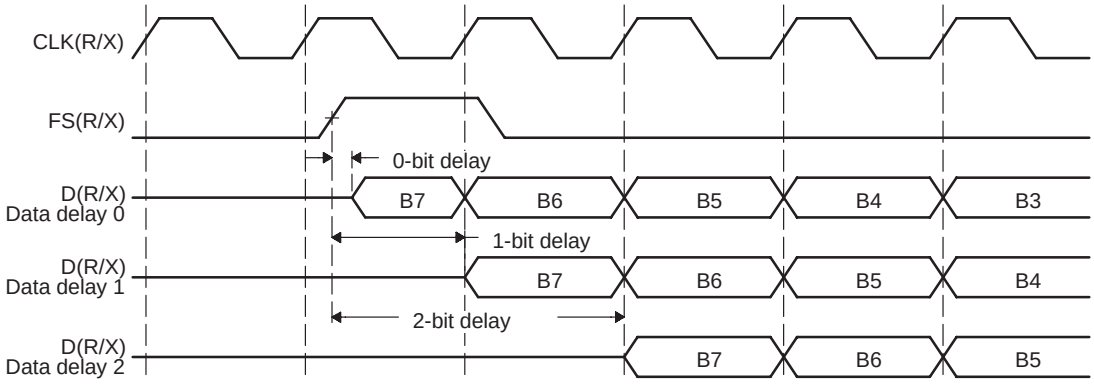
To get this result …	**Use this bit**…	
Transmit companding mode	Set XCOMPAND (bits 4:3 in XCR2) to one of the following:	
	00	No companding, any size data, MSB transmitted first
	01	No companding, 8-bit data, LSB transmitted first
	10	μ-law companding, 8-bit data, MSB received first
	11	A-law companding, 8-bit data, MSB received first

3.2.12 Set the Transmit Data Delay

The start of a frame is defined by the first clock cycle in which frame synchronization is found to be active. The beginning of actual data reception or transmission with respect to the start of the frame can be delayed if required. This delay is called data delay.

XDATDLY specifies the data delay for transmission. The range of programmable data delay is zero to two bit-clocks (XDATDLY = 00b–10b), as described in Figure 3−14. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on. Typically a 1-bit delay is selected, because data often follows a 1-cycle active frame-sync pulse.

Figure 3−14. Range of Programmable Data Delay



3.2.12.1 0-Bit Data Delay

Normally, a frame-sync pulse is detected or sampled with respect to an edge of serial clock internal CLK(R/X). Thus, on the following cycle or later (depending on the data delay value), data may be received or transmitted. However, in the case of 0-bit data delay, the data must be ready for reception and/or transmission on the same serial clock cycle.

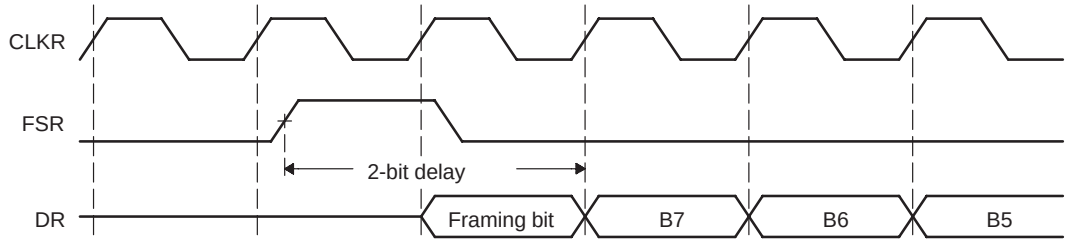
For reception this problem is solved because receive data is sampled on the first falling edge of CLKR where an active-high internal FSR is detected. However, data transmission must begin on the rising edge of the internal CLKX clock that generated the frame synchronization. Therefore, the first data bit is assumed to be present in XSR1, and thus DX. The transmitter then asynchronously detects the frame synchronization, FSX, going active high, and immediately starts driving the first bit to be transmitted on the DX pin.

3.2.12.2 2-Bit Data Delay

A data delay of two bit periods allows the serial port to interface to different types of T1 framing devices where the data stream is preceded by a framing

bit. During reception of such a stream with data delay of two bits (framing bit appears after a 1-bit delay and data appears after a 2-bit delay), the serial port essentially discards the framing bit from the data stream, as shown in the following figure. In this figure, the data transferred is an 8-bit value with bits labeled B7, B6, B5, and so on.

Figure 3−15. 2-Bit Data Delay Used to Skip a Framing Bit



To get this result …	**Use this bit**…
Set the transmit data delay	Set XDATDLY (bits 1:0 in XCR2 to:
	00 0-bit data delay
	01 1-bit data delay
	10 2-bit data delay
	11 Reserved

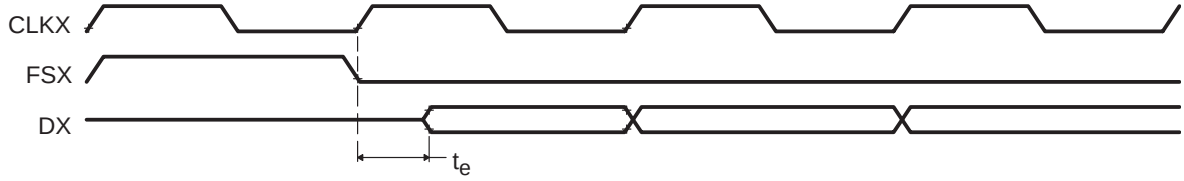
3.2.13 Set the Transmit DXENA Mode

The DXENA bit controls the delay enabler on the DX pin. Set DXENA to enable an extra delay for turn-on time (for the length of the delay, see the data sheet for your TMS320C55x DSP). Note that this bit does not control the data itself, so only the first bit is delayed, unless the A-bis mode is on. In the A-bis mode, any bit can be delayed because any bit can go from the high-impedance state to the valid state.

If you tie together the DX pins of multiple McBSPs, make sure DXENA = 1 to avoid having more than one McBSP transmit on the data line at one time.

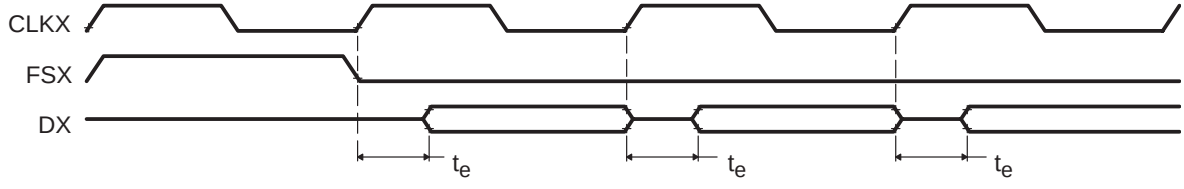
The following two figures show the timing of the DX pin for DXENA = 1. The first figure shows the effect of the DX delay enabler when the A-bis mode is off. The second figure shows the effect of the DX delay enabler when the A-bis mode is on.

Figure 3−16. DX Delay When A-bis Mode is Off



Note: te = extra delay for turn on time with DXENA = 1

Figure 3−17. DX Delays When A-bis Mode is On



Note: te = extra delay for turn on time with DXENA = 1

To get this result …	**Use this bit**…
Turn DX delay enabler mode off	Set DXENA (bit 7 in SPCR1) to 0
Turn DX delay enabler mode on	Set DXENA (bit 7 in SPCR1) to 1

3.2.14 Set the Transmit Interrupt Mode

The transmitter interrupt (XINT) signals the CPU of changes to the serial port status. Four options exist for configuring this interrupt. The options are set by the transmit interrupt mode bits, XINTM, in SPCR2.

- ❑ XINTM = 00b. Interrupt on every serial word by tracking the XRDY bit in SPCR2. Note that regardless of the value of XINTM, XRDY can be read to detect the XRDY = 1 condition.
- ❑ XINTM = 01b. In any of the transmit multichannel selection modes, interrupt after every 16-channel block boundary has been crossed within a frame and at the end of the frame. In any other serial transfer case, this setting is not applicable and, therefore, no interrupts are generated.
- ❑ XINTM = 10b. Interrupt on detection of each transmit frame-sync pulse. This generates an interrupt even when the transmitter is in its reset state. This is done by synchronizing the incoming frame-sync pulse to the CPU clock and sending it to the CPU via XINT.
- ❑ XINTM = 11b. Interrupt on frame-synchronization error. Note that regardless of the value of XINTM, XSYNCERR can be read to detect this

condition. For more information on using XSYNCERR, see *Unexpected Transmit Frame-Sync Pulse* on page 1-46.

To get this result …	Use this bit…	
Set the transmit interrupt mode	Set XINTM (bits 5:4 in SPCR2 to:	
	00	XINT generated when XRDY changes from 0 to 1
	01	XINT generated by an end-of-block or end-of-frame condition in the transmit multichannel selection mode
	10	XINT generated by a new transmit frame-sync pulse
	11	XINT generated when XSYNCERR is set

3.2.15 Set the Transmit Frame-Sync Mode

Table 3−13 shows how FSXM and FSGM select the source of transmit frame-sync pulses. The three choices are:

- External frame-sync input
- Sample rate generator frame-sync signal (FSG).
- Internal signal that indicates a DXR-to-XSR copy has been made

Table 3−13 also shows the effect of each bit setting on the FSX pin. The polarity of the signal on the FSX pin is determined by the FSXP bit.

Table 3−13. How FSXM and FSGM Select the Source of Transmit Frame-Sync Pulses

FSXM	FSGM	Source of Transmit Frame Synchronization	FSX Pin Status
0	0 or 1	An external frame-sync signal enters the McBSP through the FSX pin. The signal is then inverted by FSXP before being used as internal FSX.	Input
1	1	Internal FSX is driven by the sample rate generator frame-sync signal (FSG).	Output. FSG is inverted by FSXP before being driven out on FSX pin.
1	0	A DXR-to-XSR copy causes the McBSP to generate a transmit frame-sync pulse that is 1 cycle wide.	Output. The generated frame-sync pulse is inverted as determined by FSXP before being driven out on FSX pin.

If the sample rate generator creates a frame-sync signal (FSG) that is derived from an external input clock, the GSYNC bit determines whether FSG is kept synchronized with pulses on the FSR pin. For more details, see *Synchronizing Sample Rate Generator Outputs to an External Clock* on page 1-31.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master and must provide a slave-enable signal ($\overline{SS}$) on the FSX pin, make sure that FSXM = 1 and FSGM = 0, so that FSX is an output and is driven active for the duration of each transmission. If the McBSP is a slave, make sure that FSXM = 0, so that the McBSP can receive the slave-enable signal on the FSX pin.

To get this result …	**Use this bit**…
To supply transmit frame synchronization mode by an external source via the FSX pin	Set FSXM (bit 11 in PCR) to 0.
To have the McBSP supply the transmit frame synchronization mode by way of the FSGM bit of SRGR2.	Set FSXM to 1.
Have the McBSP generate a transmit frame-sync pulse when the content of DXR is copied to XSR	Set FSGM (bit 12 of SRGR2) to 0. This mode is used when FSXM = 1.
Use frame-sync pulses generated by the sample rate generator	Set FSGM (bit 12 of SRGR2) to 1.

3.2.16 Set the Transmit Frame-Sync Polarity

Transmit frame-sync pulses can be either generated internally by the sample rate generator (see section 1.4.2 on page 1-30) or driven by an external source. The source of frame sync is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see *Set the Transmit Frame-Sync Mode* on page 3-39). Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see *Set the Transmit Clock Mode* on page 3-43).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. Note that these internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

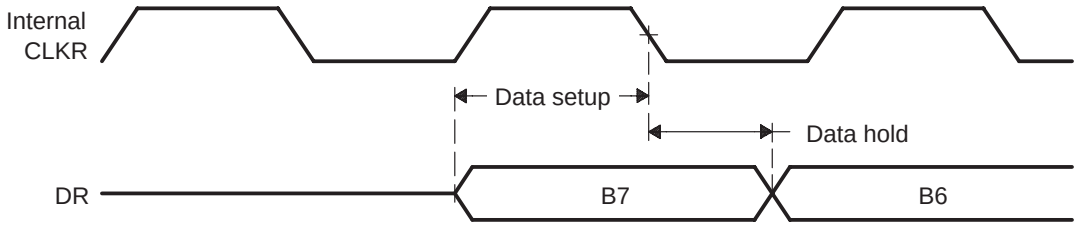
FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-sync signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-sync signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-sync signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin. The figure in the topic Clock and Frame Generation shows this inversion using XOR gates.

On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Note that data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1, and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. Note that the receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1, and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 3−18 shows how data clocked by an external serial device using a rising edge, may be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 3−18. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



To get this result …	Use this bit…
Set frame-synchronization pulse FSX to active high	Set FSXP (bit 3 in the PCR register) to 0
Set frame-synchronization pulse FSX to active low	Set FSXP (bit 3 in the PCR register) to 1

3.2.17 Set the SRG Frame-Sync Period and Pulse Width

To get this result …	Use this bit…
Set the sample rate generator frame-sync period	For the frame-sync signal FSG, (FPER + 1) determines the period from the start of a frame-sync pulse to the start of the next frame-sync pulse. Set FPER (bits 11−0 in the SRGR2 register) from 1 to 4096 cycles.
Set the sample rate generator frame-sync pulse width	FWID (bits 15−8 in SRGR1) plus 1 determines the width of each frame-sync pulse (from 1 to 256 CLKG cycles).

The sample rate generator can produce a clock signal, CLKG, and a frame-sync signal, FSG. If the sample rate generator is supplying receive or transmit frame synchronization, you must program the bit fields FPER and FWID.

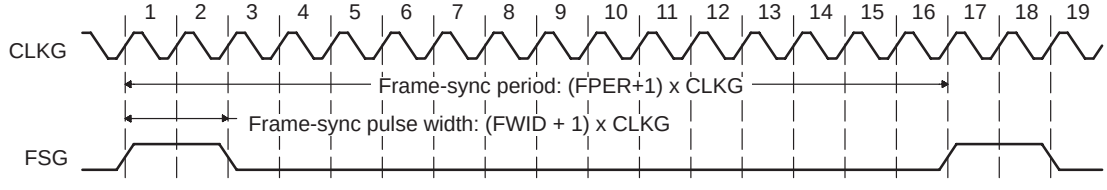
On FSG, the period from the start of a frame-sync pulse to the start of the next pulse is (FPER + 1) CLKG cycles. The 12 bits of FPER allow a frame-sync period of 1 to 4096 CLKG cycles, which allows up to 4096 data bits per frame. When GSYNC = 1, FPER is a don’t care value.

Each pulse on FSG has a width of (FWID + 1) CLKG cycles. The eight bits of FWID allow a pulse width of 1 to 256 CLKG cycles. It is recommended that FWID be programmed to a value less than the programmed word length.

The values in FPER and FWID are loaded into separate down-counters. The 12-bit FPER counter counts down the generated clock cycles from the programmed value (4095 maximum) to 0. The 8-bit FWID counter counts down from the programmed value (255 maximum) to 0.

Figure 3−19 shows a frame-sync period of 16 CLKG periods (FPER = 15 or 00001111b) and a frame-sync pulse with an active width of 2 CLKG periods (FWID = 1).

Figure 3−19. Frame of Period 16 CLKG Periods and Active Width of 2 CLKG Periods



When the sample rate generator comes out of reset, FSG is in its inactive state. Then, when $\overline{\text{FRST}}$ = 1 and FSGM = 1, a frame-sync pulse is generated. The frame width value (FWID + 1) is counted down on every CLKG cycle until it reaches 0, at which time FSG goes low. At the same time, the frame period value (FPER + 1) is also counting down. When this value reaches 0, FSG goes high, indicating a new frame.

3.2.18 Set the Transmit Clock Mode

To get this result …	**Use this bit**…
Set the transmitter to get its clock signal from an external source via the CLKX pin.	Set CLKXM (bit 9 in the PCR register) to 0
Set the CLKX output pin driven by the sample rate generator	Set CLKXM (bit 9 in the PCR register) to 1

3.2.18.1 Selecting a Source for the Transmit Clock and a Data Direction for the CLKX Pin

Table 3−14 shows how the CLKXM bit selects the transmit clock and the corresponding status of the CLKX pin. The polarity of the signal on the CLKX pin is determined by the CLKXP bit.

Table 3−14. How the CLKXM Bit Selects the Transmit Clock and the Corresponding Status of the CLKX Pin

CLKXM in PCR	Source of Transmit Clock	CLKX Pin Status
0	Internal CLKX is driven by an external clock on the CLKX pin. CLKX is inverted as determined by CLKXP before being used.	Input
1	Internal CLKX is driven by the sample rate generator clock, CLKG.	Output. CLKG, inverted as determined by CLKXP, is driven out on CLKX.

If the sample rate generator creates a clock signal (CLKG) that is derived from an external input clock, the GSYNC bit determines whether CLKG is kept synchronized with pulses on the FSR pin. For more details, see *Synchronizing Sample Rate Generator Outputs to an External Clock* on page 1-31.

In the clock stop mode (CLKSTP = 10b or 11b), the McBSP can act as a master or as a slave in the SPI protocol. If the McBSP is a master, make sure that CLKXM = 1, so that CLKX is an output to supply the master clock to any slave devices. If the McBSP is a slave, make sure that CLKXM = 0, so that CLKX is an input to accept the master clock signal.

3.2.18.2 Set the Transmit Clock Polarity

To get this result …	**Use this bit**…
Sample transmit data on the rising edge of CLKX	Set CLKXP (bit 1 of PCR) to 0.
Sample transmit data on the falling edge of CLKX	Set CLKXP (bit 1 of PCR) to 1.

Transmit frame-sync pulses can be either generated internally by the sample rate generator (see section 1.4.2 on page 1-30) or driven by an external source. The source of frame sync is selected by programming the mode bit, FSXM, in PCR. FSX is also affected by the FSGM bit in SRGR2. For information about the effects of FSXM and FSGM, see *Set the Transmit Frame-Sync Mode* . Similarly, transmit clocks can be selected to be inputs or outputs by programming the mode bit, CLKXM, in the PCR (see *Set the Transmit Clock Mode* on page 3-43).

When FSR and FSX are inputs (FSXM = FSRM= 0, external frame-sync pulses), the McBSP detects them on the internal falling edge of clock, internal

CLKR, and internal CLKX, respectively. The receive data arriving at the DR pin is also sampled on the falling edge of internal CLKR. Note that these internal clock signals are either derived from external source via CLK(R/X) pins or driven by the sample rate generator clock (CLKG) internal to the McBSP.

When FSR and FSX are outputs, implying that they are driven by the sample rate generator, they are generated (transition to their active state) on the rising edge of internal clock, CLK(R/X). Similarly, data on the DX pin is output on the rising edge of internal CLKX.

FSRP, FSXP, CLKRP, and CLKXP in the pin control register (PCR) configure the polarities of the FSR, FSX, CLKR, and CLKX signals, respectively. All frame-sync signals (internal FSR, internal FSX) that are internal to the serial port are active high. If the serial port is configured for external frame synchronization (FSR/FSX are inputs to McBSP), and FSRP = FSXP = 1, the external active-low frame-sync signals are inverted before being sent to the receiver (internal FSR) and transmitter (internal FSX). Similarly, if internal synchronization (FSR/FSX are output pins and GSYNC = 0) is selected, the internal active-high frame-sync signals are inverted, if the polarity bit FS(R/X)P = 1, before being sent to the FS(R/X) pin. The figure in the topic Clock and Frame Generation shows this inversion using XOR gates.

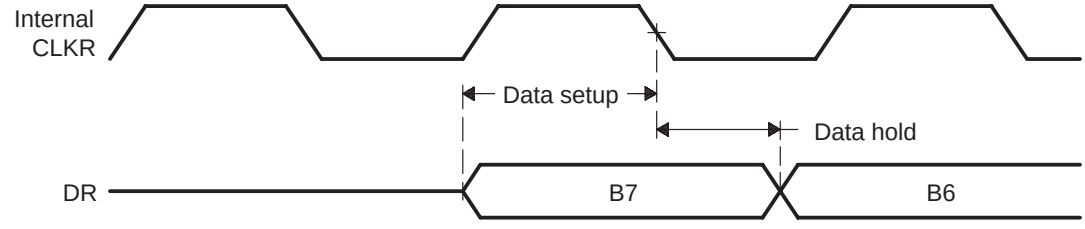
On the transmit side, the transmit clock polarity bit, CLKXP, sets the edge used to shift and clock out transmit data. Note that data is always transmitted on the rising edge of internal CLKX. If CLKXP = 1, and external clocking is selected (CLKXM = 0 and CLKX is an input), the external falling-edge triggered input clock on CLKX is inverted to a rising-edge triggered clock before being sent to the transmitter. If CLKXP = 1, and internal clocking selected (CLKXM = 1 and CLKX is an output pin), the internal (rising-edge triggered) clock, internal CLKX, is inverted before being sent out on the CLKX pin.

Similarly, the receiver can reliably sample data that is clocked with a rising edge clock (by the transmitter). The receive clock polarity bit, CLKRP, sets the edge used to sample received data. Note that the receive data is always sampled on the falling edge of internal CLKR. Therefore, if CLKRP = 1 and external clocking is selected (CLKRM = 0 and CLKR is an input pin), the external rising-edge triggered input clock on CLKR is inverted to a falling-edge triggered clock before being sent to the receiver. If CLKRP = 1, and internal clocking is selected (CLKRM = 1), the internal falling-edge triggered clock is inverted to a rising-edge triggered clock before being sent out on the CLKR pin.

Note that CLKRP = CLKXP in a system where the same clock (internal or external) is used to clock the receiver and transmitter. The receiver uses the opposite edge as the transmitter to ensure valid setup and hold of data around this edge. Figure 3−20 shows how data clocked by an external serial device

using a rising edge, may be sampled by the McBSP receiver on the falling edge of the same clock.

Figure 3−20. Data Clocked Externally Using a Rising Edge and Sampled by the McBSP Receiver on a Falling Edge



3.2.19 Set the SRG Clock Divide-Down Value

To get this result …	**Use this bit**…
Set the divide down value of the sample rate generator clock	Set CLKGDV (bits 7−0 of SRGR1) to generate the required SRG clock frequency. The default value is 1 (divide input clock by 2).

The first divider stage generates the serial data bit clock from the input clock. This divider stage utilizes a counter, preloaded by CLKGDV, that contains the divide ratio value.

The output of the first divider stage is the data bit clock, which is output as CLKG and which serves as the input for the second and third stages of the divider.

CLKG has a frequency equal to 1/(CLKGDV + 1) of sample rate generator input clock. Thus, the sample generator input clock frequency is divided by a value between 1 and 256. When CLKGDV is odd or equal to 0, the CLKG duty cycle is 50%. When CLKGDV is an even value, 2p, representing an odd divide-down, the high-state duration is p+1 cycles and the low-state duration is p cycles.

3.2.19.1 Set the SRG Clock Synchronization Mode

GSYNC is used only when the input clock source for the sample rate generator is external—on the CLKR or CLKX pin.

For more details on using the clock synchronization feature, see *Synchronizing Sample Rate Generator Outputs to an External Clock* on page 1-31.

To get this result …	Use this bit…
The sample rate generator clock (CLKG) is free running. CLKG oscillates without adjustment, and FSG pulses every (FPER + 1) CLKG cycles.	Set GSYNC (bit 15 of SRGR2) to 0
Clock synchronization is performed. When a pulse is detected on the FSR pin: ❑ CLKG is adjusted as necessary so that it is synchronized with the input clock on the CLKR or CLKX pin. ❑ FSG pulses. FSG **only** pulses in response to a pulse on the FSR pin. The frame-sync period defined in FPER is ignored.	Set GSYNC (bit 15 of SRGR2) to 1

3.2.20 Set the SRG Clock Mode (Choose an Input Clock)

To get this result …	Use this bit…
Sample rate generator clock derived from CLKS pin	Set SCLKME (bit 7 of PCR) = 0 and CLKSM (bit 13 of SRGR2) = 0
Sample rate generator clock derived from CPU clock (This is the condition forced by a DSP reset.)	SCLKME = 0 CLKSM = 1
Sample rate generator clock derived from CLKR pin	SCLKME = 1 CLKSM = 0
Sample rate generator clock derived from CLKX pin	SCLKME = 1 CLKSM = 1

The sample rate generator can produce a clock signal (CLKG) for use by the receiver, the transmitter, or both, but CLKG is derived from an input clock. The preceding table shows the four possible sources of the input clock. For more details on generating CLKG, see *Clock Generation in the Sample Rate Generator* on page 1-27.

3.2.20.1 Set the SRG Input Clock Polarity

CLKSP determines the input clock polarity when the CLKS pin supplies the input clock (SCLKME = 0 and CLKSM = 0).

To get this result …	Use this bit…
Rising edge on CLKS pin generates CLKG and FSG.	Set CLKSP (bit 14 of SRGR2) to 0
Falling edge on CLKS pin generates CLKG and FSG.	Set CLKSP (bit 14 of SRGR2) to 1

CLKXP determines the input clock polarity when the CLKX pin supplies the input clock (SCLKME = 1 and CLKSM = 1).

To get this result …	Use this bit…
Rising edge on CLKX pin generates transitions on CLKG and FSG.	Set CLKXP (bit 1 of PCR) to 0
Falling edge on CLKX pin generates transitions on CLKG and FSG.	Set CLKXP to 1

CLKRP determines the input clock polarity when the CLKR pin supplies the input clock (SCLKME = 1 and CLKSM = 0).

To get this result …	Use this bit…
Rising edge on CLKR pin generates transitions on CLKG and FSG.	Set CLKRP (bit 0 of PCR) to 1
Falling edge on CLKR pin generates transitions on CLKG and FSG.	Set CLKRP to 0

3.2.20.2 Using CLKXP/CLKRP to Choose an Input Clock Polarity

The sample rate generator can produce a clock signal (CLKG) and a frame-sync signal (FSG) for use by the receiver, the transmitter, or both. To produce CLKG and FSG, the sample rate generator must be driven by an input clock signal derived from the CPU clock or from an external clock on the CLKX, or CLKR pin. If you use a pin, choose a polarity for that pin by using the appropriate polarity bit (CLKXP for the CLKX pin, CLKRP for the CLKR pin). The polarity determines whether the rising or falling edge of the input clock generates transitions on CLKG and FSG.

Chapter 4

Emulation and Reset Considerations

This section covers emulation mode and how to reset and initialize the various parts of the McBSP.

Topic **Page**

4.1 McBSP Emulation Mode

FREE and SOFT are special emulation bits in SPCR2 that determine the state of the McBSP when a breakpoint is encountered in the high-level language debugger. If FREE = 1, upon a software breakpoint the clock continues to run and data is still shifted out. When FREE = 1, the SOFT bit is a *don't care.*

If FREE = 0, the SOFT bit takes effect: If SOFT = 0 when breakpoint occurs, the clock stops immediately, thus aborting a transmission. If SOFT = 1 and a breakpoint occurs while transmission is in progress, the transmission continues until completion of the transfer, and then the clock halts. These options are listed in the following table.

The McBSP receiver functions in a similar fashion. Note that if a mode other than the immediate stop mode (SOFT = FREE = 0) is chosen, the receiver continues running and an overrun error is possible.

Table 4−1. McBSP Emulation Modes Selectable With the FREE and SOFT Bits of SPCR2

FREE	SOFT	McBSP Emulation Mode
0	0	Immediate stop mode (reset condition) The transmitter or receiver stops immediately in response to a breakpoint.
0	1	Soft stop mode When a breakpoint occurs, the transmitter stops after completion of the current word. The receiver is not affected.
1	0 or 1	Free run mode The transmitter and receiver continue to run when a breakpoint occurs.

4.1.1 Resetting and Initializing a McBSP

Table 4−2 shows the state of McBSP pins when the serial port is reset due to a DSP reset and due to a direct receiver or transmitter reset.

Table 4−2. Reset State of Each McBSP Pin

Pin	Possible State(s)	State Forced By DSP Reset	State Forced By Receiver/Transmitter Reset
			Receiver Reset ($\overline{\text{RRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DR	I	Input	Input
CLKR	I/O/Z	Input	Known state if Input; CLKR running if output
FSR	I/O/Z	Input	Known state if Input; FSRP inactive state if output
			Transmitter Reset ($\overline{\text{XRST}}$ = 0 and $\overline{\text{GRST}}$ = 1)
DX	O/Z	High impedance	High impedance
CLKX	I/O/Z	Input	Known state if Input; CLKX running if output
FSX	I/O/Z	Input	Known state if Input; FSXP inactive state if output

Note: In Possible State(s) column, I = Input, O = Output, Z = High impedance
In the 28x family, at device reset, all I/Os default to GPIO function and generally as inputs.

When the McBSP is reset in either of the above two ways, the machine is reset to its initial state, including reset of all counters and status bits. The receive status bits include RFULL, RRDY, and RSYNCERR. The transmit status bits include $\overline{\text{XEMPTY}}$, XRDY, and XSYNCERR.

- **DSP reset.** When the whole DSP is reset ($\overline{\text{XRSN}}$ signal is driven low), the entire serial port, including the transmitter, receiver, and the sample rate

generator, is reset. All input-only pins and three-state pins should be in a known state. The output-only pin DX is in the high-impedance state.

The DSP reset forces the sample rate generator clock, CLKG, to have the frequency of LSPCLK clock. No pulses are generated on the sample rate generator's frame-sync signal, FSG.

When the device is pulled out of reset, the serial port remains in the reset state. In this state the DR and DX pins may be used as general-purpose I/O pins as described in section 4.3 on page 4-11.

- **McBSP reset.** When the receiver and transmitter reset bits, $\overline{\text{RRST}}$ and $\overline{\text{XRST}}$, are loaded with 0s, the respective portions of the McBSP are reset, and activity in the corresponding section of the serial port stops. All input-only pins, such as DR, and all other pins that are configured as inputs, are in a known state. The FSR and FSX pins are driven to their inactive state if they are not outputs. If the CLKR and CLKX pins are programmed as outputs, they will be driven by CLKG, provided that $\overline{\text{GRST}}$ = 1. Lastly, the DX pin will be in the high-impedance state when the transmitter and/or the device is reset.

  During normal operation, the sample rate generator is reset if the $\overline{\text{GRST}}$ bit is cleared. $\overline{\text{GRST}}$ should be 0 only when neither the transmitter nor the receiver is using the sample rate generator. In this case, the internal sample rate generator clock (CLKG) and its frame-sync signal (FSG) are driven inactive low.

  When the sample rate generator is not in the reset state ($\overline{\text{GRST}}$ = 1), pins FSR and FSX are in an inactive state when $\overline{\text{RRST}}$ = 0 and $\overline{\text{XRST}}$ = 0, respectively, even if they are outputs driven by FSG. This ensures that when only one portion of the McBSP is in reset, the other portion can continue operation when $\overline{\text{FRST}}$ = 1 and its frame synchronization is driven by FSG.

- **Sample rate generator reset.** The sample rate generator is reset when the DSP is reset or when $\overline{\text{GRST}}$ is loaded with 0. In the case of a DSP reset, the sample rate generator clock, CLKG, is driven by the LSPCLK, and the frame-sync signal, FSG, is driven inactive low.

  When neither the transmitter nor the receiver is fed by CLKG and FSG, you can reset the sample rate generator by clearing $\overline{\text{GRST}}$. In this case, CLKG and FSG are driven inactive low. If you then set $\overline{\text{GRST}}$, CLKG starts and runs as programmed. Later, if $\overline{\text{FRST}}$ = 1, FSG pulses active high after the programmed number of CLKG cycles has elapsed.

4.1.1.1 McBSP Initialization Procedure

The serial port initialization procedure is as follows:

1) Make $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = $\overline{\text{FRST}}$ = 0 in SPCR[1,2]. If coming out of a DSP reset, this step is not required.

2) While the serial port is in the reset state, program only the McBSP configuration registers (not the data registers) as required.

3) Wait for two clock cycles. This ensures proper internal synchronization.

4) Set up data acquisition as required (such as writing to DXR[1,2]).

5) Make $\overline{\text{XRST}}$ = $\overline{\text{RRST}}$ = 1 to enable the serial port. Make sure that as you set these reset bits, you do not modify any of the other bits in SPCR1 and SPCR2. Otherwise, you will change the configuration you selected in step 2.

6) Set $\overline{\text{FRST}}$ = 1, if internally generated frame synchronization is required.

7) Wait two clock cycles for the receiver and transmitter to become active.

Alternatively, on either write (step 1 or 5), the transmitter and receiver may be placed in or taken out of reset individually by modifying the desired bit.

The above procedure for reset/initialization can be applied in general when the receiver or transmitter has to be reset during its normal operation, and also when the sample rate generator is not used for either operation.

Notes:

1) The necessary duration of the active-low period of $\overline{\text{XRST}}$ or $\overline{\text{RRST}}$ is at least two CLKR/CLKX cycles.

2) The appropriate bits in serial port configuration registers SPCR[1,2], PCR, RCR[1,2], XCR[1,2], and SRGR[1,2] should only be modified when the affected portion of the serial port is in its reset state.

3) In most cases, the data transmit registers (DXR[1,2]) should be loaded by the CPU or by the or FIFO only when the transmitter is enabled ($\overline{\text{XRST}}$ = 1). An exception to this rule is when these registers are used for companding internal data (see section 1.3.2.2 on page 1-13).

4) The bits of the channel control registers—MCR[1,2], RCER[A–H], XCER[A–H]—can be modified at any time as long as they are not being used by the current reception/transmission in a multichannel selection mode.

4.1.1.2 Example: Resetting the Transmitter While the Receiver is Running

The following example shows values in the control registers that reset and configure the transmitter while the receiver is running.

Example 4−1. Resetting and Configuring the McBSP Transmitter While the McBSP Receiver is Running − Non-FIFO Mode

```
SPCR1 = 0001h      ; The receiver is running with the receive
SPCR2 = 0030h      ; interrupt (RINT) triggered by the
                   ; receiver ready bit (RRDY). The
                   ; transmitter is in its reset state. The
                   ; transmit interrupt (XINT) will be
                   ; triggered by the transmit frame-sync
                   ; error bit (XSYNCERR).

PCR = 0900h        ; Transmit frame synchronization is
                   ; generated internally according to the
                   ; FSGM bit of SRGR2. The transmit clock
                   ; is driven by an external source. The
                   ; receive clock continues to be driven by
                   ; sample rate generator. The input clock
                   ; of the sample rate generator is supplied
                   ; by the CLKS pin or by the CPU clock
                   ; depending on the CLKSM bit of SRGR2.

SRGR1 = 0001h      ; The CPU clock is the input clock for
SRGR2 = 2000h      ; the sample rate generator. The sample
                   ; rate generator divides the CPU clock by
                   ; 2 to generate its output clock (CLKG).
                   ; Transmit frame synchronization is tied
                   ; to the automatic copying of data from
                   ; the DXR(s) to the XSR(s).
```

```
XCR1 = 0740h       ; The transmit frame has two phases.
XCR2 = 8321h       ; Phase 1 has eight 16-bit words. Phase 2
                   ; has four 12-bit words. There is 1-bit
                   ; data delay between the start of a
                   ; frame-sync pulse and the first data bit
                   ; transmitted.

SPCR2 = 0x0031     ; The transmitter is taken out of reset.
```

4.2 Data Packing Examples

This section shows two ways you can implement data packing in the McBSP.

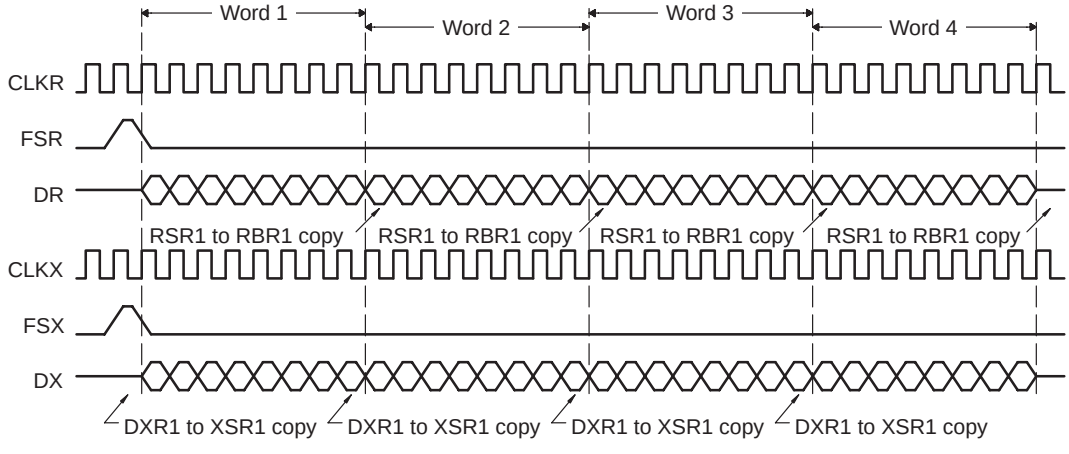
4.2.1 Data Packing Using Frame Length and Word Length

The frame length and word length can be manipulated to effectively pack data. For example, consider a situation where four 8-bit words are transferred in a single-phase frame as shown in Figure 4−1. In this case:

- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000011b: 4-word frame
- (R/X)WDLEN1 = 000b: 8-bit words

Four 8-bit data words are transferred to and from the McBSP by the CPU or by the FIFO. Thus, four reads from DRR1 and four writes to DXR1 are necessary for each frame.

Figure 4−1. Four 8-Bit Data Words Transferred To/From the McBSP



This data can also be treated as a single-phase frame consisting of one 32-bit data word, as shown in Figure 4−2. In this case:

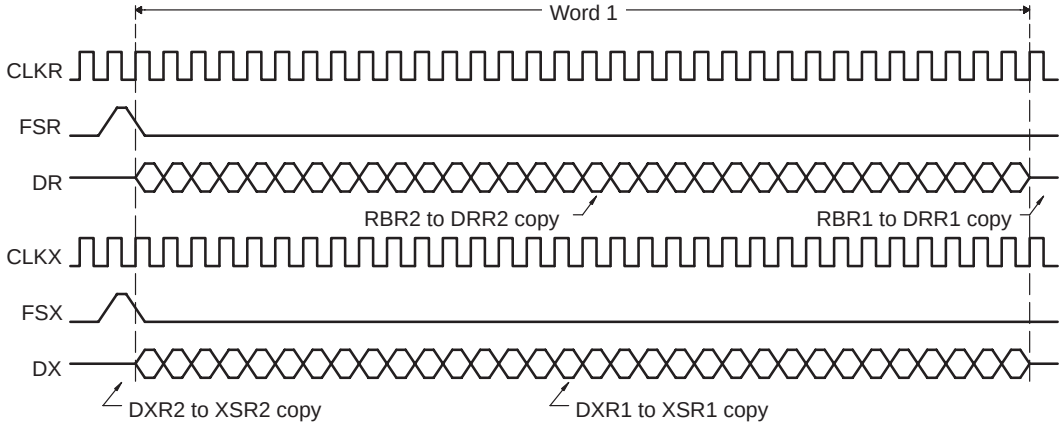
- (R/X)PHASE = 0: Single-phase frame
- (R/X)FRLEN1 = 0000000b: 1-word frame
- (R/X)WDLEN1 = 101b: 32-bit word

Two 16-bit data words are transferred to and from the McBSP by the CPU or FIFO. Thus, two reads, from DRR2 and DRR1, and two writes, to DXR2 and DXR1, are necessary for each frame. This results in only half the number of transfers compared to the previous case. This manipulation reduces the percentage of bus time required for serial port data movement.

Note:

When the word length is larger than 16 bits, make sure you access DRR2/DXR2 before you access DRR1/DXR1. McBSP activity is tied to accesses of DRR1/DXR1. During the reception of 24-bit or 32-bit words, read DRR2 and then read DRR1. Otherwise, the next RBR[1,2]-to-DRR[1,2] copy occurs before DRR2 is read. Similarly, during the transmission of 24-bit or 32-bit words, write to DXR2 and then write to DXR1. Otherwise, the next DXR[1,2]-to-XSR[1,2] copy occurs before DXR2 is loaded with new data.

Figure 4−2. One 32-Bit Data Word Transferred To/From the McBSP



4.2.2 Data Packing Using Word Length and the Frame-Sync Ignore Function

When there are multiple words per frame, you can implement data packing by increasing the word length (defining a serial word with more bits) and by ignoring frame-sync pulses. First, consider Figure 4−3, which shows the McBSP operating at the maximum packet frequency. Here, each frame only has a single 8-bit word. Note the frame-sync pulse that initiates each frame transfer for reception and for transmission. For reception, this configuration requires one read operation for each word. For transmission, this configuration requires one write operation for each word.

Figure 4−3. 8-Bit Data Words Transferred at Maximum Packet Frequency

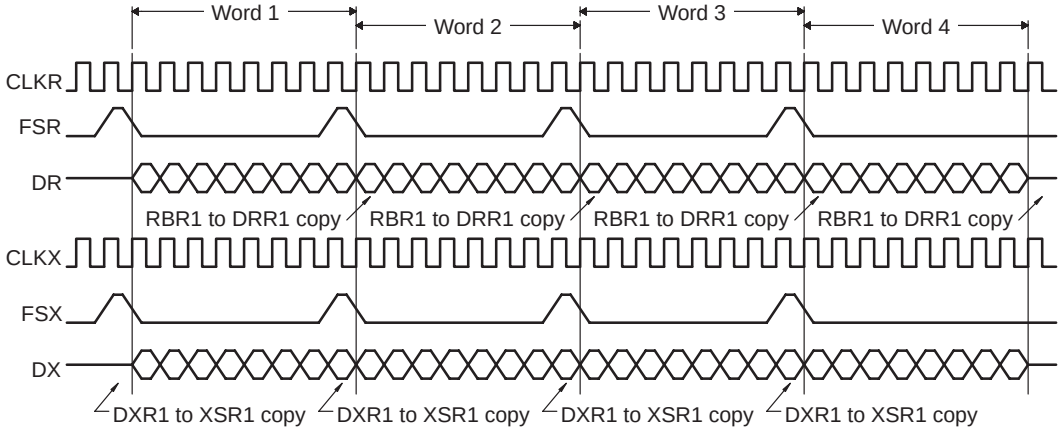
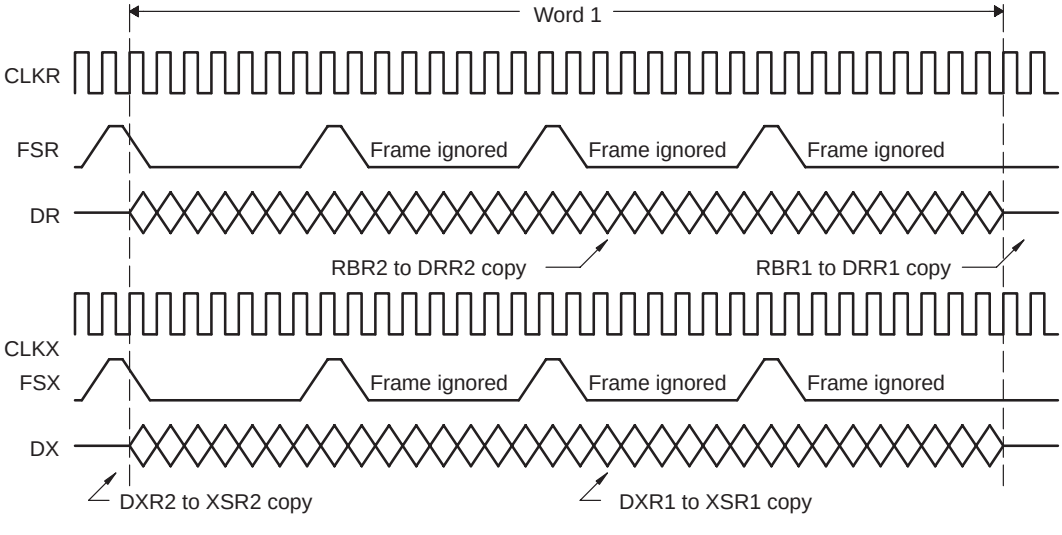


Figure 4−4 shows the McBSP configured to treat this data stream as a continuous 32-bit word. In this example, the McBSP responds to an initial frame-sync pulse. However, (R/X)FIG = 1 so that the McBSP ignores subsequent pulses. Only two read transfers or two write transfers are needed every 32 bits. This configuration effectively reduces the required bus bandwidth to half the bandwidth needed to transfer four 8-bit words.

Figure 4−4. Configuring the Data Stream of Figure 4−3 as a Continuous 32-Bit Word



4.3 GPIO Function

Note: General-purpose I/O Selection

Do not use the GPIO function using RIOEN/XIOEN bits 12 and 13 in the PCR register. This feature is not applicable to the 28x McBSP implementation; therefore, these bits are reserved on 28x devices.

Chapter 5

McBSP FIFO and Interrupts

This chapter describes the FIFO interface logic and how interrupts are handled in the McBSP module.

Topic **Page**

5.1 McBSP FIFO Overview

McBSP module is interfaced with a 16x16 bit (16 level) FIFO for each of the data registers, DRR2/DRR1and DXR2/DXR1. The top of the FIFO registers share the same addresses as of the data registers in nonFIFO mode. Each of the FIFO data registers pair has a separate set of control registers. Table 5−1 lists the register details for easy reference.

Table 5−1. McBSP FIFO Registers

Address Offset 0x000 xxh	16-Bit Access	Type R/W †	Reset Value (Hex)	McBSP FIFO Registers
				FIFO mode registers applicable only for FIFO mode in 28xx implementation
00	DRR2	R	0x0000	McBSP Data Receive Register2 −Top of receive FIFO Read first FIFO index will not advance
01	DRR1	R	0x0000	McBSP Data Receive Register1−Top of receive FIFO Read second for FIFO index to advance
02	DXR2	W	0x0000	McBSP Data Transmit Register2−Top of transmit FIFO Write first FIFO index will not advance
03	DXR1	W	0x0000	McBSP Data Transmit Register1−Top of transmit FIFO Write second for FIFO index to advance
20	MFFTX	R/W	0xA000	McBSP transmit FIFO register
21	MFFRX	R/W	0x201F	McBSP receive FIFO register
22	MFFCT	R/W	0x0000	McBSP FIFO control register
23	MFFINT	R/W	0x0000	McBSP FIFO interrupt register
24	MFFST	R/W	0x0000	McBSP FIFO status register

† R = read, W = write

This section describes the following requirements for the McBSP module in FIFO mode.

1) McBSP functionality and limitation under FIFO mode
2) McBSP FIFO operation
3) McBSP receive interrupt generation
4) McBSP transmit interrupt generation
5) McBSP FIFO access constraints
6) McBSP FIFO register description

5.2 McBSP Functionality and Limitation Under FIFO Mode

McBSP, in its normal mode, communicates with various types of Codecs with variable word size. This mode is common and is available while the FIFO is enabled. Apart from this mode, the McBSP uses time-division multiplexed (TDM) data stream while communicating with other McBSPs or serial devices. The multichannel mode provides flexibility while transmitting/receiving selected channels or all the channels in a TDM stream.

Table 5−2 provides a quick reference to McBSP mode selection and its limitation if any in FIFO mode.

Table 5−2. McBSP Mode Selection

No.	McBSP Word Size	Register Bits That Decide the Mode Selection					Description of the Mode and its Function
		MCR1 bit 9,0		MCR2 bit 9,1,0		SPCR1 –6	
		RMCME	RMCM	XMCME	XMCM	A-bis	
							Normal Mode
1	8/12/16/24/32 bit words	0	0	0	00	0	All types of Codec interface will use this selection
							Multichannel Mode
2	8-bit words						**2 Partition** or 32-channel Mode
		0	1	0	01	0	All channels are disabled,unless selected in X/RCERA/B
		0	1	0	10	0	All channels are enabled,but masked unless selected in X/RCERA/B
		0	1	0	11	0	Symmetric transmit, receive
							8 Partition or 128 Channel Mode Transmit/ Receive
							Channels selected by X/RCERA to X/RCERH bits
							Multichannel Mode is ON
		1	1	1	01	0	All channels are disabled,unless selected in XCERs
		1	1	1	10	0	All channels are enabled,but masked unless selected in XCERs
		1	1	1	11	0	Symmetric transmit, receive
							Continuous Mode − Transmit
		1	0	1	00	0	**Multi-Channel Mode is OFF**
							All 128 channels are active and enabled
							A-bis Mode
4	16-bit words	0	0	0	0	1	A-bis mode. McBSP can receive /transmit up to 1024 bits on PCM link. Bit selection is provide by RCERA/B , XCERA/B registers only
							X/RCERC−H are not used

Note: x - Don’t care. However, design needs to validate A-bis mode selection and bit definitions in this table.

5.3 McBSP FIFO Operation

The McBSP powers up without the FIFOs enabled. The FIFO mode of the McBSP can be enabled using the FIFO-enable bits in FIFO register MFFTX. The following steps explain the FIFO features and help programming the McBSP with FIFOs.

1) Reset. At reset the McBSP powers up in standard McBSP mode with the FIFOs features disabled. The FIFO registers MFFTX, MFFRX, and MFFCT will remain inactive.

2) NonFIFO mode. The McBSP without the FIFO registers is enabled. McBSP registers can be programmed to receive and transmit data through DRR2/DRR1 and DXR2/DXR1 registers, respectively. The CPU can directly access these registers to move data from memory to these registers. Interrupt signals will be based on these register pair contents and its related flags.

3) FIFO mode. FIFOs are enabled by setting the MFFENA bit to 1 in the MFFTX register.

4) Active registers. All the McBSP registers and McBSP FIFO registers MFFTX, MFFRX (once FIFOs are enabled), and MFFCT will be active.

5) Interrupts. MRINT/MXINT will generate CPU interrupts for receive and transmit conditions of the FIFO. See the interrupt chapter for details. See the table interrupt section for details on these bit selection and interrupt functionality.

6) Buffers. Transmit and receive data registers of the McBSP will be supplemented with four 16x16 FIFOs. The receive data registers DRR2 and DRR1 will have a 16x16bit FIFO each. Receive channel buffer from DR pin will stack as shown below.

   ```
   DR pin – RSR[1,2] – RBR[1,2] – DRR[1,2]+16x16 FIFO[1,2].
   ```

   Similarly the transmit data register DXR2 and DXR1 will have a 16x16bit FIFO each. Transmit channel buffer up to DX pin will stack as shown below.

   ```
   DX pin – XSR[1,2] –  DXR[1,2]+  16x16 FIFO[1,2]
   ```

7) Transmit FIFO to DXR2/DXR1 data transfer will be based on XEVT/XINT interrupt signal from McBSP and its related flags.

   DRR2/DRR1 to Receive FIFO data register will be based on REVT/RINT interrupt signal from McBSP and its related flags.

8) FIFO status bits. Both transmit and receive FIFOs have status bits TXFFST (bit 12−0) or RXFFST (bits 12−0). They define the number of words available in the FIFOs at any time. The status bits are updated on every 2 word transfer, DXR2/DXR1 and DRR2/DRR1 register, respectively. The transmit FIFO reset bit TXFIFO and receive reset bit RXFIFO will reset the FIFO pointers to zero when these bits are set 1. The FIFOs will resume operation from start once these bits are cleared to zero.

9) Programmable interrupt levels. Both transmit and receive FIFO can generate CPU interrupts. The interrupt trigger is generated whenever the transmit FIFO status bits TXFFST (bits 12−8) match the interrupt trigger level bits TXFFIL (bits 4−0). This provides a programmable interrupt trigger for transmit and receive sections of the McBSP. Default value for these trigger level bits will be 0x11111 for receive FIFO and 0x00000 for transmit FIFO respectively.

5.4 McBSP Receive Interrupt Generation

In the McBSP module, data receive and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA. Since there is no DMA implemented with this module, the DMA interrupt signals are used by the FIFO control logic. The FIFO control logic references these signal to move the data between the FIFO registers and the actual receive registers. The following section maps the interrupt signal selection in FIFO mode and nonFIFO mode for the receive channel.

Figure 5−1. Receive Interrupt Generation

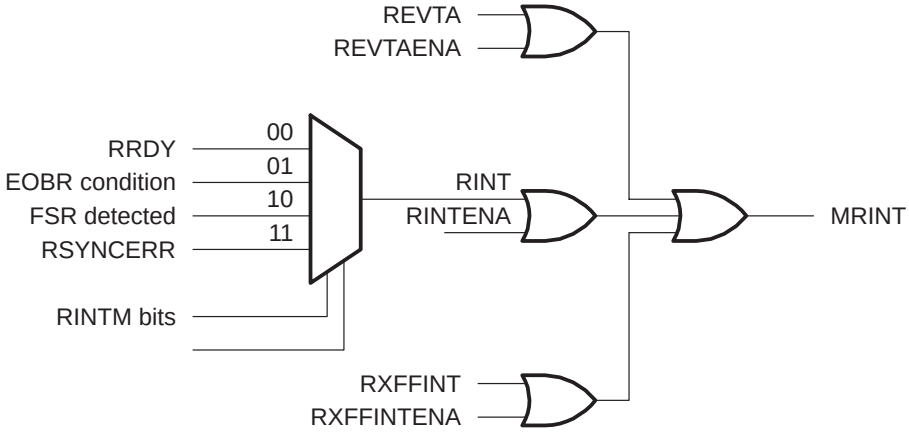


Table 5−3. Receive Interrupt Sources and Signals in NonFIFO Mode and FIFO Mode

Modes	McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR1	Interrupt Enables	Type of Interrupt	Interrupt Line
NonFIFO Mode			**RINTM Bits**			
Receive	RINT	RRDY	00	RINTENA	Every word receive	MRINT
		EOBR	01	RINTENA	Every 16 channel block boundary	
		FSR	10	RINTENA	On every FSR	
		RSYNCERR	11	RINTENA	Frame sync error	
	REVTA		xx	REVTA ENA	A-bis mode update for CPU	
FIFO Mode						
Receive FIFO Interrupt		RXFFINT or RXFFOVF	xx	RXFFINT ENA	Transmit FIFO status interrupt	MRINT
	REVT				Not Used by CPU and FIFO	

Note: x – Don't care

Note:

Since X/RINT, X/REVTA, and X/RXFFINT share the same CPU interrupt, it is recommended that all applications use one of the above selections for interrupt generation. If multiple interrupt enables are selected at the same time, there is a likelihood of interrupts being masked or not recognized.

5.5 McBSP Transmit Interrupt Generation

McBSP module data transmit and error conditions generate two sets of interrupt signals. One set is used for the CPU and the other set is for DMA. Since there is no DMA implemented with this module, the DMA interrupt signals are used by the FIFO control logic. The FIFO control logic will reference these signal to move the data between the FIFO registers and the actual receive registers. The following section maps the interrupt signal selection in FIFO mode and nonFIFO mode for the transmit channel.

Figure 5−2. Transmit Interrupt Generation

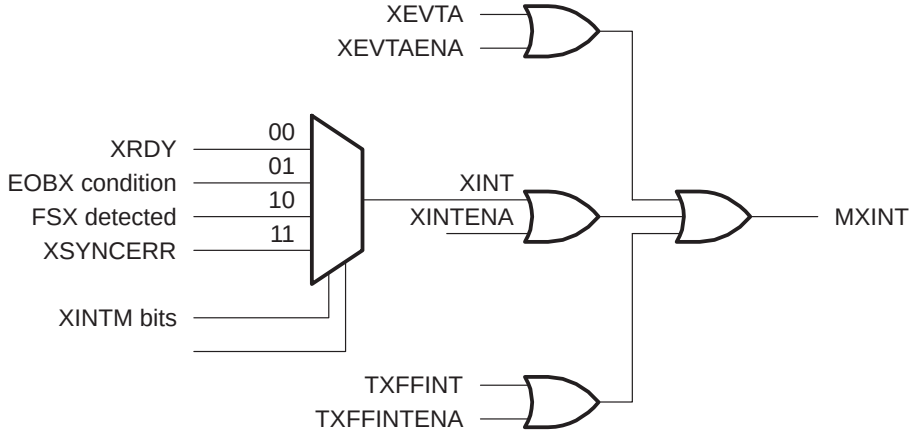


Table 5−4. Transmit Interrupt Sources and Signals in NonFIFO Mode and FIFO Mode

Modes	McBSP Interrupt Signal	Interrupt Flags	Interrupt Enables in SPCR1	Interrupt Enables	Type of Interrupt	Interrupt Line
NonFIFO Mode			**XINTM Bits**			
Transmit	XINT	XRDY	00	XINTENA	Every word receive	MXINT
		EOBX	01	XINTENA	Every 16-channel block boundary	
		FSX	10	XINTENA	On every FSX	
		XSYNCERR	11	XINTENA	Frame sync error	
	XEVTA		XX	XEVTA ENA	A-bis mode update for CPU	
FIFO Mode						
Transmit FIFO Interrupt		TXFFINT	XX	TXFFINT ENA	Transmit FIFO status interrupt	MXINT
	XEVT				Not used by CPU	

Note: x – Don't care.

5.5.1 FIFO Data Register Access Constraints

McBSP registers are allowed only 16-bit access via the peripheral bus. This includes the pair of data registers on receive and transmit channels. DRR2/DRR1is the receive register pair and DXR2/DXR1is the transmit register pair. Based on the word size selected (8/12/16/24/32 bit) on the receive side either the DRR2/DRR1 register pair is used or only the DRR1 is used. Similarly, on the transmit side either the DXR2/DXR1register pair is used or only the DXR1 is used. This is true in nonFIFO mode and FIFO mode of the McBSP module. The following tables explain the read/write order based on the word size. In the FIFO mode, there is only one set of FIFO pointer for both receive and transmit register pair. The FIFO pointer update is valid only if the listed order is maintained. This will guarantee the same data integrity as applicable in the nonFIFO mode of the McBSP.

Table 5−5. Receive FIFO Read Order

	Normal/ Multichannel Mode	Receive FIFO Pointer	Normal Mode	Receive FIFO Pointer
	8/16 Bit Word Read Order		24/32 Bit Word Read Order	
Case1	DRR2	No change	DRR2	No change
	DRR1	Decrement	DRR1	Decrement
Case2	DRR2	No change	DRR2	No change
	DRR2, or n times	no change	DRR2 or n times	no change
	DRR1	Decrement	DRR1	Decrement
Case3	DRR1	Decrement	DRR1	no change
	DRR2	no change	DRR2	no change

Table 5−6. Transmit FIFO Write Order

	Normal/ Multichannel Mode	Receive FIFO Pointer	Normal Mode	Receive FIFO Pointer
	8/16 Bit Word Write Order		24/32 Bit Word Write Order	
Case1	DXR2	No change	DXR2	No change
	DXR1	Increment	DXR1	Increment
Case2	DXR2	No change	DXR2	No change
	DXR2, or n times	No change	DXR2 or n times	No change
	DXR1	Increment	DXR1	Increment
Case3	DXR1	Increment	DXR1	No change
	DXR2	No change	DXR2	No change

Note: DRR2/DRR1 and DXR2/DXR1 registers read and write constraints are native to the McBSP module. Hence maintaining these constraints in FIFO mode will leave no additional conflicts.

5.5.2 FIFO Error Flags

The McBSP has several error flags both on receive and transmit channel. In FIFO mode these bits either made non functional or equivalent flags are provided. Table 5−7 explains the error flags and their new meaning in FIFO mode

Table 5−7. McBSP Error Flags

Error Flags	Function in McBSP, NonFIFO Mode	Function in McBSP, FIFO Mode
RFULL	Indicates DRR2/DRR1 are not read and RXR register is overwritten	This bit will never get set, as the DRR2/DRR1 values are read into the FIFO registers. Use RXFFOVF instead for error condition.
RXFFOVF	Not applicable in nonFIFO mode	FIFO receive overflow flag that will be set whenever the receive FIFO is overflows. The data at the top (or the FIRST IN data) of the FIFO will be lost.
RSYNCERR	Indicates unexpected frame-sync condition, current data reception will abort and restart. Use RINTM bit 11 for interrupt generation on this condition.	Indicates unexpected frame-sync condition, current data reception will abort and restart Use RINTM bits 11 for interrupt generation on this conditions.
XSYNCERR	Indicates unexpected frame-sync condition, current data transmission will abort and restart. Use XINTM bit 11 for interrupt generation on this condition.	Indicates unexpected frame-sync condition, current data transmission will abort and restart. Use RINTM bits 11 for interrupt generation on this condition.

5.5.3 McBSP IDLE Mode

Internal IDLE mode of the McBSP module is not to be implemented. Bit 14 in PCR register for IDLE_EN will be a read/write bit.

5.5.4 McBSP Reset Conditions

The McBSP can be reset either through hardware DSP reset or reset control on receive or transmit channel. The receive channel and transmit channel resets are controlled by RRST (bit 0 in SPCR1) and XRST (bit 0 in SPCR2) bits The McBSP module native reset state of the McBSP pins, register status bits will remain as is in both FIFO nonFIFO modes.

However, the FIFO registers have additional register bits that will control of the reset state of the FIFO pointers and register contents. RXFIFO and TXFIFO reset bits in MFFRX (bit 13) and MFFTX (bit 13) reset the FIFO pointers to zero and power up in reset state.

5.6 McBSP FIFO Register Descriptions

Figure 5−3. McBSP FIFO Transmit Register (MFFTX)

15	14	13	12	11	10	9	8
Reserved	MFFENA	TXFIFO Reset	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
R-0	R/W-0	R/W-1	R-0	R-0	R-0	R-0	R-0
7	6	5	4	3	2	1	0
TXFFINT Flag	TXFFINT Clear	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
R-0	W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read only; W = Write only; R/W = Read/Write; -n = Reset Value

Bit(s)	Name	Description	
15	Reserved	Reserved	
14	MFFENA	0	McBSP FIFO enhancements are disabled and FIFO is in reset.
		1	McBSP FIFO enhancements are enabled.
13	TXFIFO Reset	0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable Transmit FIFO operation
12−8	TXFFST4−0	00000	Transmit FIFO is empty.
		00001	Transmit FIFO has 1 word
		00010	Transmit FIFO has 2 words
		00011	Transmit FIFO has 3 words
		0xxxx	Transmit FIFO has x words
		10000	Transmit FIFO has 16 words
7	TXFFINT	0	TXFIFO interrupt has not occurred. Read only bit
		1	TXFIFO interrupt has occurred. Read only bit
6	TXFFINT CLR	0	Write 0 has no effect on TXFIFINT flag bit, bit reads back a zero
		1	Write 1 to clear TXFFINT flag in bit 7
5	TXFFIENA	0	TX FIFO interrupt based on TXFFIVL match (less than or equal to) will be disabled
		1	TX FIFO interrupt based on TXFFIVL match (less than or equal to) will be enabled.
4−0	TXFFIL4−0	TXFFIL4−0 Transmit FIFO interrupt level bits	
		Transmit FIFO will generate interrupt when the FIFO status bits (TXFFST4−0) and FIFO level bits (TXFFIL4−0) match (less than or equal to).	
		Default value should be 0x00000	

Note: Bit 15 used to be MRST bit and will be reserved in this FIFO implementation

Figure 5−4. McBSP FIFO Receive Register (MFFRX)

15	14	13	12	11	10	9	8
RXFFOVF Flag	RXFFOVF Clear	RXFIFO Reset	RXFFST4	RXFFST3	RXFFST2	RXFFST1	RXFFST0
R-0	W-0	R/W-1	R-0	R-0	R-0	R-0	R-0

7	6	5	4	3	2	1	0
RXFFINT Flag	RXFFINT Clear	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
R-0	W-0	R/W-0	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1

Legend: R = Read only; W = Write only; R/W = Read/Write; -n = Reset Value

Bit(s)	Name	Description	
15	RXFFOVF	0	Receive FIFO has not overflowed, read only bit.
		1	Receive FIFO has overflowed, read only bit. More than 16 words have been received in to the FIFO, and the first received word is lost.
14	RXFFOVFCLR	0	Write 0 has no effect on RXFFOVF flag bit. Bit reads back a zero
		1	Write 1 to clear RXFFOVF flag in bit 15
13	RXFIFO Reset	0	Write 0 to reset the FIFO pointer to zero, and hold in reset.
		1	Re-enable Transmit FIFO operation
12−8	RXFFST4−0	00000	Receive FIFO is empty.
		00001	Receive FIFO has 1 word
		00010	Receive FIFO has 2 words
		00011	Receive FIFO has 3 words
		0xxxx	Receive FIFO has x words
		10000	Receive FIFO has 16 words.
7	RXFFINT	0	RXFIFO interrupt has not occurred. Read only bit
		1	RXFIFO interrupt has occurred. Read only bit
6	RXFFINT CLR	0	Write 0 has no effect on RXFIFINT flag bit, Bit reads back a zero
		1	Write 1 to clear RXFFINT flag in bit 7
5	RXFFIENA	0	RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be disabled
		1	RX FIFO interrupt based on RXFFIVL match (less than or equal to) will be enabled.
4−0	RXFFIL0 −4	RXFFIL4−0 Receive FIFO interrupt level bits	
		Receive FIFO will generate interrupt when the FIFO status bits (RXFFST4−0) and FIFO level bits (RXFFIL4−0) match (greater than or equal to) default value of these bits after reset − 11111. This will avoid frequent interrupts, after reset, as the receive FIFO will be empty most of the time.	

Figure 5−5. McBSP FIFO Control Register (MFFCT)

15	14	8
IACKM	Reserved	
R/W-0	R-0	

7	6	5	4	3	2	1	0
FFTXDLY7	FFTXDLY6	FFTXDLY5	FFTXDLY4	FFTXDLY3	FFTXDLY2	FFTXDLY1	FFTXDLY0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read only; R/W = Read/Write

Bit(s)	Name	Reset	Description
15	IACKM	0	Default value to be written
		1	Reserved function. Do not write 1 to this bit.
14−8		0	Reserved
7−0	FFTXDLY7−0	0x0000	FFTXDLY7−0 are used only in SPI mode. In McBSP mode they are “don’t care”. These bits define the delay between every transfer from FIFO transmit FIFO to transmit (DXR2/DXR1) register. The delay is defined in number CLKX serial clock or baud clock cycles. The 8 bit register could define a minimum delay of 0 serial clock cycles and a maximum of 256 serial clock cycles. In the FIFO mode the transfer from FIFO to (DXR2/DXR1) registers should occur only after the shift register has completed shifting of the last bit. This ensures that the delay count is between two words in the data stream. In McBSP/SPI mode with FIFO and delay enabled, the McBSP registers DXR2/DXR1 are not to be treated as one additional level of buffer.

Figure 5−6. McBSP FIFO Interrupt Register (MFFINT)

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		REVTA ENA	RINT ENA	XEVTA ENA	XINT ENA
R-0		R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read only; R/W = Read/Write

Bit(s)	Name	Reset	Description
15−4	Reserved		Reserved

Figure 5−6.McBSP FIFO Interrupt Register (MFFINT) (Continued)

Bit(s)	Name	Reset	Description
3	REVTA ENA		Enable for A-bis Receive Interrupt on every 16 CLKX/XLKR cycles. Enabled only in A-bis FIFO mode
		0	A-bis Receive interrupt is disabled
		1	A-bis Receive interrupt is enabled
2	RINT ENA		Enable for Receive Interrupt Enabled only in nonFIFO mode
		0	Receive interrupt on XRDY is disabled
		1	Receive interrupt on XRDY is enabled
			This interrupt will be active for any of the conditions selected for RINTM bits
1	XEVTA ENA		Enable for A-bis Transmit Interrupt on every 16 CLKX/XLKR cycles. Enabled only in A-bis FIFO mode
		0	A-bis transmit interrupt is disabled
		1	A-bis transmit interrupt is enabled
0	XINT ENA		Enable for transmit Interrupt Enabled only in nonFIFO mode
		0	Transmit interrupt on XRDY is disabled
		1	Transmit interrupt on XRDY is enabled
			This interrupt will be active for any of the conditions selected for XINTM bits

Figure 5−7. McBSP FIFO Status Register (MFFST)

15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		FSR Flag	EOBR Flag	FSX Flag	EOBX Flag
R-0		R/W-x	R/W-0	R/W-x	R/W-0

Legend: R = Read only; R/W = Read/Write

Bit(s)	Name	Description
15−4	Reserved	Reserved
3	FSR Flag	New frame-sync FSR pulse detection flag . This flag will be set irrespective of whether RINTM bits are 10. Can be used to detect FSR pulse without interrupt. At reset, this bit will be updated based on the FSX/FSR pin status. If left unconnected, they will be set to 1 due to the internal pullups.
		0 FSR Frame sync pulse not detected

Bit(s)	Name	Description
		1 FSR Frame sync pulse detected Write 0 to clear

Figure 5−7.McBSP FIFO Status Register (MFFST) (Continued)

Bit(s)	Name	Description
2	EOBR Flag	End of receive block in multichannel mode . This flag will be set irrespective of whether RINTM bits are 01. Can be used to detect EOB without interrupt. 0 EOBX End of block condition(EOB) not occurred. Write 0 to clear. 1 EOBX End of block condition occurred
1	FSX Flag	New Frame sync FSX pulse detection flag. This flag will be set irrespective of whether XINTM bits are 10. Can be used to detect FSX without interrupt. 0 FSX Frame sync pulse not detected 1 FSX Frame sync pulse detected Write 0 to clear
0	EOBX Flag	End of transmit block in multichannel mode. This flag will be set irrespective of XINTM bits are 01. Can be used detect EOB condition without interrupt. 0 EOBX End of block condition (EOB) not occurred 1 EOBX End of block condition occurred Write 0 to clear

Chapter 6

McBSP Registers

This chapter includes the register and bit descriptions.

Topic **Page**

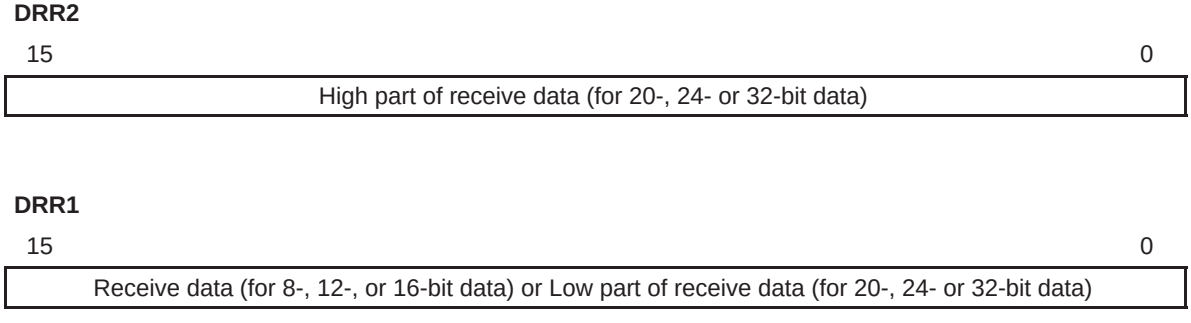
6.1 Data Receive and Transmit Registers

This section includes the Data Receive Registers (DRR2 and DRR1) and the Data Transmit Registers (DXR2 and DXR1).

6.1.1 Data Receive Registers (DRR2 and DRR1)

The CPU or the FIFO controller reads received data from one or both of the data receive registers. If the serial word length is 16 bits or smaller only DRR1 is used. If the serial length is larger than 16 bits, both DRR1 and DRR2 are used, and DRR2 holds the most significant bits. Each frame of receive data in the McBSP can have one phase or two phases, each with its own serial word length.

Figure 6−1. Data Receive Registers (DRR2 and DRR1)



6.1.2 How Data Travels From the Data Receive (DR) Pin to the DRRs

If the serial word length is 16 bits or smaller, receive data on the DR pin is shifted into receive shift register 1 (RSR1) and then copied into receive buffer register 1 (RBR1). The content of RBR1 is then copied to DRR1, which can be read by the CPU or by the FIFO.

If the serial word length is larger than 16 bits, receive data on the DR pin is shifted into both of the receive shift registers (RSR2, RSR1) and then copied into both of the receive buffer registers (RBR2, RBR1). The content of the RBRs is then copied into both of the DRRs, which can be read by the CPU or by the FIFO.

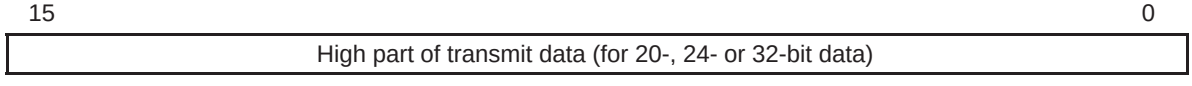
If companding is used during the copy from RBR1 to DRR1 (RCOMPAND = 10b or 11b), the 8-bit compressed data in RBR1 is expanded to a left-justified 16-bit value in DRR1. If companding is disabled, the data copied from RBR[1,2] to DRR[1,2] is justified and bit filled according to the RJUST bits.

6.1.3 Data Transmit Registers (DXR2 and DXR1)

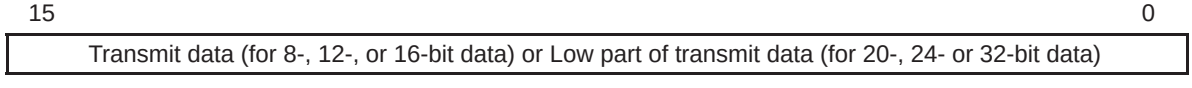
For transmission, the CPU or the FIFO writes data to one or both of the data transmit registers. If the serial word length is 16 bits or smaller, only DXR1 is used. If the word length is larger than 16 bits, both DXR1 and DXR2 are used, and DXR2 holds the most significant bits. Each frame of transmit data in the McBSP can have one phase or two phases, each with its own serial word length.

Figure 6−2. Data Transmit Registers (DXR2 and DXR1)

DXR2



DXR1



6.1.4 How Data Travels From the DXRs to the Data Transmit (DX) Pin

If the serial word length is 16 bits or fewer, data written to DXR1 is copied to transmit shift register 1 (XSR1). From XSR1, the data is shifted onto the DX pin one bit at a time.

If the serial word length is more than 16 bits, data written to DXR1 and DXR2 is copied to both transmit shift registers (XSR2, XSR1). From the XSRs, the data is shifted onto the DX pin one bit at a time.

If companding is used during the transfer from DXR1 to XSR1 (XCOMPAND = 10b or 11b), the McBSP compresses the 16-bit data in DXR1 to 8-bit data in the μ-law or A-law format in XSR1. If companding is disabled, the McBSP passes data from the DXR(s) to the XSR(s) without modification.

The XSRs are not accessible.

6.2 Serial Port Control Registers (SPCR1 and SPCR2)

Each McBSP has two serial port control registers of the form shown in Figure 6−3. and describe the bits in SPCR1 and SPCR2, respectively. These memory-mapped registers enable you to:

- Control various McBSP modes: digital loopback mode (DLB), sign-extension and justification mode for reception (RJUST), clock stop mode (CLKSTP), A-bis mode (ABIS), interrupt modes (RINTM and XINTM), emulation mode (FREE and SOFT)
- Turn on and off the DX-pin delay enabler (DXENA)
- Check the status of receive and transmit operations (RSYNCERR, XSYNCERR, RFULL, $\overline{\text{XEMPTY}}$, RRDY, XRDY)
- Reset portions of the McBSP ($\overline{\text{RRST}}$, $\overline{\text{XRST}}$, $\overline{\text{FRST}}$, $\overline{\text{GRST}}$)

Figure 6−3. Serial Port Control 2 Register (SPCR2)

15					10	9	8
Reserved						FREE	SOFT
R-0						R/W−0	R/W−0
7	6	5	4	3	2	1	0
FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST
R/W-0	R/W−0	R/W−0		R/W−0	R−0	R−0	R/W−0

Note: R = Read access, W = Write access, R/W − Read/write access

Bit(s)	Name	Description
15−10	Reserved	Reserved
9	FREE	Free running mode (FREE and SOFT are taking effect when EMUSUSPEND is set) 0 Free running mode is disabled 1 Free running mode is enabled
8	SOFT	Soft bit (FREE and SOFT is taking effect when EMUSUSPEND is set) 0 SOFT mode is disabled 1 SOFT mode is enabled
7	$\overline{\text{FRST}}$	Frame sync generator reset 0 Frame Synchronization logic is reset. Frame sync signal FSG is not generated by the sample rate generator. 1 Frame sync signal FSG is generated after (FPER+1) number of CLKG clocks i.e. all frame counters are loaded with their programmed values.

Figure 6−3.Serial Port Control 2 Register (SPCR2) (Continued)

Bit(s)	Name	Description
6	$\overline{\text{GRST}}$	Sample rate generator reset
		0 Sample rate generator is reset.
		1 Sample rate generator is pulled out of reset. CLKG is driven as per programmed value in Sample Rate Generator Registers (SRGR).
5−4	XINTM	Transmit Interrupt mode
		00b (X)INT driven by (X)RDY (i.e. end of word) and end of frame in A-bis mode.
		01b (X)INT generated by end-of-block or end-of-frame in multi-channel operation
		10b (X)INT generated by a new frame synchronization
		11b (X)INTM=11b, (X)INT generated by (X)SYNCERR
3	XSYNCERR	Transmit synchronization error
		0 No synchronization error
		1 Synchronization error detected by McBSP.
2	$\overline{\text{XEMPTY}}$	Transmit shift register (XSR) empty
		0 XSR is empty
		1 XSR is not empty XSREMPTY bit
1	XRDY	Transmitter ready
		0 Transmitter is not ready.
		1 Transmitter is ready with data in DXR.
0	$\overline{\text{XRST}}$	Transmitter reset. This resets and enables the transmitter.
		0 The serial port transmitter is disabled and in reset state.
		1 The serial port transmitter is enabled.

Figure 6−4. SPCR1 Register

15	14	13	12	11	10	8
DLB	RJUST		CLKSTP		Reserved	
R/W-0	R/W−0		R/W−0		R−0	

7	6	5	4	3	2	1	0
DXENA	ABIS	RINTM		RSYNCERR	RFULL	RRDY	RRST
R/W-0	R/W−0	R/W−0		R/W−0	R−0	R−0	R/W−0

Notes: R − Read-only access, R/W − Read/write access, −n − n is the value after a DSP reset

Bit(s)	Name	Description
15	DLB	Digital loop back mode
		0 Disabled
		1 Enabled
14−13	RJUST	Receive sign-Extension and justification mode
		0 0 Right-justify and zero-fill MSBs in DRR
		0 1 Right-justify and sign-extend MSBs in DRR
		1 0 Left-justify and zero-fill LSBs in DRR
		1 1 Reserved
12−11	CLKSTP	Clock stop mode. In SPI mode, this field is related to settings on CLKXP and CLKRP in the PCR register.
		0 0 Clock stop mode disabled. Normal clocking for non-SPI mode.
		CLKXP CLKRP
		1 0 0 0 In SPI mode: Clock starts with rising edge without delay
		1 0 1 0 In SPI mode: Clock starts with falling edge without delay
		1 1 0 1 In SPI mode: Clock starts with rising edge with delay
		1 1 1 1 In SPI mode: Clock starts with falling edge with delay
10−8	Reserved	

Figure 6−4.SPCR1 Register (Continued)

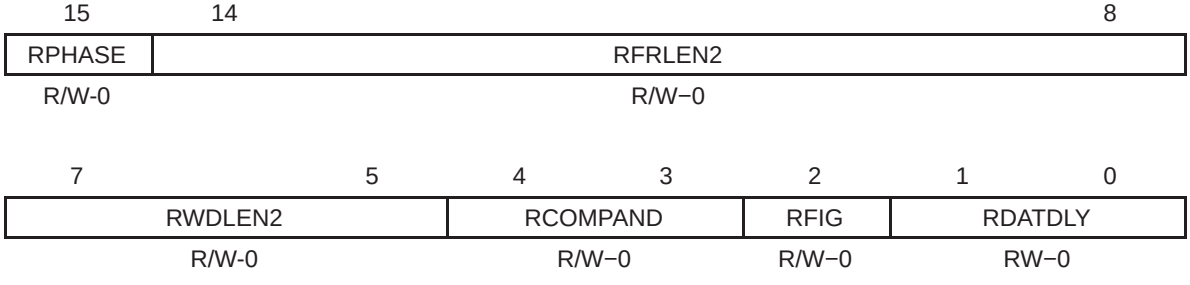
Bit(s)	Name	Description
7	DXENA	DX enabler. Enable extra delay for turn-on time. This bit controls the HI-z enable on the DX pin, not the data itself, so only the first bit will be delayed in the normal mode. In A-bis mode, any bit can be delayed since any bit can go from Hi-Z to valid
		0 DX enabler is off
		1 DX enabler is on
6	ABIS	ABIS mode
		0 A-bis mode is disabled
		1 A-bis mode is enabled
5−4	RINTM	Receive Interrupt mode
		00b (R)INTM driven by (R)RDY (i.e. end of word) and end of frame in A-bis mode.
		01b (R)INTM generated by end-of-block or end-of-frame in multi-channel operation
		10b (R)INTM generated by a new frame synchronization
		11b (R)INTM generated by (R)SYNCERR
3	RSYNCERR	Receive synchronization error
		0 No synchronization error
		1 Synchronization error detected by McBSP.
2	RFULL	Receive shift register (RSR) full
		0 RBR is not in overrun condition
		1 DRR is not read, RBR is full and RSR is also full with new word.
1	RRDY	Receiver ready
		0 Receiver is not ready.
		1 Receiver is ready with data to be read from DRR.
0	$\overline{\text{RRST}}$	Receiver reset. This resets and enables the receiver
		0 The serial port receiver/transmitter is disabled and in reset state.
		1 The serial port receiver/transmitter is enabled.

6.3 Receive Control Registers (RCR1 and RCR2)

Each McBSP has two receive control registers of the form shown in Figure 6−5. These memory-mapped registers enable you to:

- ❑ Specify one or two phases for each frame of receive data (RPHASE)
- ❑ Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (RWDLEN1, RWDLEN2) and the number of words (RFRLEN1, RFRLEN2)
- ❑ Choose a receive companding mode, if any (RCOMPAND)
- ❑ Enable or disable the receive frame-sync ignore function (RFIG)
- ❑ Choose a receive data delay (RDATDLY)

Figure 6−5. Receive Control 2 Register (RCR2)



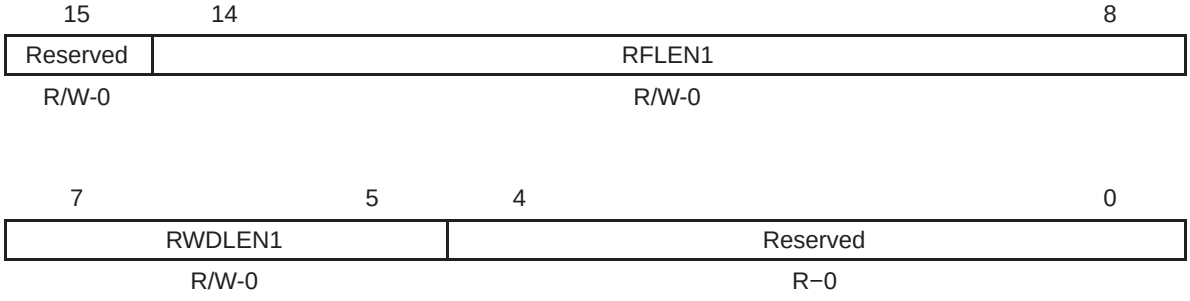
Notes: R − Read-only access, R/W − Read/write access, −n − n is the value after a DSP reset

Bit(s)	Name	Reset	Description	
15	RPHASE	0	Receive phases	
			0	Single phase frame
			1	Dual phase frame
14−8	RFRLEN2	0	Receive frame length 2	
			00b	1 word per frame
			01b	2 words per frame
				|
				|
			11b	128 words per frame

Figure 6−5.Receive Control 2 Register (RCR2) (Continued)

Bit(s)	Name	Description
7−5	RWDLEN2	Receive word length 2
		000b 8 bits
		001b 12 bits
		010b 16 bits
		011b 20 bits
		100b 24 bits
		101b 32 bits
		11Xb Reserved TDM serial port control register (TSPC):TXM bit
4−3	RCOMPAND	Receive companding mode. Modes other than 00b are only enabled when the appropriate (R)WDLEN is 000b, indicating 8-bit data.
		00b No companding, data transfer starts with MSB first.
		01b No companding, 8-bit data, transfer starts with LSB first.
		10b Compand using μ-law for Receive data.
		11b Compand using A-law for Receive data.
2	RFIG	Receive frame ignore
		0 Receive Frame synchronization pulses after the first restarts the transfer.
		1 Receive Frame synchronization pulses after the first are ignored.
1−0	RDATDLY	Receive data delay
		00b 0-bit data delay
		01b 1-bit data delay
		10b 2-bit data delay
		11b Reserved

Figure 6−6. RCR1 Register



Notes: R − Read-only access, R/W − Read/write access, −n − n is the value after a DSP reset

Bit(s)	Name	Description
15	Reserved	Reserved
		0 Valid to write a zero not a 1
		1 Reserved value.
14−8	RFRLEN1	Receive frame length 1
		00b 1 word per frame
		01b 2 words per frame
		|
		|
		11b 128 words per frame
7−5	RWDLEN1	Receive word length 1
		000b 8 bits
		001b 12 bits
		010b 16 bits
		011b 20 bits
		100b 24 bits
		101b 32 bits
		11Xb Reserved
4	Reserved	Reserved
		0 Valid to write a zero not a 1
		1 Reserved value.
3−0	Reserved	Reserved

6.4 Transmit Control Registers (XCR1 and XCR2)

Each McBSP has two transmit control registers of the form shown in Figure 6−7. These I/O-mapped registers enable you to:

- ❑ Specify one or two phases for each frame of transmit data (XPHASE)
- ❑ Define two parameters for phase 1 and (if necessary) phase 2: the serial word length (XWDLEN1, XWDLEN2) and the number of words (XFRLEN1, XFRLEN2)
- ❑ Choose a transmit companding mode, if any (XCOMPAND)
- ❑ Enable or disable the transmit frame-sync ignore function (XFIG)
- ❑ Choose a transmit data delay (XDATDLY)

Figure 6−7. Transmit Control 2 Register (XCR2)

15	14	8
XPHASE	XFRLEN2	
R/W-0	R/W−0	

7	5	4	3	2	1	0
XWDLEN2		XCOMPAND		XFIG	XDATDLY	
R/W-0		R/W−0		R/W−0	R/W−0	

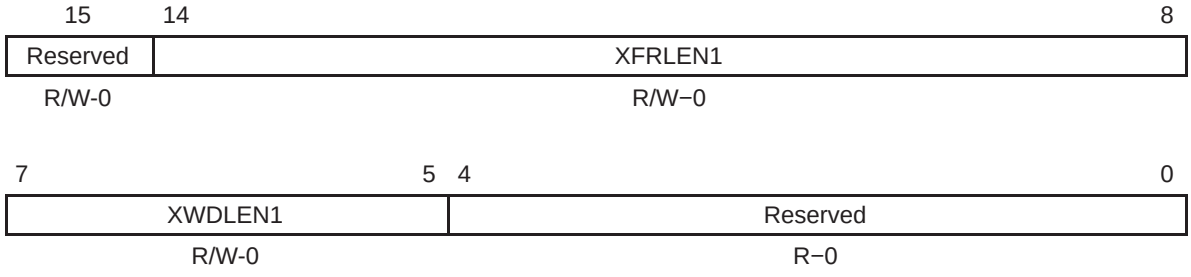
Notes: R − Read-only access, R/W − Read/write access, −X − X is the value after a DSP reset

Bit(s)	Name	Description
15	XPHASE	Transmit phase
		0 Single phase frame
		1 Dual phase frame
14−8	XRFRLEN2	Transmit frame length 2
		00b 1 word per frame
		01b 2 words per frame
		|
		|
		11b 128 words per frame

Figure 6−7.Transmit Control 2 Register (XCR2) (Continued)

Bit(s)	Name	Description
7−5	XWDLEN1	Transmit word length 2
		000b 8 bits
		001b 12 bits
		010b 16 bits
		011b 20 bits
		100b 24 bits
		101b 32 bits
		11Xb Reserved
4−2	XCOMPAND	Transmit companding mode. Modes other than 00b are only enabled when the appropriate XWDLEN is 000b, indicating 8-bit data.
		00b No companding, data transfer starts with MSB first.
		01b No companding, 8-bit data, transfer starts with LSB first.
		10b Compand using μ-law for Transmit data.
		11b Compand using A-law for Transmit data.
1	XFIG	Transmit frame ignore
		0 Transmit Frame synchronization pulses after the first restarts the transfer.
		1 Transmit Frame synchronization pulses after the first are ignored.
0	XDATDLY	Transmit data delay
		00b 0-bit data delay
		01b 1-bit data delay
		10b 2-bit data delay
		11b Reserved

Figure 6−8. XCR1 Register



Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description	
15	Reserved	Reserved	
		0	Valid to write a zero not a 1
		1	Reserved value.
14−8	XFRLEN1	Receive frame length 1	
		00b	1 word per frame
		01b	2 words per frame
			|
			|
		11b	128 words per frame
7−5	XWDLEN1	Transmit word length 1	
		000b	8 bits
		001b	12 bits
		010b	16 bits
		011b	20 bits
		100b	24 bits
		101b	32 bits
		11Xb	Reserved
4	Reserved	Reserved	
		0	Valid to write a zero not a 1
		1	Reserved value.
3−0	Reserved	Reserved	

6.5 Sample Rate Generator Registers (SRGR1 and SRGR2)

Each McBSP has two sample rate generator registers of the form shown in Figure 6−9. The sample rate generator can generate a clock signal (CLKG) and a frame-sync signal (FSG). The I/O-mapped registers SRGR1 and SRGR2 enable you to:

- Select the input clock source for the sample rate generator (CLKSM, in conjunction with the SCLKME bit of PCR)
- Divide down the frequency of CLKG (CLKGDV)
- Select whether internally-generated transmit frame-sync pulse are driven by FSG or by activity in the transmitter (FSGM).
- Specify the width of frame-sync pulses on FSG (FWID) and specify the period between those pulses (FPER)

When an external source (via the CLKR or CLKX pin) provides the input clock source for the sample rate generator:

- If the CLKX/CLKR pin is used, the polarity of the input clock is selected with CLKXP/CLKRP of PCR.
- The GSYNC bit of SRGR2 allows you to make CLKG synchronized to an external frame-sync signal on the FSR pin, so that CLKG is kept in phase with the input clock.

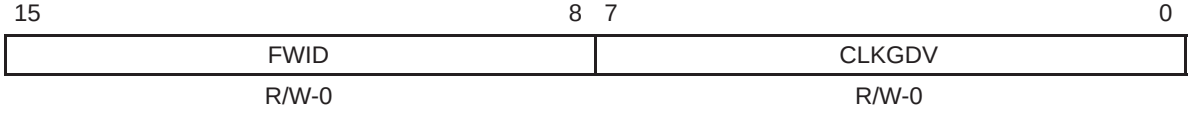
Figure 6−9. Sample Rate Generator 2 Register (SRGR2)

15	14	13	12	11	0
GSYNC	Res	CLKSM	FSGM	FPER	
R/W-0	R/W-1	R/W-1	R/W-0	R/W-0	

Note: R − Read-only access, R/W − Read/write access, −X − X is the value after a DSP reset

Bit(s)	Name	Description
15	GSYNC	Sample rate generator clock synchronization. Only used when the external clock (CLK) drives the sample rate generator clock (CLKSM=0). GSYNC = 0, 0 The sample rate generator clock (CLKG) is free running. 1 The sample rate generator clock (CLKG) is running. But CLKG is re-synchronize and frame sync signal (FSG) is generated only after detecting the receive frame synchronization signal (FSR). Also, frame period, FPER, is a don't care because the period is dictated by the external frame sync pulse.
14	Reserved	Reserved
13	CLKSM	McBSP sample rate generator clock mode. Works with SCLKME bit in PCR register (bit 7) to decide the input clock to sample rate generator module SCLKME: CLKSM 0 0 Reserved 0 1 LSPCLK – Internal clock 1 0 External CLKR pin clock 1 1 External CLKX pin clock
12	FSGM	Sample rate generator Transmit frame synchronization mode. Used when FSXM = 1 in PCR. 0 Transmit frame sync signal (FSX) due to DXR(1/2)-to-XSR(1/2) copy. 1 Transmit frame sync signal driven by the sample rate generator frame sync signal, FSG. TDM serial port control register (TSPC):TXM bit
11−0	FPER	Frame period. This determines when the next frame sync signal should become active. Range: up to 2^{12} ; 1 to 4096 CLKG periods.

Figure 6−10. Sample Rate Generator 1 Register (SRGR1)



Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

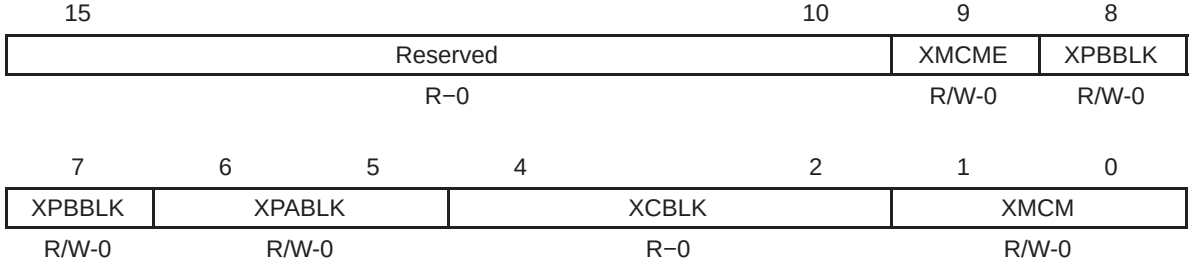
Bit(s)	Name	Description
15−8	FWID	Frame width. Determines the width of the frame sync pulse, FSG, during its active period. Range: up to 2^8 ; 1 to 256 CLKG periods.
7−0	CLKGDV	Sample rate generator clock divider. This value is used as the divide-down number to generate the required sample rate generator clock frequency. Default value is 1.

6.6 Multichannel Control Registers (MCR1 and MCR2)

Each McBSP has two multichannel control registers of the form shown in Figure 6−11. MCR1 has control and status bits (with an R prefix) for multichannel selection operation in the receiver. MCR2 contains the same type of bits (bit with an X prefix) for the transmitter. These I/O-mapped registers enable you to:

- ❑ Enable all channels or only selected channels for reception (RMCM)
- ❑ Choose which channels are enabled/disabled and masked/unmasked for transmission (XMCM)
- ❑ Specify whether two partitions (32 channels at a time) or eight partitions (128 channels at a time) can be used (RMCME for reception, XMCME for transmission)
- ❑ Assign blocks of 16 channels to partitions A and B when the 2-partition mode is selected (RPABLK and RPBBLK for reception, XPABLK and XPBBLK for transmission)
- ❑ Determine which block of 16 channels is currently involved in a data transfer (RCBLK for reception, XCBLK for transmission)

Figure 6−11. Multichannel Control 2 Register (MCR2)



Note: R = Read access, W = Write access, C = Clear only, −n = value after reset

Bit(s)	Name	Description
15−10	Reserved	Reserved
9	XMCME	Enhanced transmit multichannel selection enable (XMCME) XMCME operates in conjunction with RMCME. The RMCME and XMCME bit values need to be the same. ❑ RMCME = 0 and XMCME = 0: (normal multichannel selection mode) (Default value) Maximum 32 channels can be enabled at one time. ❑ RMCME = 1 and XMCME = 1: (128 channel selection mode) Maximum 128 channels can be enabled at one time. All other modes reserved.

Figure 6−11. Multichannel Control 2 Register (MCR2) (Continued)

Bit(s)	Name	Description
8−7	XPBBLK	Receive/transmit partition B block
		00b Block 1. Channel 16 to channel 31
		01b Block 3. Channel 48 to channel 63
		10b Block 5. Channel 80 to channel 95
		11b Block 7. Channel 112 to channel 127
6−5	XPABLK	Receive/transmit partition A block
		00b Block 0. Channel 0 to channel 15
		01b Block 2. Channel 32 to channel 47
		10b Block 4. Channel 64 to channel 79
		11b Block 6. Channel 96 to channel 111
4−2	XCBLK	Receive/transmit current block
		000b Block 0. Channel 0 to channel 15
		001b Block 1. Channel 16 to channel 31
		010b Block 2. Channel 32 to channel 47
		011b Block 3. Channel 48 to channel 63
		100b Block 4. Channel 64 to channel 79
		101b Block 5. Channel 80 to channel 95
		110b Block 6. Channel 96 to channel 111
		111b Block 7. Channel 112 to channel 127
1−0	XMCM	Transmit multichannel selection enable
		00b All channels enabled without masking (DX is always driven during transmission of data).
		01b All channels disabled and therefore masked by default. Required channels are selected by enabling XP(A/B)BLK and XCER(A/B) appropriately. Also, these selected channels are not masked and therefore DX is always driven.
		10b All channels enabled, but masked. Selected channels enabled via XP(A/B)BLK and XCER(A/B) are unmasked.
		11b All channels disabled and therefore masked by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately. Selected channels can be unmasked by RP(A/B)BLK and XCER(A/B). This mode is used for symmetric transmit and receive operation. BSP serial port control extension register (SPCE):HALTR bit

Figure 6−12. Multichannel Control 1 Register (MCR1)

15	10	9	8
Reserved		RMCME	RPBBLK
R−0		R/W-0	R/W-0

7	6	5	4	2	1	0
RPBBLK	RPABLK		RCBLK		Reserved	RMCM
R/W-0	R/W-0		R−0		R-0	R/W-0

Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description
15−10	Reserved	Reserved
9	RMCME	Enhanced receive multichannel selection enable (RMCME) RMCME operates in conjunction with XMCME. The RMCME and XMCME bit values need to be the same. ❑ RMCME = 0 and XMCME = 0: (normal multichannel selection mode) (Default value) Maximum 32 channels can be enabled at one time. ❑ RMCME = 1 and XMCME = 1: (128 channel selection mode) Maximum 128 channels can be enabled at one time. All other modes reserved.
8−7	RPBBLK	Receive/transmit partition b block 00b Block 1. Channel 16 to channel 31 01b Block 3. Channel 48 to channel 63 10b Block 5. Channel 80 to channel 95 11b Block 7. Channel 112 to channel 127
6−5	RPABLK	Receive/transmit partition a block 00b Block 0. Channel 0 to channel 15 01b Block 2. Channel 32 to channel 47 10b Block 4. Channel 64 to channel 79 11b Block 6. Channel 96 to channel 111

Figure 6−12. Multichannel Control 1 (MCR1) Register (Continued)

Bit(s)	Name	Description
4−2	RCBLK	Receive/transmit current block 000b Block 0. Channel 0 to channel 15 001b Block 1. Channel 16 to channel 31 010b Block 2. Channel 32 to channel 47 011b Block 3. Channel 48 to channel 63 100b Block 4. Channel 64 to channel 79 101b Block 5. Channel 80 to channel 95 110b Block 6. Channel 96 to channel 111 111b Block 7. Channel 112 to channel 127
1	Reserved	Reserved
0	RMCM	Receive multichannel selection enable 0 All 128 channels enabled. 1 All channels disabled by default. Required channels are selected by enabling RP(A/B)BLK and RCER(A/B) appropriately.BSP serial port control extension register (SPCE):HALTR bit

6.7 Pin Control Register (PCR)

Each McBSP has one pin control register of the form shown in Figure 6−13. This I/O-mapped register enables you to:

- ❑ Choose a frame-sync mode for the transmitter (FSXM) and for the receiver (FSRM)
- ❑ Choose a clock mode for transmitter (CLKXM) and for the receiver (CLKRM)
- ❑ Select the input clock source for the sample rate generator (SCLKME, in conjunction with the CLKSM bit of SRGR2)
- ❑ Read or write data when the CLKS, DX, and DR pins are configured as general-purpose I/O pins (CLKS_STAT, DX_STAT, and DX_STAT)
- ❑ Choose whether frame-sync signals are active low or active high (FSXP for transmission, FSRP for reception)
- ❑ Specify whether data is sampled on the falling edge or the rising edge of the clock signals (CLKXP for transmission, CLKRP for reception)

Figure 6−13. Pin Control Register (PCR)

15	12	11	10	9	8
Reserved		FSXM	FSRM	CLKXM	CLKRM
R−0		R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
SCLKME	CLKS_STAT	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP
R/W-0	R−0	R/W-0	R−0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description
15−12	Reserved	Reserved
11	FSXM	Transmit frame synchronization mode
		0 Frame synchronization pulses generated by an external device. FSR is an input pin
		1 Frame synchronization generated internally by sample rate generator. FSR is an output pin except when GSYNC = 1 in SRGR.

Figure 6−13. Pin Control Register (PCR) (Continued)

Bit(s)	Name	Description
10	FSRM	Receive frame synchronization mode 0 Frame synchronization signal derived from an external source 1 Frame synchronization is determined by the Sample Rate Generator frame synchronization mode bit FSGM in the SRGR2.
9	CLKXM	Transmitter clock mode 0 Receiver/Transmitter clock is driven by an external clock with CLK(R/X) as an input pin. 1 CLK(R/X) is an output pin and is driven by the internal sample rate generator. During SPI mode (CLKSTP is a non-zero value): 0 McBSP is a slave and clocks (CLKX) is driven by the SPI master in the system. CLKR is internally driven by CLKX. 1 McBSP is a master and generates the clock (CLKX) to drive its receive clock (CLKR) and the shift clock of the SPI-compliant slaves in the system.
8	CLKRM	Receiver clock mode Case 1: Digital Loop Back Mode not set (DLB = 0) in SPCR1 0 Receive clock (CLKR) is an input driven by an external clock. 1 CLKR is an output pin and is driven by the internal sample rate generator. Case 2: Digital loop back mode set (DLB = 1) in SPCR1 0 Receive clock (not the CLKR pin) is driven by transmit clock (CLKX) which is based on CLKXM bit in PCR. CLKR pin is in high impedance. 1 CLKR is an output pin and is driven by the transmit clock. The transmit clock is derived based on CLKXM bit in the PCR.

Figure 6−13. Pin Control Register (PCR) (Continued)

Bit(s)	Name	Description
7	SCLKME	Enhanced sample clock mode selection bit. McBSP allow either the receive clock pin (CLKR) or the transmit clock pin (CLKX) to be configured as the input clock to the sample rate generator. This enhancement is enabled through two register bits: pin control register (PCR) bit 7 – (SCLKME), and sample rate generator register 2 (SRGR2) bit 13 – McBSP sample rate generator clock mode (CLKSM) SCLKME CLKSM 0 0 Reserved 0 1 LSPCLK − Internal clock 1 0 External CLKR pin clock 1 1 External CLKX pin clock
6	CLKS_ STAT	Reserved in this implementation. Read 0
5	DX_STAT	DX pin status. Reflects value driven on to DX pin when selected as a general purpose output.
4	DR_STAT	DR pin status. Reflects value on DR pin when selected as a general purpose input.
3	FSXP	Transmit frame synchronization polarity 0 Frame synchronization pulse FSXP is active high 1 Frame synchronization pulse FSXP is active low.
2	FSRP	Receive frame synchronization polarity 0 Frame synchronization pulse FSRP is active high 1 Frame synchronization pulse FSRP is active low
1	CLKXP	Transmit clock polarity 0 Transmit data sampled on rising edge of CLKX 1 Transmit data sampled on falling edge of CLKX
0	CLKRP	Receive clock polarity 0 Receive data sampled on falling edge of CLKR 1 Receive data sampled on rising edge of CLKR

6.8 Receive Channel Enable Registers (RCERA − RCERH)

Each McBSP has eight receive channel enable registers of the form shown in Figure 6−14. There is one for each of the receive partitions: A, B, C, D, E, F, G, and H.

These memory-mapped registers are only used when the receiver is configured to allow individual enabling and disabling of the channels (RMCM = 1) or when the receiver needs bit-enable patterns for the A-bis mode (ABIS = 1). For more details about the way these registers are used be sure to read the topics at the end of this section:

- ❑ *RCERs Used in the Receive Multichannel Selection Mode* (page 6-26)
- ❑ *RCERs Used in the A-bis Mode* (page 6-27)

Figure 6−14. Receive Channel Enable Register (RCERA/B)

15	14	13	12	11	10	9	8
RCEA15	RCEA14	RCEA13	RCEA12	RCEA11	RCEA10	RCEA9	RCEA8
R/W-0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

7	6	5	4	3	2	1	0
RCEA7	RCEA6	RCEA5	RCEA4	RCEA3	RCEA2	RCEA1	RCEA0
R/W-0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description
	RCERA	**Receive/Transmit Enable Register A**
15−0	RCEA*n* 0 ≤ n ≤ 15	Receive/transmit channel enable
		0 Disables reception/transmission of *n*th channel in an even-numbered block in partition A
		1 Enables reception/transmission of *n*th channel in an even-numbered block in partition A
	RCERB	**Receive/Transmit enable register B**
15−0	RCER*n* 0 ≤ n ≤ 15	Receive/transmit channel enable
		0 Disables reception/transmission of *n*th channel in odd-numbered block in partition B.
		1 Enables reception/transmission of *n*th channel in odd-numbered block in partition B.

Figure 6−15. RCER(A−G)−Receive Channel Enable Registers − A, C, E, G

15	14	13	12	11	10	9	8
RCERx15	RCERx14	RCERx13	RCERx12	RCERx11	RCERx10	RCERx9	RCERx8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RCERx7	RCERx6	RCERx5	RCERx4	RCERx3	RCERx2	RCERx1	RCERx0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description
15−0	RCERx	Receive channel enable register
		0 Disables reception of *n*th channel in partition x.
		1 Enables reception of *n*th channel in partition x.

Note: Receive channel enable register bit layout for 128 channels
x= Partition A, C, E, G; n = bit 15−0
y = Partition B, D, F, H; n = bit 15−0

Figure 6−16. RCER(B−H)−Receive Channel Enable Registers − B,D,F,H

15	14	13	12	11	10	9	8
RCERy15	RCERy14	RCERy13	RCERy12	RCERy11	RCERy10	RCERy9	RCERy8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0
RCERy7	RCERy6	RCERy5	RCERy4	RCERy3	RCERy2	RCERy1	RCERy0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description
15−0	RCERy	Receive channel enable register
		0 Disables reception of *n*th channel in partition y.
		1 Enables reception of *n*th channel in partition y.

Note: Receive channel enable register bit layout for 128 channels
x= Partition A, C, E, G; n = bit 15−0
y = Partition B, D, F, H; n = bit 15−0

6.8.1 RCERs Used in the Receive Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the RCERs depends on whether 32 or 128 channels are individually selectable, as defined by the RMCME bit (see Table 6−1).

Table 6−1. Use of the Receive Channel Enable Registers in theReceive Multichannel Selection Mode

Selectable Channels	Block Assignments	Channel Assignments
32 (RMCME = 0)	RCERA: Channels n–(n + 15) (channels assigned with the RPABLK bits)	RCEA0: Channel n RCEA1: Channel (n + 1) RCEA2: Channel (n + 2) : RCEA15: Channel (n + 15)
	RCERB: Channels m–(m + 15) (channels assigned with the RPBBLK bits)	RCEB0: Channel m RCEB1: Channel (m + 1) RCEB2: Channel (m + 2) : RCEB15: Channel (m + 15)
	Other RCERs not used	–
128 (RMCME = 1)	RCERA: Block 0	RCEA0: Channel 0 RCEA1: Channel 1 RCEA2: Channel 2 : RCEA15: Channel 15
	RCERB: Block 1	RCEB0: Channel 16 RCEB1: Channel 17 RCEB2: Channel 18 : RCEB15: Channel 31
	RCERC: Block 2	RCEC0: Channel 32 RCEC1: Channel 33 RCEC2: Channel 34 : RCEC15: Channel 47
	RCERD: Block 3	RCED0: Channel 48 RCED1: Channel 49 RCED2: Channel 50 : RCED15: Channel 63

Table 6−1. Use of the Receive Channel Enable Registers in theReceive Multichannel Selection Mode (Continued)

Selectable Channels	Block Assignments	Channel Assignments
128 (continued)	RCERE: Block 4	RCEE0: Channel 64 RCEE1: Channel 65 RCEE2: Channel 66 : RCEE15: Channel 79
	RCERF: Block 5	RCEF0: Channel 80 RCEF1: Channel 81 RCEF2: Channel 82 : RCEF15: Channel 95
	RCERG: Block 6	RCEG0: Channel 96 RCEG1: Channel 97 RCEG2: Channel 98 : RCEG15: Channel 111
	RCERH: Block 7	RCEH0: Channel 112 RCEH1: Channel 113 RCEH2: Channel 114 : RCEH15: Channel 127

6.8.2 RCERs Used in the A-bis Mode

In A-bis mode operation, only RCERA and RCERB are used. Sixteen bits at a time are passed to the receiver. As each of the 16 bits arrives on the DR pin (the MSB, bit 15, arrives first), the bit is stored or ignored, depending on the bit-enable pattern in RCERA or RCERB. The first 16 bits that arrive are handled according to the bit-enable pattern in RCERA. For example, if RCEA6 = 1 and all the other bits of RCERA are 0s, only bit 6 is stored. Each of the next 16 bits is stored or ignored according to the bit-enable pattern in RCERB. The receiver alternately uses RCERA and RCERB for consecutive 16-bit words.

Table 6−2 shows how bits of RCERA correspond to the bits of the incoming word.

Table 6−2. Use of Receive Channel Enable Registers A and B in the A-bis Mode

Register	Bits
RCERA	RCEA15: Enables or masks the first bit of the incoming word RCEA14: Enables or masks bit 14 of the incoming word RCEA13: Enables or masks bit 13 of the incoming word : RCEA0: Enables or masks the last bit of the incoming word
RCERB	RCEB15: Enables or masks the first bit of the incoming word RCEB14: Enables or masks bit 14 of the incoming word RCEB13: Enables or masks bit 13 of the incoming word : RCEB0: Enables or masks the last bit of the incoming word

6.9 Transmit Channel Enable Registers (XERA − XCERH)

Each McBSP has eight transmit channel enable registers of the form shown in the following figure. There is one for each of the transmit partitions: A, B, C, D, E, F, G, and H. provides a general bit description that applies to each of the transmit channel enable registers.

These I/O-mapped registers are only used when transmitter is configured to allow individual disabling/enabling and masking/unmasking of the channels (XMCM is nonzero) or when the transmitter needs bit-enable patterns for the A-bis mode (ABIS = 1). For more details about the way these registers are used, read the topics at the end of this section:

- *XCERs Used in a Transmit Multichannel Selection Mode* (page 6-30)
- *XCERs Used in the A-bis Mode* (page 6-32)

Figure 6−17. Transmit Channel Enable Registers A. C. E. G (XCERA−XCERG)

15	14	13	12	11	10	9	8
XCERx15	XCERx14	XCERx13	XCERx12	XCERx11	XCERx10	XCERx9	XCERx8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XCERx7	XCERx6	XCERx5	XCERx4	XCERx3	XCERx2	XCERx1	XCERx0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description
15−0	XCERx	Receive channel enable register
		0 Disables transmit of *n*th channel in partition x.
		1 Enables transmit of *n*th channel in partition x.

Note: Transmit channel enable register bit layout for 128 channels
x= Partition A, C, E, G; n = bit 15−0
y = Partition B, D, F, H; n = bit 15−0

Figure 6−18. Transmit Channel Enable Registers–B, D, F, H (XCERB−XCERH)

15	14	13	12	11	10	9	8
XCERy15	XCERy14	XCERy13	XCERy12	XCERy11	XCERy10	XCERy9	XCERy8
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
XCERy7	XCERy6	XCERy5	XCERy4	XCERy3	XCERy2	XCERy1	XCERy0
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, C = Clear only, −n = value after reset;

Bit(s)	Name	Description
15−0	XCERy	Receive channel enable register
		0 Disables transmit of *n*th channel in partition y.
		1 Enables transmit of *n*th channel in partition y.

Note: Transmit channel enable register bit layout for 128 channels
x= Partition A, C, E, G; n = bit 15−0
y = Partition B, D, F, H; n = bit 15−0

6.9.1 XCERs Used in a Transmit Multichannel Selection Mode

For multichannel selection operation, the assignment of channels to the XCERs, depends on whether 32 or 128 channels are individually selectable, as defined by the XMCME bit, as shown in the following table.

Note:

When XMCM = 11b (for symmetric transmission and reception), the transmitter uses the receive channel enable registers (RCERs) to enable channels and uses the XCERs to unmask channels for transmission.

Table 6−3. Use of the Transmit Channel Enable Registers in a Transmit Multichannel Selection Mode

Selectable Channels	Block Assignments	Channel Assignments
32 (XMCME = 0)	XCERA: Channels n–(n + 15) (channels assigned with the XPABLK bits for XMCM = 01b or 10b; assigned with the RPABLK bits for XMCM = 11b)	XCEA0: Channel n XCEA1: Channel (n + 1) XCEA2: Channel (n + 2) : XCEA15: Channel (n + 15)
	XCERB: Channels m–(m + 15) (channels assigned with the XPBBLK bits for XMCM = 01b or 10b; assigned with the RPBBLK bits for XMCM = 11b)	XCEB0: Channel m XCEB1: Channel (m + 1) XCEB2: Channel (m + 2) : XCEB15: Channel (m + 15)
	Other XCERs not used	–
128 (XMCME = 1)	XCERA: Block 0	XCEA0: Channel 0 XCEA1: Channel 1 XCEA2: Channel 2 : XCEA15: Channel 15
	XCERB: Block 1	XCEB0: Channel 16 XCEB1: Channel 17 XCEB2: Channel 18 : XCEB15: Channel 31
	XCERC: Block 2	XCEC0: Channel 32 XCEC1: Channel 33 XCEC2: Channel 34 : XCEC15: Channel 47
128 (continued)	XCERD: Block 3	XCED0: Channel 48 XCED1: Channel 49 XCED2: Channel 50 : XCED15: Channel 63
	XCERE: Block 4	XCEE0: Channel 64 XCEE1: Channel 65 XCEE2: Channel 66 : XCEE15: Channel 79

Table 6−3. Use of the Transmit Channel Enable Registers in a Transmit Multichannel Selection Mode (Continued)

Selectable Channels	Block Assignments	Channel Assignments
	XCERF: Block 5	XCEF0: Channel 80 XCEF1: Channel 81 XCEF2: Channel 82 : XCEF15: Channel 95
	XCERG: Block 6	XCEG0: Channel 96 XCEG1: Channel 97 XCEG2: Channel 98 : XCEG15: Channel 111
	XCERH: Block 7	XCEH0: Channel 112 XCEH1: Channel 113 XCEH2: Channel 114 : XCEH15: Channel 127

6.9.2 XCERs Used in the A-bis Mode

In A-bis mode operation, only XCERA and XCERB are used. Sixteen bits at a time are passed to data transmit register 1 (DXR1) by the CPU or FIFO. Each of the 16 bits is transmitted on the DX pin or is ignored, depending on the bit-enable pattern in XCERA or XCERB. The first 16 bits that enter the transmitter are handled according to the bit-enable pattern in XCERA. For example, if XCEA13 = 1 and all the other bits of XCERA are 0s, only bit 13 is transmitted. Each of the next 16 bits is transmitted or ignored according to the bit-enable pattern in XCERB. The transmitter alternately uses XCERA and XCERB for consecutive 16-bit words.

Table 6−4 shows how bits of XCERA and XCERB correspond to the bits of a word written to DXR1 (MSB = most significant bit, LSB = least significant bit).

Table 6−4. Use of Transmit Channel Enable Registers A and B in the A-bis Mode

Register	Bits
XCERA	XCEA15: Enables or masks the MSB of the word written to DXR1 XCEA14: Enables or masks bit 14 of the word written to DXR1 XCEA13: Enables or masks bit 13 of the word written to DXR1 : XCEA0: Enables or masks the LSB of the word written to DXR1
XCERB	XCEB15: Enables or masks the MSB of the word written to DXR1 XCEB14: Enables or masks bit 14 of the word written to DXR1 XCEB13: Enables or masks bit 13 of the word written to DXR1 : XCEB0: Enables or masks the LSB of the word written to DXR1

6.10 Register Bit Summary

Table 6−5. Register Bit Summary (Base Address 0x00 7800)

Register Address Offset 0x00	MSB bits / LSB bits	15 / 7	14 / 6	13 / 5	12 / 4	11 / 3	10 / 2	9 / 1	8 / 0
7800	DRR2	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
		Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
7801	DRR1	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
7802	DXR2	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
		Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
7803	DXR1	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
7804	SPCR2	Reserved						FREE	SOFT
		R−0						R/W−0	R/W−0
		FRST	GRST	XINTM		XSYNCERR	XEMPTY	XRDY	XRST
		R/W−0	R/W−0	R/W−0		R/W−0	R−0	R−0	R/W−0
7805	SPCR1	DLB	RJUST		CLKSTP		Reserved		
		R/W−0	R/W−0		R/W−0		R−0		
		DXENA	ABIS	RINTM		RSYNCERR	RFULL	RRDY	RRST
		R/W−0	R/W−0	R/W−0		R/W−0	R−0	R−0	R/W−0
7806	RCR2	RPHASE	RFRLEN2						
		R/W−0	R/W−0						
		RWDLEN2			RCOMPAND		RFIG	RDATDLY	
		R/W−0			R/W−0		R/W−0	R/W−0	
7807	RCR1	Reserved	RFRLEN1						
		R/W−0	R/W−0						
		RWDLEN1			Reserved				
		R/W−0			R−0				
7808	XCR2	XPHASE	XFRLEN2						
		R/W−0	R/W−0						

Table 6−5. Register Bit Summary (Continued)(Base Address 0x00 7800) (Continued)

Register Address Offset 0x00	MSB bits	15	14	13	12	11	10	9	8
	LSB bits	7	6	5	4	3	2	1	0
		XWDLEN2			XCOMPAND		XFIG	XDATDLY	
		R/W−0			R/W−0		R/W−0	R/W−0	
7809	XCR1	Reserved	XFRLEN1						
		R/W−0	R/W−0						
		XWDLEN1			Reserved				
		R/W−0			R−0				
780A	SRGR2	GSYNC	Reserved	CLKSM	FSGM	FPER			
		R/W−0	R/W−0	R/W−1	R/W−0	R/W−0			
		FPER							
		R/W−0							
780B	SRGR1	FWID							
		R/W−0							
		CLKGDV							
		R/W−0							
780C	MCR2	Reserved						XMCME	XPBBLK
		R−0						R/W−0	R/W−0
		XPBBLK	XPABLK		XCBLK			XMCM	
		R/W−0	R/W−0		R−0			R/W−0	
780D	MCR1	Reserved						RMCME	RPBBLK
		R−0						R/W−0	R/W−0
		RPBBLK	RPABLK		RCBLK			Reserved	RMCM
		R/W−0	R/W−0		R−0			R−0	R/W−0
780E	RCERA	RCEA15	RCEA14	RCEA13	RCEA12	RCEA11	RCEA10	RCEA9	RCEA8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCEA7	RCEA6	RCEA5	RCEA4	RCEA3	RCEA2	RCEA1	RCEA0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
780F	RCERB	RCEB15	RCEB14	RCEB13	RCEB12	RCEB11	RCEB10	RCEB9	RCEB8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCEB7	RCEB6	RCEB5	RCEB4	RCEB3	RCEB2	RCEB1	RCEB0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7810	XCERA	XCEA15	XCEA14	XCEA13	XCEA12	XCEA11	XCEA10	XCEA9	XCEA8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		XCEA7	XCEA6	XCEA5	XCEA4	XCEA3	XCEA2	XCEA1	XCEA0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7811	XCERB	XCEB15	XCEB14	XCEB13	XCEB12	XCEB11	XCEB10	XCEB9	XCEB8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Table 6−5. Register Bit Summary (Continued)(Base Address 0x00 7800) (Continued)

Register Address Offset 0x00	MSB bits	15	14	13	12	11	10	9	8
	LSB bits	7	6	5	4	3	2	1	0
		XCEB7	XCEB6	XCEB5	XCEB4	XCEB3	XCEB2	XCEB1	XCEB0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7812	PCR	Reserved				FSXM	FSRM	CLKXM	CLKRM
		R−0				R/W−0	R/W−0	R/W−0	R/W−0
		SCLKME	Reserved	DX_STAT	DR_STAT	FSXP	FSRP	CLKXP	CLKRP
		R/W−0	R−0	R/W−0	R−0	R/W−0	R/W−0	R/W−0	R/W−0
7813	RCERC	RCEC15	RCEC14	RCEC13	RCEC12	RCEC11	RCEC10	RCEC9	RCEC8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCEC7	RCEC6	RCEC5	RCEC4	RCEC3	RCEC2	RCEC1	RCEC0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7814	RCERD	RCED15	RCED14	RCED13	RCED12	RCED11	RCED10	RCED9	RCED8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCED7	RCED6	RCED5	RCED4	RCED3	RCED2	RCED1	RCED0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7815	XCERC	XCEC15	XCEC14	XCEC13	XCEC12	XCEC11	XCEC10	XCEC9	XCEC8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		XCEC7	XCEC6	XCEC5	XCEC4	XCEC3	XCEC2	XCEC1	XCEC0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7816	XCERD	XCED15	XCED14	XCED13	XCED12	XCED11	XCED10	XCED9	XCED8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		XCED7	XCED6	XCED5	XCED4	XCED3	XCED2	XCED1	XCED0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7817	RCERE	RCEB15	RCEB14	RCEB13	RCEB12	RCEB11	RCEB10	RCEB9	RCEB8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCEB7	RCEB6	RCEB5	RCEB4	RCEB3	RCEB2	RCEB1	RCEB0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7818	RCERF	RCEF15	RCEF14	RCEF13	RCEF12	RCEF11	RCEF10	RCEF9	RCEF8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCEF7	RCEF6	RCEF5	RCEF4	RCEF3	RCEF2	RCEF1	RCEF0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
7819	XCERE	XCEB15	XCEB14	XCEB13	XCEB12	XCEB11	XCEB10	XCEB9	XCEB8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		XCEB7	XCEB6	XCEB5	XCEB4	XCEB3	XCEB2	XCEB1	XCEB0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
781A	XCERF	XCEF15	XCEF14	XCEF13	XCEF12	XCEF11	XCEF10	XCEF9	XCEF8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

Table 6−5. Register Bit Summary (Continued)(Base Address 0x00 7800) (Continued)

Register Address Offset 0x00	MSB bits	15	14	13	12	11	10	9	8
	LSB bits	7	6	5	4	3	2	1	0
		XCEF7	XCEF6	XCEF5	XCEF4	XCEF3	XCEF2	XCEF1	XCEF0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
781B	RCERG	RCEG15	RCEG14	RCEG13	RCEG12	RCEG11	RCEG10	RCEG9	RCEG8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCEG7	RCEG6	RCEG5	RCEG4	RCEG3	RCEG2	RCEG1	RCEG0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
781C	RCERH	RCEH15	RCEH14	RCEH13	RCEH12	RCEH11	RCEH10	RCEH9	RCEH8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		RCEH7	RCEH6	RCEH5	RCEH4	RCEH3	RCEH2	RCEH1	RCEH0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
781D	XCERG	XCEG15	XCEG14	XCEG13	XCEG12	XCEG11	XCEG10	XCEG9	XCEG8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		XCEG7	XCEG6	XCEG5	XCEG4	XCEG3	XCEG2	XCEG1	XCEG0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
781E	XCERH	XCEH15	XCEH14	XCEH13	XCEH12	XCEH11	XCEH10	XCEH9	XCEH8
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
		XCEH7	XCEH6	XCEH5	XCEH4	XCEH3	XCEH2	XCEH1	XCEH0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0

The registers bits are summarized in Table 6−6.

Table 6−6. FIFO Register Bit Descriptions (Base address 0x00 7800)

Register Address-Offset 0x000	MSB bits	15	14	13	12	11	10	9	8
	LSB bits	7	6	5	4	3	2	1	0
00	DRR2	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
		Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
01	DRR1	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		R−0	R−0	R−0	R−0	R−0	R−0	R−0	R−0
02	DXR2	Bit31	Bit30	Bit29	Bit28	Bit27	Bit26	Bit25	Bit24
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
		Bit23	Bit22	Bit21	Bit20	Bit19	Bit18	Bit17	Bit16

Table 6−6. FIFO Register Bit Descriptions (Base address 0x00 7800) (Continued)

Register Address-Offset 0x000	MSB bits	15	14	13	12	11	10	9	8
	LSB bits	7	6	5	4	3	2	1	0
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
03	DXR1	Bit15	Bit14	Bit13	Bit12	Bit11	Bit10	Bit9	Bit8
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
		Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
		W−0	W−0	W−0	W−0	W−0	W−0	W−0	W−0
20	MFFTX	Reserved	MFFENA	TXFIFO Reset	TXFFST4	TXFFST3	TXFFST2	TXFFST1	TXFFST0
		R−0	R/W−0	R/W−1	R−0	R−0	R−0	R−0	R−0
		TXFFINT Flag	TXFFINT CLR	TXFFIENA	TXFFIL4	TXFFIL3	TXFFIL2	TXFFIL1	TXFFIL0
		R−0	W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
21	MFFRX	RXFFOVF Flag	RXFFOVF CLR	RXFIFO Reset	RXFFST4	RXFFST3	RXFIFST2	RXFFST1	RXFFST0
		R−0	W−0	R/W−1	R−0	R−0	R−0	R−0	R−0
		RXFFINT-Flag	RXFFINT−CLR	RXFFIENA	RXFFIL4	RXFFIL3	RXFFIL2	RXFFIL1	RXFFIL0
		R−0	W−0	R/W−0	R/W−1	R/W−1	R/W−1	R/W−1	R/W−1
22	MFFCT	Reserved							
		R−0							
		FFTXDLY7	FFTXDLY6	FFTXDLY5	FFTXDLY4	FFTXDLY3	FFTXDLY2	FFTXDLY1	FFTXDLY0
		R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0	R/W−0
23	MFFINT	Reserved							
		R−0							
		Reserved				REVTA ENA	RINT ENA	XEVTA ENA	XINT ENA
		R−0				R/W−0	R/W−0	R/W−0	R/W−0
24	MFFST	Reserved							
		R−0							
		Reserved				FSR Flag	EOBR Flag	FSX Flag	EOBX Flag
		R−0				R/W−x	R/W−0	R/W−x	R/W−0

Appendix A

Revision History

This document was revised to SPRU061B from SPRU061A.

The scope of the revisions was limited to adding technical changes as described on the next page.

A.1 Changes Made in This Revision

The following changes were made in this revision:

Page	Additions/Modifications/Deletions
	Global change to title to include new devices
1-1	Added a paragraph regarding 281x devices
6-6	Modified description of CLKSTP field (bits 12−11) in the SPCR1 register

Index

A

B

C

D

E

F

O

P

R

S

T

U

W

X

TMS320x281x DSP Event Manager (EV) Reference Guide

Literature Number: SPRU065B
November 2003

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Contents

Figures

Tables

This page intentionally left blank.

Chapter 1

Overview

The event-manager (EV) modules provide a broad range of functions and features that are particularly useful in motion control and motor control applications. The EV modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits. The two EV modules, EVA and EVB, are identical peripherals, intended for multi-axis/motion-control applications.

Each EV is capable of controlling three Half-H bridges, when each bridge requires a complementary PWM pair for control. Each EV also has two additional PWMs with no complementary outputs.

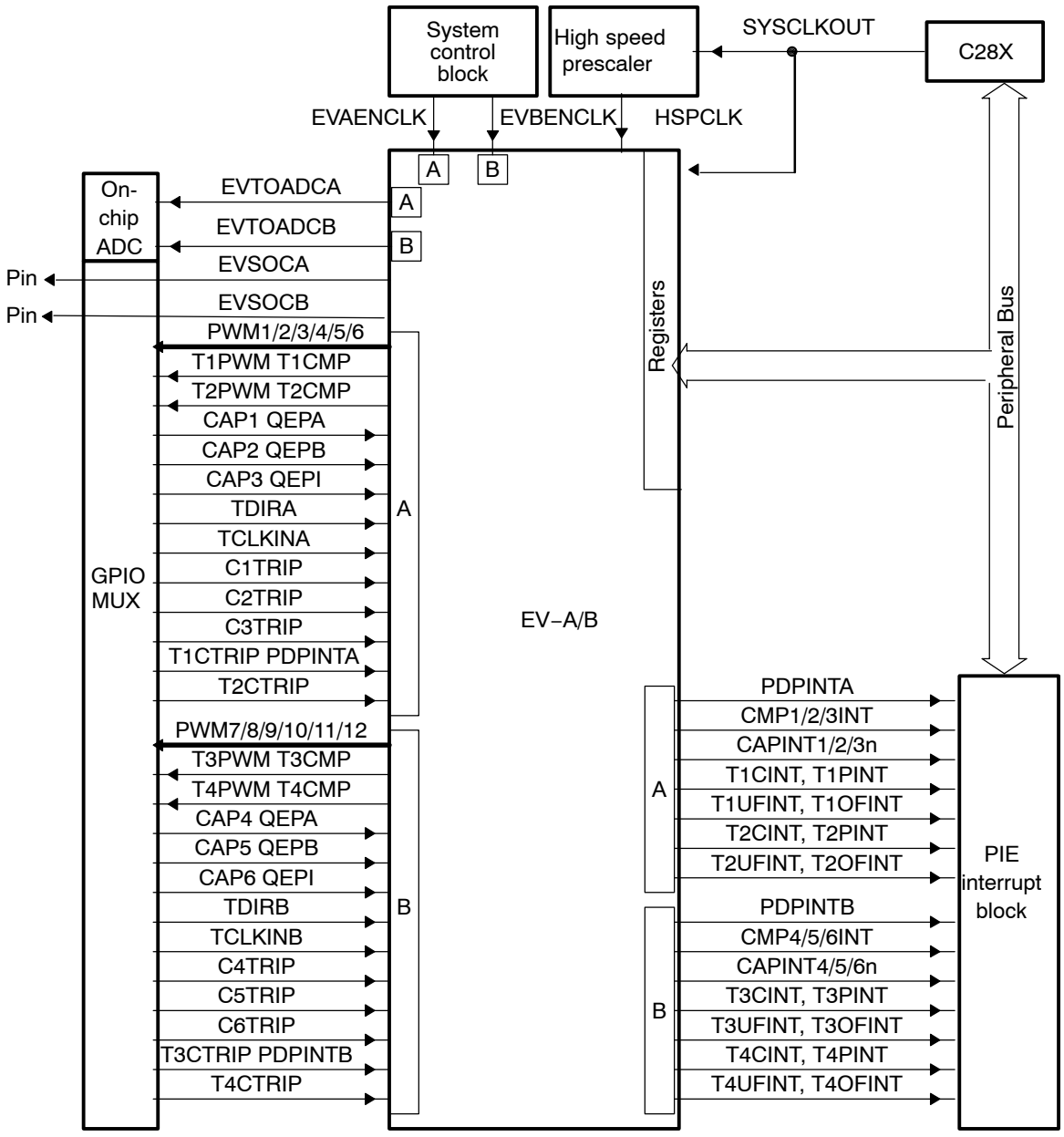
Topic **Page**

1.1 Event Manager Functions

EVA and EVB timers, compare units, and capture units function identically. However, timer/unit names differ for EVA and EVB. Table 1–1 shows the features and functionality available for the event-manager modules and highlights EVA nomenclature.

Event managers A and B have identical peripheral register sets with EVA starting at 7400h and EVB starting at 7500h. The paragraphs in this section describe the function of GP timers, compare units, capture units, and QEPs using EVA nomenclature. These paragraphs are applicable to EVB with regard to function; however, module/signal names differ.

Figure 1−1. Event Manager (EV) Device Interfaces



Note: EXTCONA is an added control register to enable and disable the added/modified features. It is required for compatibility with 240x EV. EXTCONA enables and disables the additions and modifications in features. All additions and modifications are disabled by default to keep compatibility with 240x EV. See Section 5.7 for details about the EXTCONx register.

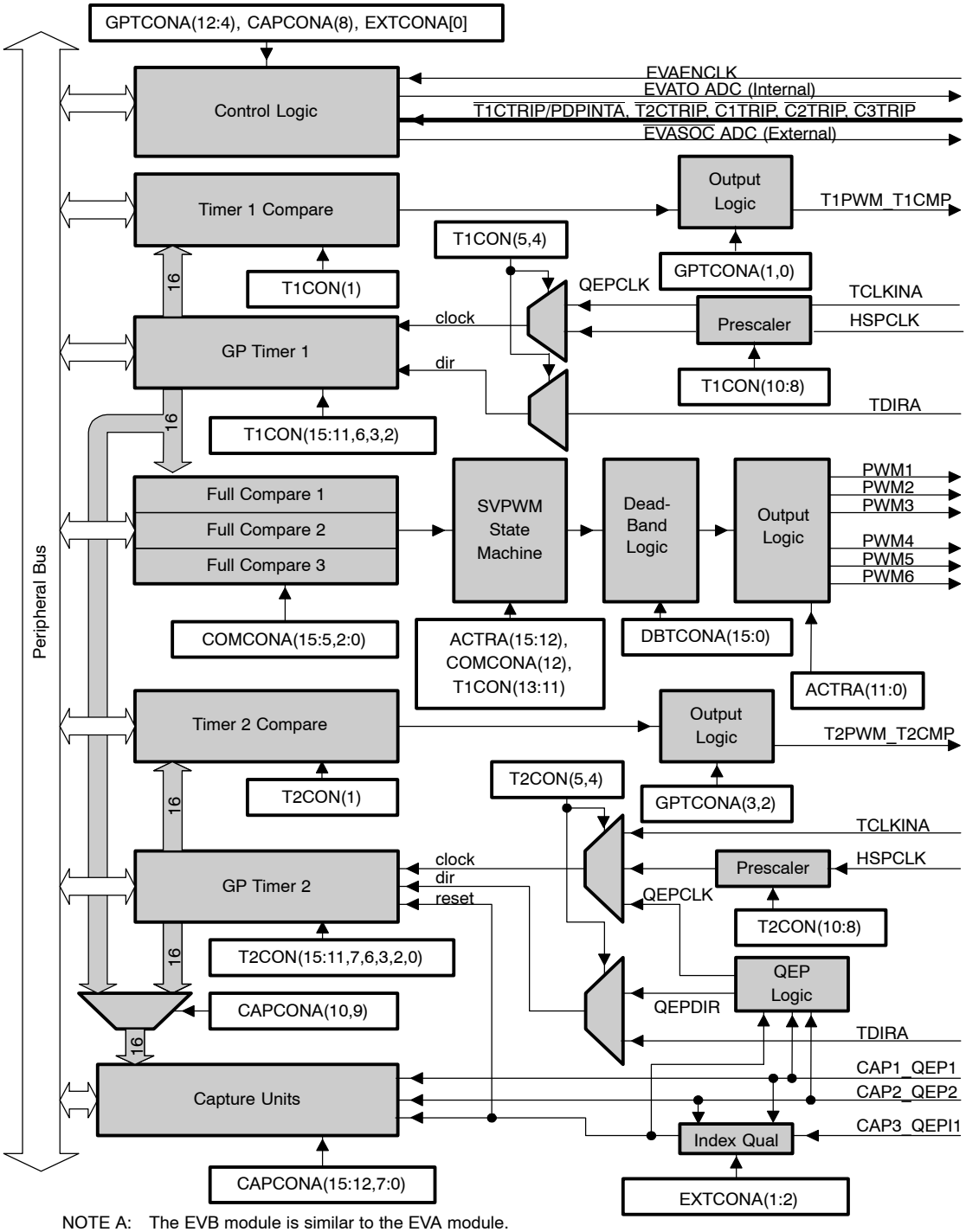
Table 1−1. Module and Signal Names for EVA and EVB

Event Manager Modules	EVA		EVB	
	Module	**Signal**	**Module**	**Signal**
GP timers	GP Timer 1 GP Timer 2	T1PWM/T1CMP T2PWM/T2CMP	GP Timer 3 GP Timer 4	T3PWM/T3CMP T4PWM/T4CMP
Compare units	Compare 1 Compare 2 Compare 3	PWM1/2 PWM3/4 PWM5/6	Compare 4 Compare 5 Compare 6	PWM7/8 PWM9/10 PWM11/12
Capture units	Capture 1 Capture 2 Capture 3	CAP1 CAP2 CAP3	Capture 4 Capture 5 Capture 6	CAP4 CAP5 CAP6
QEP channels	QEP	QEP1 QEP2 QEPI1	QEP	QEP3 QEP4 QEP12
External timer inputs	Timer-direction external clock	TDIRA TCLKINA	Timer-direction external clock	TDIRB TCLKINB
External compare-output trip inputs	Compare	$\overline{\text{C1TRIP}}$ $\overline{\text{C2TRIP}}$ $\overline{\text{C3TRIP}}$		$\overline{\text{C4TRIP}}$ $\overline{\text{C5TRIP}}$ $\overline{\text{C6TRIP}}$
External timer-compare trip inputs		$\overline{\text{T1CTRIP}}$/ $\overline{\text{T2CTRIP}}$		$\overline{\text{T3CTRIP}}$/ $\overline{\text{T4CTRIP}}$
External trip inputs		$\overline{\text{PDPINTA}}$†		$\overline{\text{PDPINTB}}$†
External ADC SOC trigger outputs		EVASOC		EVBSOC

† In the 240x™-compatible mode, the $\overline{\text{T1CTRIP}}$/$\overline{\text{PDPINTA}}$ pin functions as $\overline{\text{PDPINTA}}$ and the $\overline{\text{T3CTRIP}}$/$\overline{\text{PDPINTB}}$ pin functions as $\overline{\text{PDPINTB}}$.

240x is a trademark of Texas Instruments.

Figure 1–2. Event Manager A Functional Block Diagram



NOTE A: The EVB module is similar to the EVA module.

1.1.1 General-Purpose (GP) Timers

There are two GP timers in each EV module. The GP timer x (x = 1 or 2 for EVA; x = 3 or 4 for EVB) includes:

- A 16-bit timer, up-/down-counter, TxCNT, for reads or writes
- A 16-bit timer-compare register, TxCMPR (double-buffered with shadow register), for reads or writes
- A 16-bit timer-period register, TxPR (double-buffered with shadow register), for reads or writes
- A 16-bit timer-control register,TxCON, for reads or writes
- Selectable internal or external input clocks
- A programmable prescaler for internal or external clock inputs
- Control and interrupt logic, for four maskable interrupts: *underflow*, *overflow*, *timer compare*, and *period interrupts*
- A selectable direction input pin (TDIRx) (to count up or down when directional up-/down-count mode is selected)

The GP timers can be operated independently or synchronized with each other. The compare register associated with each GP timer can be used for compare function and PWM-waveform generation. There are three continuous modes of operations for each GP timer in up- or up/down-counting operations. Internal or external input clocks with programmable prescaler are used for each GP timer. GP timers also provide the time base for the other event-manager submodules: GP timer 1 for all the compares and PWM circuits, GP timer 2/1 for the capture units and the quadrature-pulse counting operations. Double-buffering of the period and compare registers allows programmable change of the timer (PWM) period and the compare/PWM pulse width as needed.

1.1.2 Full-Compare Units

There are three full-compare units on each event manager. These compare units use GP timer1 as the time base and generate six outputs for compare and PWM-waveform generation using programmable deadband circuit. The state of each of the six outputs is configured independently. The compare registers of the compare units are double-buffered, allowing programmable change of the compare/PWM pulse widths as needed.

1.1.3 Programmable Deadband Generator

The deadband generator circuit includes three 4-bit counters and an 16-bit compare register. Desired deadband values can be programmed into the

compare register for the outputs of the three compare units. The deadband generation can be enabled/disabled for each compare unit output individually. The deadband-generator circuit produces two outputs (with or without deadband zone) for each compare unit output signal. The output states of the deadband generator are configurable and changeable as needed by way of the double-buffered ACTRx register.

1.1.4 PWM Waveform Generation

Up to eight PWM waveforms (outputs) can be generated simultaneously by each event manager: three independent pairs (six outputs) by the three full-compare units with *programmable deadbands*, and two independent PWMs by the GP-timer compares.

1.1.5 PWM Characteristics

Characteristics of the PWMs are as follows:

- ❏ 16-bit registers
- ❏ Wide range of programmable deadband for the PWM output pairs
- ❏ Change of the PWM carrier frequency for PWM frequency wobbling as needed
- ❏ Change of the PWM pulse widths within and after each PWM period as needed
- ❏ External-maskable power and drive-protection interrupts
- ❏ Pulse-pattern-generator circuit, for programmable generation of asymmetric, symmetric, and eight-space vector PWM waveforms
- ❏ Minimized CPU overhead using auto-reload of the compare and period registers
- ❏ The PWM pins are driven to a high-impedance state when the $\overline{\text{PDPINTx}}$ pin is driven low and **after** $\overline{\text{PDPINTx}}$ signal qualification. The $\overline{\text{PDPINTx}}$ pin (after qualification) is reflected in bit 8 of the COMCONx register.
  - $\overline{\text{PDPINTA}}$ pin status is reflected in bit 8 of COMCONA register.
  - $\overline{\text{PDPINTB}}$ pin status is reflected in bit 8 of COMCONB register.

1.1.6 Capture Unit

The capture unit provides a logging function for different events or transitions. The values of the selected GP timer counter is captured and stored in the two-

level-deep FIFO stacks when selected transitions are detected on capture input pins, CAPx (x = 1, 2, or 3 for EVA; and x = 4, 5, or 6 for EVB). The capture unit consists of three capture circuits.

- Capture units include the following features:
  - One 16-bit capture control register, CAPCONx (R/W)
  - One 16-bit capture FIFO status register, CAPFIFOx
  - Selection of GP timer 1/2 (for EVA) or 3/4 (for EVB) as the time base
  - Three 16-bit 2-level-deep FIFO stacks, one for each capture unit
  - Three capture input pins (CAP1/2/3 for EVA, CAP4/5/6 for EVB)—one input pin per capture unit. [All inputs are synchronized with the device (CPU) clock. In order for a transition to be captured, the input must hold at its current level to meet two rising edges of the device clock. The input pins CAP1/2 and CAP4/5 can also be used as QEP inputs to the QEP circuit.]
  - User-specified transition (rising edge, falling edge, or both edges) detection
  - Three maskable interrupt flags, one for each capture unit

1.1.7 Quadrature-Encoder Pulse (QEP) Circuit

Two capture inputs (CAP1 and CAP2 for EVA; CAP4 and CAP5 for EVB) can be used to interface the on-chip QEP circuit with a quadrature encoder pulse. Full synchronization of these inputs is performed on-chip. Direction or leading-quadrature pulse sequence is detected, and GP timer 2/4 is incremented or decremented by the rising and falling edges of the two input signals (four times the frequency of either input pulse).

1.1.8 External Analog-to-Digital Converter (ADC) Start-of-Conversion

EVA/EVB start-of-conversion (SOC) can be sent to an external pin (EVASOC) for external ADC interface. EVASOC and EVBSOC are MUXed with $\overline{\text{T2CTRIP}}$ and $\overline{\text{T4CTRIP}}$, respectively.

1.1.9 Power Drive Protection Interrupt ($\overline{\text{PDPINTx}}$, x = A or B)

The $\overline{\text{PDPINTx}}$ is a safety feature that is provided for the safe operation of systems such as power converters and motor drives. $\overline{\text{PDPINTx}}$ can be used to inform the monitoring program of motor drive abnormalities such as over-voltage, over-current, and excessive temperature rise. If the $\overline{\text{PDPINTx}}$ inter-

rupt is unmasked, all PWM output pins will be put in the high-impedance state immediately after the $\overline{\text{PDPINTx}}$ pin is driven low. An interrupt will also be generated. See the EXTCONx register bit function for individual pulse-width modulation (PWM) pair, power protection, or trip functions.

The interrupt flag associated with $\overline{\text{PDPINTx}}$ is also set when such an event occurs; however, it must wait until the transition on $\overline{\text{PDPINTx}}$ has been qualified and synchronized with the internal clock. The qualification and synchronization cause a delay of two clock cycles. The setting of the flag does not depend on whether or not the $\overline{\text{PDPINTx}}$ interrupt is masked: it happens when a qualified transition occurs on the $\overline{\text{PDPINTx}}$ pin. This interrupt is enabled following reset. If the $\overline{\text{PDPINTx}}$ interrupt is disabled, the action of driving the PWM outputs to the high-impedance state (upon a valid $\overline{\text{PDPINTx}}$ interrupt) is also disabled.

1.1.10 EV Registers

The EV registers occupy two 64-word (16-bit) frames of address space. The EV module decodes the lower six-bits of the address; while the upper 10 bits of the address are decoded by the peripheral address decode logic, which provides a module select to the Event Manager when the peripheral address bus carries an address within the range designated for the EV on that device.

On 28x devices (as with the C240 device), EVA registers are located in the range 7400h to 7431h. EVB registers are located in the range of 7500h to 7531h.

The undefined registers and undefined bits of the EV registers all return zero when read by user software. Writes have no effect. See Section 1.3, *Event Manager(EV) Register Addresses*, on page 1-12.

1.1.11 EV Interrupts

Each EV interrupt group has multiple interrupt sources, the CPU interrupt requests are processed by the peripheral interrupt expansion (PIE) module. See the *Peripheral Interrupt Expansion Peripheral Reference Guide* (literature number SPRU045) for details. The stages of response are as follows:

1) *Interrupt source*. If peripheral interrupt conditions occur, the respective flag bits in registers EVxIFRA, EVxIFRB, or EVxIFRC (x = A or B) are set. Once set, these flags remain set until explicitly cleared by the software. It is mandatory to clear these flags in the software or future interrupts will not be recognized.

2) *Interrupt enable*. The Event Manager interrupts can be individually enabled or disabled by interrupt mask registers EVxIMRA, EVxIMRB, and

EVxIMRC (x = A or B). Each bit is set to 1 to enable/unmask the interrupt or cleared to 0 to disable/mask the interrupt.

3) *PIE request*. If both interrupt flag bits and interrupt mask bits are set, then the peripheral issues a peripheral interrupt request to the PIE module. The PIE module can receive more than one interrupt from the peripheral. The PIE logic records all the interrupt requests and generates the respective CPU interrupt. (INT1, 2, 3, 4, or 5) based on the preassigned priority of the received interrupts.

4) *CPU response*. On receipt of an INT1, 2, 3, 4, or 5 interrupt request, the respective bit in the CPU interrupt flag register (IFR) will be set. If the corresponding interrupt mask register (IER) bit is set and INTM bit is cleared, then the CPU recognizes the interrupt and issues an acknowledgement to the PIE. Following this, the CPU finishes executing the current instruction and jumps to the interrupt vector address corresponding to INT1.y, 2.y, 3.y, 4.y, or 5.y in the PIE vector table. At this time, the respective IFR bit will be cleared and the INTM bit will be set disabling further interrupt recognition. The interrupt vector contains an address for the interrupt service routine. From here, the interrupt response is controlled by the software.

5) *PIE response*. The PIE logic uses the acknowledge signal from the CPU to clear the PIEIFR bit. See the Peripheral Interrupt Expansion Peripheral Reference Guide (SPRU045) for enabling future interrupts.

6) *Interrupt software*. At this stage, the interrupt software has explicit responsibility to avoid improper interrupt response. After executing the interrupt specific code, the routine should clear the interrupt flag in the EVxIFRA, EVxIFRB, or EVxIFRC that caused the serviced interrupt. Before returning, the interrupt software should re-enable interrupts by clearing respective PIEACKx bits (by writing a 1 to the corresponding bit) and enabling the global interrupt bit INTM.

1.2 Enhanced EV Features

The F2810™ EV is largely the same as the 240x EV. A few enhancements are introduced in the F2810 EV; however, the F2810 EV is backward compatible with the 240x EV. Corresponding bits in the newly added register, EXTCON, must be set for all enhancements and changes to take effect. The following are enhancements and differences of the F2810 EV module with respect to the 240x device:

- Individual output enable bit for each timer and full compare unit
- Dedicated output trip pin for each timer and full compare unit as replacement of the PDPINT pin
- New control register added to activate and configure feature additions and modifications. This is key to maintaining compatibility.
- Trip enable bit for each trip pin. These changes allow the outputs of each compare to be enabled and disabled independently so that each compare can control a separate power stage, actuator, or drive.
- Renamed CAP3 pin can function as CAP3_QEPI (CAP3_QEPI1 for EVA, CAP6_QEPI2 for EVB). This pin is now allowed to reset Timer 2 when enabled. Also introduced a qualification mode where QEP1 and QEP2 can be used to qualify CAP3_QEPI. The QEP channel (3 pin) enables seamless interface to industry-standard three-signal quadrature encoders.
- EV ADC start-of-conversion outputs to allow synchronization with high-precision external ADCs.

F2810 is a trademark of Texas Instruments.

1.3 Event Manager (EV) Register Addresses

All EV-A registers are listed in Table 1−2 and EV-B are listed in Table 1−3.

Table 1−2. Summary of EV-A Registers

Name	Address	Description
Timer Registers		
GPTCONA	0x7400	Overall GP Timer Control Register A
T1CNT	0x7401	Timer 1 Counter Register
T1CMPR	0x7402	Timer 1 Compare Register
T1PR	0x7403	Timer 1 Period Register
T1CON	0x7404	Timer 1 Control Register
T2CNT	0x7405	Timer 2 Counter Register
T2CMPR	0x7406	Timer 2 Compare Register
T2PR	0x7407	Timer 2 Period Register
T2CON	0x7408	Timer 2 Control Register
EXTCONA	0x7409	Extension Control Register A
Compare Registers		
COMCONA	0x7411	Compare Control Register A
ACTRA	0x7413	Compare Action Control Register A
DBTCONA	0x7415	Dead−Band Timer Control Register A
CMPR1	0x7417	Compare Register 1
CMPR2	0x7418	Compare Register 2
CMPR3	0x7419	Compare Register 3
Capture Registers		
CAPCONA	0x7420	Capture Control Register A
CAPFIFOA	0x7422	Capture FIFO Status Register A
CAP1FIFO	0x7423	Two−Level Deep Capture FIFO Stack 1
CAP2FIFO	0x7424	Two−Level Deep Capture FIFO Stack 2
CAP3FIFO	0x7425	Two−Level Deep Capture FIFO Stack 3

Table 1–2. Summary of EV-A Registers (Continued)

Name	Address	Description
CAP1FBOT	0x7427	Bottom Register Of Capture FIFO Stack 1
CAP2FBOT	0x7428	Bottom Register Of Capture FIFO Stack 2
CAP3FBOT	0x7429	Bottom Register Of Capture FIFO Stack 3
Interrupt Registers		
EVAIMRA	0x742C	Interrupt Mask Register A
EVAIMRB	0x742D	Interrupt Mask Register B
EVAIMRC	0x742E	Interrupt Mask Register C
EVAIFRA	0x742F	Interrupt Flag Register A
EVAIFRB	0x7430	Interrupt Flag Register B
EVAIFRC	0x7431	Interrupt Flag Register C

Table 1–3. Summary of EV-B Registers

Name	Address	Description
Timer Registers		
GPTCONB	0x7500	Overall GP Timer Control Register B
T3CNT	0x7501	Timer 3 Counter Register
T3CMPR	0x7502	Timer 3 Compare Register
T3PR	0x7503	Timer 3 Period Register
T3CON	0x7504	Timer 3 Control Register
T4CNT	0x7505	Timer 4 Counter Register
T4CMPR	0x7506	Timer 4 Compare Register
T4PR	0x7507	Timer 4 Period Register
T4CON	0x7508	Timer 4 Control Register
EXTCONB	0x7509	Extension Control Register B
Compare Registers		
COMCONB	0x7511	Compare Control Register B

Table 1–3. Summary of EV-B Registers (Continued)

ACTRB	0x7513	Compare Action Control Register B
DBTCONB	0x7515	Dead–Band Timer Control Register B
CMPR4	0x7517	Compare Register 4
CMPR5	0x7518	Compare Register 5
CMPR6	0x7519	Compare Register 6
Capture Registers		
CAPCONB	0x7520	Capture Control Register B
CAPFIFOB	0x7522	Capture FIFO Status Register B
CAP4FIFO	0x7523	Two–Level Deep Capture FIFO Stack 4
CAP5FIFO	0x7524	Two–Level Deep Capture FIFO Stack 5
CAP6FIFO	0x7525	Two–Level Deep Capture FIFO Stack 6
CAP4FBOT	0x7527	Bottom Register Of Capture FIFO Stack 4
CAP5FBOT	0x7528	Bottom Register Of Capture FIFO Stack 5
CAP6FBOT	0x7529	Bottom Register Of Capture FIFO Stack 6
Interrupt Registers		
EVBIMRA	0x752C	Interrupt Mask Register A
EVBIMRB	0x752D	Interrupt Mask Register B
EVBIMRC	0x752E	Interrupt Mask Register C
EVBIFRA	0x752F	Interrupt Flag Register A
EVBIFRB	0x7530	Interrupt Flag Register B
EVBIFRC	0x7531	Interrupt Flag Register C

1.4 General-Purpose (GP) Timers

There are two general-purpose (GP) timers in each module. These timers can be used as independent time bases in applications such as:

- The generation of a sampling period in a control system
- Providing a time base for the operation of the quadrature encoder pulse (QEP) circuit (GP timer 2/4 only) and the capture units
- Providing a time base for the operation of the compare units and associated PWM circuits to generate PWM outputs

1.4.1 Timer Functional Blocks

Figure 1–3 shows a block diagram of a GP timer. Each GP timer includes:

- One readable and writable (RW) 16-bit up and up/down counter register TxCNT (x = 1, 2, 3, 4). This register stores the current value of the counter and keeps incrementing or decrementing depending on the direction of counting
- One RW 16-bit timer compare register (shadowed), TxCMPR (x = 1, 2, 3, 4)
- One RW 16-bit timer period register (shadowed), TxPR (x = 1, 2, 3, 4)
- RW 16-bit individual timer control register, TxCON (x = 1, 2, 3, 4)
- Programmable prescaler applicable to both internal and external clock inputs
- Control and interrupt logic
- One GP timer compare output pin, TxCMP (x = 1, 2, 3, 4)
- Output conditioning logic

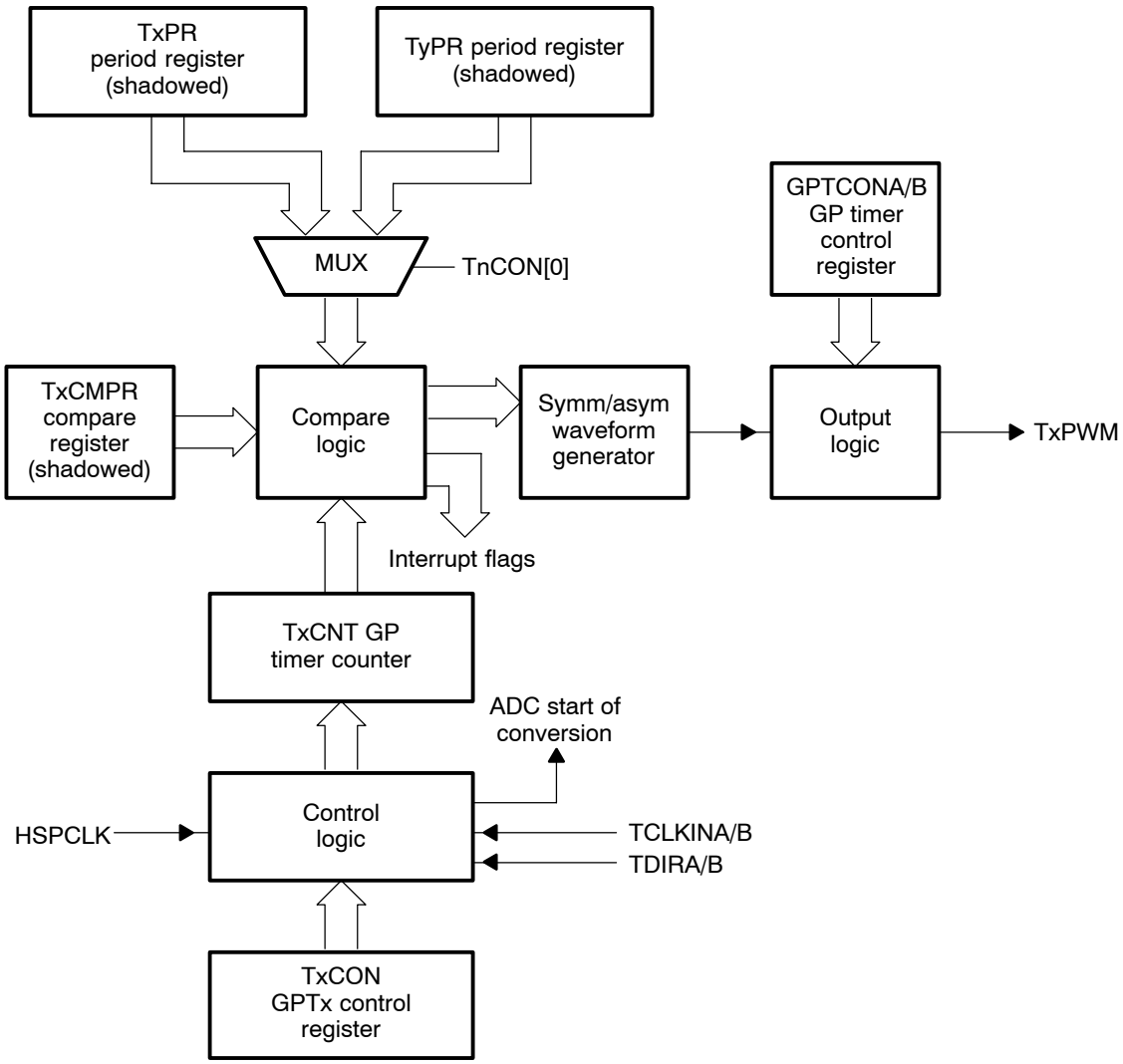
Another overall control register, GPTCONA/B, specifies the action to be taken by the timers on different timer events, and indicates the counting directions of the GP timers. GPTCONA/B is readable and writable, although writing to the status bits has no effect.

Note:

Timer 2 can select the period register of timer 1 as its period register. In Figure 1–3, the MUX is applicable only when the figure represents timer 2.

Timer 4 can select the period register of timer 3 as its period register. In Figure 1–3, the MUX is applicable only when the figure represents timer 4.

Figure 1–3. General-Purpose Timer Block Diagram (x = 2 or 4) [when x = 2: y = 1 and n = 2; when x = 4: y = 3 and n = 4]



1.4.2 GP Timer Inputs

The inputs to the GP timers are:

- The internal HSPCLK
- An external clock, TCLKINA/B, that has a maximum frequency of one-fourth that of the device clock
- Direction input, TDIRA/B, for use by the GP timers in directional up-/down-counting mode

- Reset signal, RESET

When a timer is used with the QEP circuit, the QEP circuit generates both the timer's clock and the counting direction.

1.4.3 GP Timer Outputs

The outputs of the timers are:

- GP timer compare outputs TxCMP, x = 1, 2, 3, 4
- ADC start-of-conversion signal to ADC module
- Underflow, overflow, compare match, and period match signals to its own compare logic and to the compare units
- Counting direction indication bits

1.4.4 Individual GP Timer Control Register (TxCON)

The operational mode of a timer is controlled by its individual control register TxCON. Bits in the TxCON register determine:

- Which of the four counting modes the timer is in
- Whether an internal or external clock is to be used by the GP timer
- Which of the eight input clock prescale factors (ranging from 1 to 1/128) is used
- On which condition the timer compare register is reloaded
- Whether the timer is enabled or disabled
- Whether the timer compare operation is enabled or disabled
- Which period register is used by timer 2, its own, or timer 1's period register (EVA)
  Which period register is used by timer 4, its own, or timer 3's period register (EVB)

1.4.5 Overall GP Timer Control Register (GPTCONA/B)

The control register GPTCONA/B specifies the action to be taken by the timers on different timer events and indicates their counting directions.

1.4.6 GP Timer Compare Registers

The compare register associated with a GP timer stores the value to be constantly compared with the counter of the GP timer. When a match happens, the following events occur:

- A transition occurs on the associated compare output according to the bit pattern in GPTCONA/B
- The corresponding interrupt flag is set
- A peripheral interrupt request is generated if the interrupt is unmasked

The compare operation of a GP timer can be enabled or disabled by the appropriate bit in TxCON.

The compare operation and outputs can be enabled in any of the timer modes, including QEP mode.

1.4.7 GP Timer Period Register

The value in the period register of a GP timer determines the period of the timer. A GP timer resets to 0, or starts counting downward when a match occurs between the period register and the timer counter, depending on which counting mode the timer is in.

1.4.8 Double Buffering of GP Timer Compare and Period Registers

The compare and period registers, TxCMPR and TxPR, of a GP timer are shadowed. A new value can be written to any of these registers at any time during a period. However, the new value is written to the associated shadow register. For the compare register, the content in the shadow register is loaded into the working (active) register only when a certain timer event specified by TxCON occurs. For the period register, the working register is reloaded with the value in its shadow register only when the value of the counter register TxCNT is 0. The condition on which a compare register is reloaded can be one of the following:

- Immediately after the shadow register is written
- On underflow; that is, when the GP timer counter value is 0
- On underflow or period match; that is, when the counter value is 0 or when the counter value equals the value of the period register

The double buffering feature of the period and compare registers allows the application code to update the period and compare registers at any time during a period in order to change the timer period and the width of the PWM pulse for the period that follows. On-the-fly change of the timer period value, in the case of PWM generation, means on-the-fly change of PWM carrier frequency.

Note: Period Register Initialization

The period register of a GP timer should be initialized before its counter is initialized to a non-zero value. Otherwise, the value of the period register will remain unchanged until the next underflow.

A compare register is transparent (the newly loaded value goes directly into the active register) when the associated compare operation is disabled. This applies to all Event Manager compare registers.

1.4.9 GP Timer Compare Output

The compare output of a GP timer can be specified active high, active low, forced high, or forced low, depending on how the GPTCONA/B bits are configured. It goes from low to high (high to low) on the first compare match when it is active high (low). It then goes from high to low (low to high) on the second compare match if the GP timer is in an up-/down-counting mode, or on period match if the GP timer is in up-counting mode. The timer compare output becomes high (low) right away when it is specified to be forced high (low).

1.4.10 Timer Counting Direction

The counting directions of the GP timers are reflected by their respective bits in GPTCONA/B during all timer operations as follows:

- 1 represents the up-counting direction
- 0 represents the down-counting direction

The input pin TDIRA/B determines the direction of counting when a GP timer is in directional up-/down-counting mode. When TDIRA/B is high, upward counting is specified; when TDIRA/B is low, downward counting is specified.

1.4.11 Timer Clock

The source of the GP timer clock can be the internal device clock or the external clock input, TCLKINA/B. The frequency of the external clock must be less than or equal to one-fourth of that of the device clock. GP timer 2 (EVA) and GP timer 4 (EVB) can be used with the QEP circuits, in directional up-/down-counting mode. In this case, the QEP circuits provide both the clock and direction inputs to the timer.

A wide range of prescale factors are provided for the clock input to each GP timer.

1.4.12 QEP-Based Clock Input

The quadrature encoder pulse (QEP) circuit, when selected, can generate the input clock and counting direction for GP timer 1/2/3/4 (QEPCLK is one of the clock sources for Timer 1) in the directional up/down-counting mode. This input clock cannot be scaled by GP timer prescaler circuits (that is, the prescaler

of the selected GP timer is always one if the QEP circuit is selected as the clock source). Furthermore, the frequency of the clock generated by the QEP circuits is four times that of the frequency of each QEP input channel because both the rising and falling edges of both QEP input channels are counted by the selected timer. The frequency of the QEP input must be less than or equal to one-fourth of that of the device clock.

1.4.13 GP Timer Synchronization

GP timer 2 can be synchronized with GP timer 1 (for EVA) and GP timer 4 can be synchronized with GP timer 3 (for EVB) by proper configuration of T2CON and T4CON, respectively, in the following ways:

- EVA:
  Set the T2SWT1 bit in T2CON to start GP timer 2 counting with the TENABLE bit in T1CON (thus, both timer counters start simultaneously)
- EVA:
  Initialize the timer counters in GP timers 1 and 2 with different values before starting synchronized operation
- EVA:
  Specify that GP timer 2 uses the period register of GP timer 1 as its period register (ignoring its own period register) by setting SELT1PR in T2CON
- EVB:
  Set the T4SWT3 bit in T4CON to start GP timer 4 counting with the TENABLE bit in T3CON (thus, both timer counters start simultaneously)
- EVB:
  Initialize the timer counters in GP timers 3 and 4 with different values before starting synchronized operation
- EVB:
  Specify that GP timer 4 uses the period register of GP timer 3 as its period register (ignoring its own period register) by setting SELT3PR in T4CON

This allows the desired synchronization between GP timer events. Since each GP timer starts the counting operation from its current value in the counter register, one GP timer can be programmed to start with a known delay after the other GP timer.

1.4.14 Starting the A/D Converter With a Timer Event

The bits in GPTCONA/B can specify that an ADC start signal be generated on a GP timer event such as underflow, compare match, or period match. This

feature provides synchronization between the GP timer event and the ADC start without any CPU intervention.

1.4.15 GP Timer in Emulation Suspend

The GP timer control register bits also define the operation of the GP timers during emulation suspend. These bits can be set to allow the operation of GP timers to continue when an emulation interrupt occurs making in-circuit emulation possible. They can also be set to specify that the operation of GP timers stops immediately, or after completion of the current counting period, when emulation interrupt occurs.

Emulation suspend occurs when the device clock is stopped by the emulator, for example, when the emulator encounters a break point.

1.4.16 GP Timer Interrupts

There are sixteen interrupt flags in the EVAIFRA, EVAIFRB, EVBIFRA, and EVBIFRB registers for the GP timers. Each of the four GP timers can generate four interrupts upon the following events:

- Overflow: TxOFINT (x = 1, 2, 3, or 4)
- Underflow: TxUFINT (x = 1, 2, 3, or 4)
- Compare match: TxCINT (x = 1, 2, 3, or 4)
- Period match: TxPINT (x = 1, 2, 3, or 4)

A timer compare event (match) happens when the content of a GP timer counter is the same as that of the compare register. The corresponding compare interrupt flag is set one clock cycle after the match if the compare operation is enabled.

An overflow event occurs when the value of the timer counter reaches FFFFh. An underflow event occurs when the timer counter reaches 0000h. Similarly, a period event happens when the value of the timer counter is the same as that of the period register. The overflow, underflow, and period interrupt flags of the timer are set one clock cycle after the occurrence of each individual event. Note that the definition of overflow and underflow is different from their conventional definitions.

1.4.17 GP Timer Counting Operation

Each GP timer has four possible modes of operation:

- Stop/Hold mode
- Continuous Up-Counting mode
- Directional Up-/Down-Counting mode
- Continuous Up-/Down-Counting mode

The bit pattern in the corresponding timer control register TxCON determines the counting mode of a GP timer. The timer enabling bit, TxCON[6], enables or disables the counting operation of a timer. When the timer is disabled, the counting operation of the timer stops and the prescaler of the timer is reset to x/1. When the timer is enabled, the timer starts counting according to the counting mode specified by other bits of TxCON.

1.4.18 Stop/Hold Mode

In this mode the GP timer stops and holds at its current state. The timer counter, the compare output, and the prescale counter all remain unchanged in this mode.

1.4.19 Continuous Up-Counting Mode

The GP timer in this mode counts up according to the scaled input clock until the value of the timer counter matches that of the period register. On the next rising edge of the input clock after the match, the GP timer resets to zero and starts counting up again.

The period interrupt flag of the timer is set one clock cycle after the match between the timer counter and period register. A peripheral interrupt request is generated if the flag is not masked. An ADC start is sent to the ADC module at the same time the flag is set, if the period interrupt of this timer has been selected by the appropriate bits in GPTCONA/B to start the ADC.

One clock cycle after the GP timer becomes 0, the underflow interrupt flag of the timer is set. A peripheral interrupt request is generated by the flag if it is unmasked. An ADC start is sent to the ADC module at the same time if the underflow interrupt flag of this timer has been selected by appropriate bits in GPTCONA/B to start ADC.

The overflow interrupt flag is set one clock cycle after the value in TxCNT matches FFFFh. A peripheral interrupt request is generated by the flag if it is unmasked.

The duration of the timer period is (TxPR) + 1 cycles of the scaled clock input except for the first period. The duration of the first period is the same if the timer counter is zero when counting starts.

The initial value of the GP timer can be any value between 0h and FFFFh inclusive. When the initial value is greater than the value in the period register, the timer counts up to FFFFh, resets to zero, and continues the operation as if the initial value was zero. When the initial value in the timer counter is the same as that of the period register, the timer sets the period interrupt flag, resets to

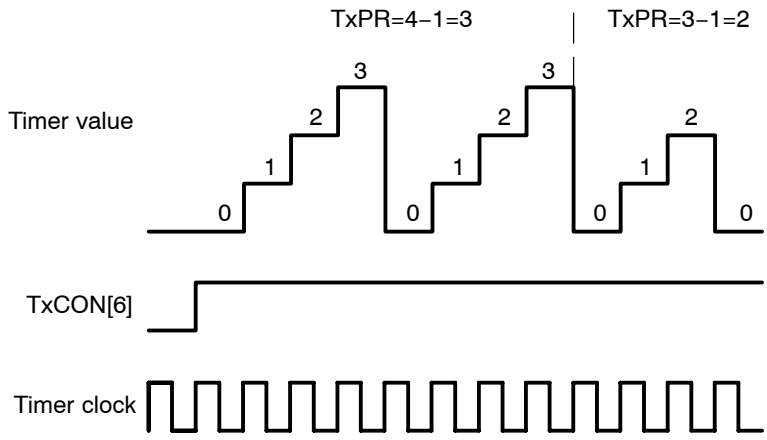
zero, sets the underflow interrupt flag, and then continues the operation again as if the initial value was zero. If the initial value of the timer is between zero and the contents of the period register, the timer counts up to the period value and continue to finish the period as if the initial counter value was the same as that of the period register.

The counting direction indication bit in GPTCONA/B is one for the timer in this mode. Either the external or internal device clock can be selected as the input clock to the timer. TDIRA/B input is ignored by the GP timer in this counting mode.

The continuous up-counting mode of the GP timer is particularly useful for the generation of edge-triggered or asynchronous PWM waveforms and sampling periods in many motor and motion control systems.

Figure 1−4 shows the continuous up-counting mode of the GP timer.

Figure 1−4. GP Timer Continuous Up-Counting Mode (TxPR = 3 or 2)



As shown in Figure 1−4, *GP Timer Continuous Up-Counting Mode (TxPR = 3 or 2)*, no clock cycle is missed from the time the counter reaches the period register value to the time it starts another counting cycle.

1.4.20 Directional Up-/Down-Counting Mode

The GP timer in directional up-/down-counting mode counts up or down according to the scaled clock and TDIRA/B inputs. The GP timer starts counting up until its value reaches that of the period register (or FFFFh if the initial count is greater than the period) when the TDIRA/B pin is held high. When the timer value equals that of its period register (or FFFFh) the timer resets to zero and continues counting up to the period again. When TDIRA/B is held low, the GP

timer counts down until its value becomes zero. When the value of the timer has counted down to zero, the timer reloads its counter with the value in the period register and starts counting down again.

The initial value of the timer can be any value between 0000h to FFFFh. When the initial value of the timer counter is greater than that of the period register, the timer counts up to FFFFh before resetting itself to zero and counting up to the period. If TDIRA/B is low when the timer starts with a value greater than the period register, it counts down to the value of the period register and continues counting down to zero, at which point the timer counter gets reloaded with the value from the period register as normal.

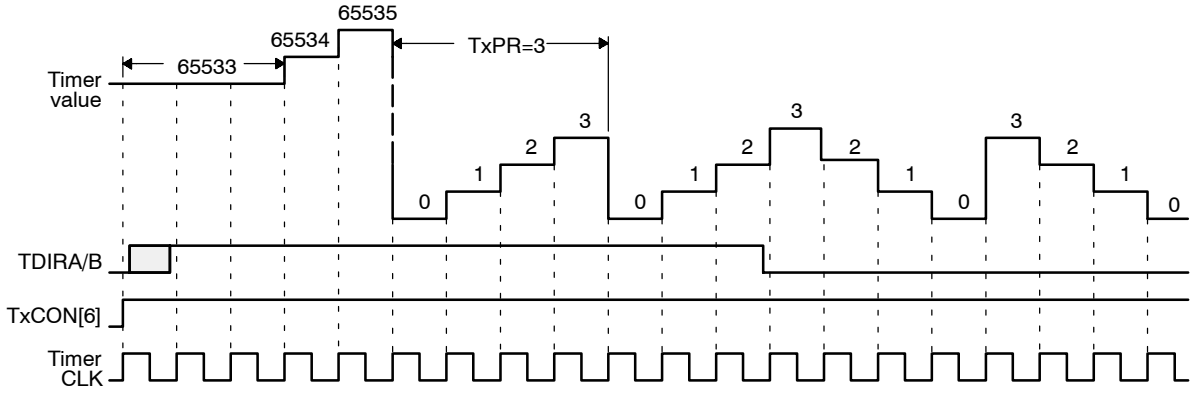
The period, underflow, and overflow interrupt flags, interrupts, and associated actions are generated on respective events in the same manner as they are generated in the continuous up-counting mode.

The latency from a change of TDIRA/B to a change of counting direction is one clock cycle after the end of the current count (that is, after the end of the current prescale counter period).

The direction of counting is indicated for the timer in this mode by the corresponding direction indication bit in GPTCONA/B: 1 means counting up; 0 means counting down. Either the external clock from the TCLKINA/B pin or the internal device clock can be used as the input clock for the timer in this mode.

Figure 1–5 shows the directional up-/down-counting mode of the GP timers.

Figure 1–5. GP Timer Directional Up-/Down-Counting Mode: Prescale Factor 1 and TxPR = 3



The directional up-/down-counting mode of GP timer 2/4 can be used with the quadrature encoder pulse (QEP) circuits in the EV module. The QEP circuits

provide both the counting clock and direction for GP timer 2/4 in this case. This mode of operation can also be used to time the occurrence of external events in motion/motor control and power electronics applications.

1.4.21 Continuous Up-/Down-Counting Mode

This mode of operation is the same as the directional up-/down-counting mode, but the TDIRA/B pin has no effect on the counting direction. The counting direction only changes from up to down when the timer reaches the period value (or FFFFh if the initial timer value is greater than the period). The timer direction only changes from down to up when the timer reaches zero.

The period of the timer in this mode is 2*(TxPR) cycles of the scaled clock input, except for the first period. The duration of the first counting period is the same if the timer counter is zero when counting starts.

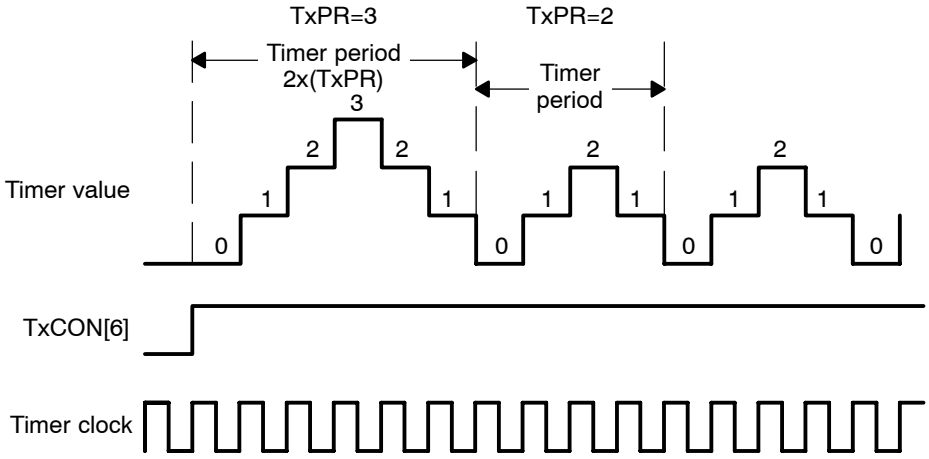
The initial value of the GP timer counter can be any value between 0h and FFFFh inclusive. When the initial value is greater than that of the period register, the timer counts up to FFFFh, resets to zero, and continues the operation as if the initial value was zero. When the initial value in the timer counter is the same as that of the period register, the timer counts down to zero and continues again as if the initial value was zero. If the initial value of the timer is between zero and the contents of the period register, the timer counts up to the period value and continues to finish the period as if the initial counter value was the same as that of the period register.

The period, underflow, and overflow interrupt flags, interrupts, and associated actions are generated on respective events in the same manner as they are generated in continuous up-counting mode.

The counting direction indication bit for this timer in GPTCONA/B is one when the timer counts upward and zero when the timer counts downward. Either the external clock from the TCLKINA/B pin or the internal device clock can be selected as the input clock. TDIRA/B input is ignored by the timer in this mode.

Figure 1−6 shows the continuous up-/down-counting mode of the GP timer.

Figure 1–6. GP Timer Continuous Up-/Down-Counting Mode (TxPR = 3 or 2)



Continuous up-/down-counting mode is particularly useful in generating centered or symmetric PWM waveforms found in a broad range of motor/motion control and power electronics applications.

1.4.22 GP Timer Compare Operation

Each GP timer has an associated compare register TxCMPR and a PWM output pin TxPWM. The value of a GP timer counter is constantly compared to that of its associated compare register. A compare match occurs when the value of the timer counter is the same as that of the compare register. Compare operation is enabled by setting TxCON[1] to one. If it is enabled, the following happens on a compare match:

- The compare interrupt flag of the timer is set one clock cycle after the match
- A transition occurs on the associated PWM output according to the bit configuration in GPTCONA/B, one device clock cycle after the match
- If the compare interrupt flag has been selected by the appropriate GPTCONA/B bits to start ADC, an ADC start signal is generated at the same time the compare interrupt flag is set

A peripheral interrupt request is generated by the compare interrupt flag if it is unmasked.

1.4.23 PWM Transition

The transition on the PWM output is controlled by an asymmetric and symmetric waveform generator and the associated output logic, and depends on the following:

- ❏ Bit definition in GPTCONA/B
- ❏ Counting mode the timer is in
- ❏ Counting direction when the counting mode is continuous-up/-down mode

1.4.24 Asymmetric/Symmetric Waveform Generator

The asymmetric/symmetric waveform generator generates an asymmetric or symmetric PWM waveform based on the counting mode the GP timer is in.

Asymmetric Waveform Generation

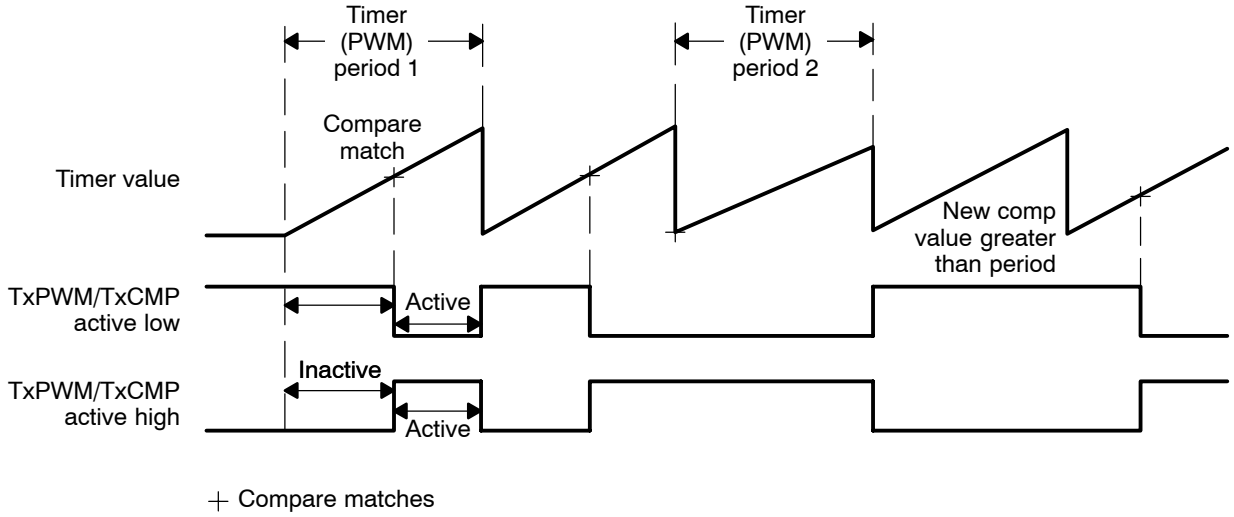
An asymmetric waveform (Figure 1–7) is generated when the GP timer is in continuous up-counting mode. When the GP timer is in this mode, the output of the waveform generator changes according to the following sequence:

- ❏ zero before the counting operation starts
- ❏ remains unchanged until the compare match happens
- ❏ toggles on compare match
- ❏ remains unchanged until the end of the period
- ❏ resets to zero at the end of a period on period match, if the new compare value for the following period is not zero

The output is one for the whole period, if the compare value is zero at the beginning of a period. The output does not reset to zero if the new compare value for the following period is zero. This is important because it allows the generation of PWM pulses of 0% to 100% duty cycle without glitches. The output is zero for the whole period if the compare value is greater than the value in the period register. The output is one for one cycle of the scaled clock input if the compare value is the same as that of the period register.

One characteristic of asymmetric PWM waveforms is that a change in the value of the compare register only affects one side of the PWM pulse.

Figure 1–7. GP Timer Compare/PWM Output in Up-Counting Mode



Symmetric Waveform Generation

A symmetric waveform (Figure 1–8) is generated when the GP timer is in continuous up-/down-counting modes. When the GP timer is in this mode, the state of the output of the waveform generator is determined by the following:

- ❏ Zero before the counting operation starts
- ❏ Remains unchanged until first compare match
- ❏ Toggles on the first compare match
- ❏ Remains unchanged until the second compare match
- ❏ Toggles on the second compare match
- ❏ Remains unchanged until the end of the period
- ❏ Resets to zero at the end of the period if there is no second compare match, and the new compare value for the following period is not zero

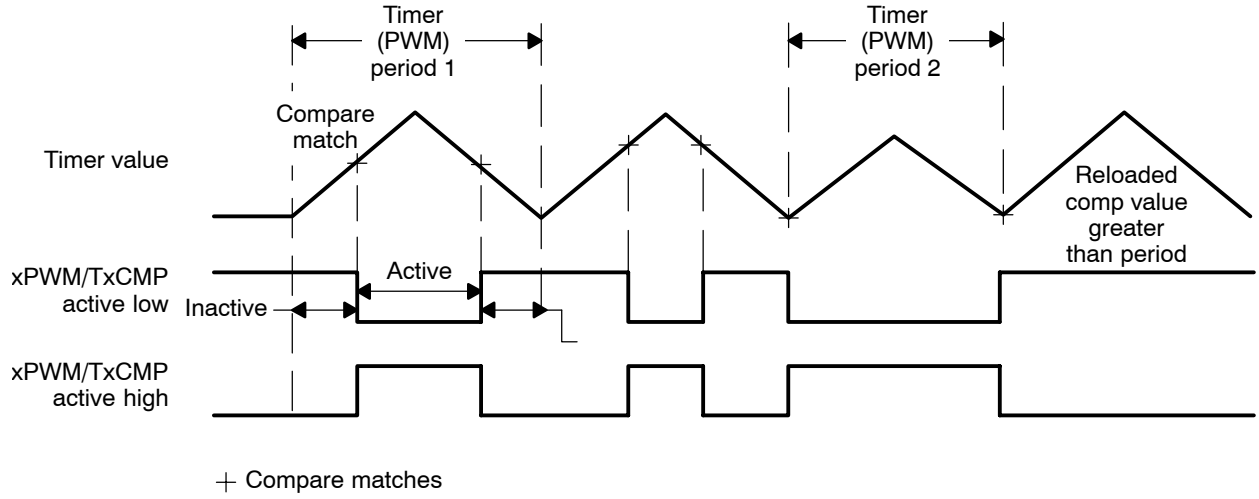
The output is set to one at the beginning of a period and remains one until the second compare match if the compare value is zero at the beginning of a period. After the first transition, the output remains one until the end of the period if the compare value is zero for the second half of the period. When this happens, the output does not reset to zero if the new compare value for the following period is still zero. This is done again to assure the generation of PWM pulses of 0% to 100% duty cycle without any glitches. The first transition does

not happen if the compare value is greater than or equal to that of the period register for the first half of the period. However, the output still toggles when a compare match happens in the second half of the period. This error in output transition, often as a result of calculation error in the application routine, is corrected at the end of the period because the output resets to zero, unless the new compare value for the following period is zero. In this case, the output remains one, which again puts the output of the waveform generator in the correct state.

Note:

The output logic determines what the active state is for all output pins.

Figure 1–8. GP Timer Compare/PWM Output in Up-/Down-Counting Modes



Output Logic

The output logic further conditions the output of the waveform generator to form the ultimate PWM output that controls different kinds of power devices. The PWM output can be specified active high, active low, forced low, and forced high by proper configuration of the appropriate GPTCONA/B bits.

The polarity of the PWM output is the same as that of the output of the associated asymmetric/symmetric waveform generator when the PWM output is specified active high.

The polarity of the PWM output is the opposite of that of the output of the associated asymmetric/symmetric waveform generator when the PWM output is specified active low.

The PWM output is set to one (or zero) immediately after the corresponding bits in GPTCONA/B are set, and the bit pattern specifies that the state of PWM output is forced high (or low).

In summary, during a normal counting mode, transitions on the GP timer PWM outputs happen according to Table 1–4 for the continuous up-counting mode and according to Table 1–5 for the continuous up-/down-counting mode, assuming compare is enabled.

Setting active means setting high for active high and setting low for active low. Setting inactive means the opposite.

The asymmetric/symmetric waveform generation, based on the timer counting mode and the output logic, is also applicable to the compare units.

Table 1–4. GP Timer Compare Output in Continuous Up-Counting Modes

Time in a Period	State of Compare Output
Before compare match	Inactive
On compare match	Set active
On period match	Set inactive

Table 1–5. GP Timer Compare Output in Continuous Up-/Down-Counting Modes

Time in a Period	State of Compare Output
Before 1st compare match	Inactive
On 1st compare match	Set active
On 2nd compare match	Set inactive
After 2nd compare match	Inactive

All GP timer PWM outputs are put in the high-impedance state when any of the following events occurs:

- GPTCONA/B[6] is set to zero by software
- $\overline{\text{PDPINTx}}$ is pulled low and is not masked
- Any reset event occurs
- TxCON[1] is set to zero by software

1.4.25 Active/Inactive Time Calculation

For the continuous up-counting mode, the value in the compare register represents the elapsed time between the beginning of a period and the occurrence of the first compare match (length of the inactive phase). This elapsed time is equal to the period of the scaled input clock multiplied by the value of TxCMPR. Therefore, the length of the active phase (the output pulse width) is given by (TxPR) − (TxCMPR) + 1 cycle of the scaled input clock.

For the continuous up-/down-counting mode, the compare register can have a different value while counting down from the value while counting up. The length of the active phase (output pulse width) for up-/down-counting modes is given by (TxPR) − $(TxCMPR)_{up}$ + (TxPR) − $(TxCMPR)_{dn}$ cycles of the scaled input clock, where $(TxCMPR)_{up}$ is the compare value on the way up and $(TxCMPR)_{dn}$ is the compare value on the way down.

When the value in TxCMPR is zero, the GP timer compare output is active for the whole period if the timer is in the up-counting mode. For the up-/down-counting mode, the compare output is active at the beginning of the period if $(TxCMPR)_{up}$ is zero. The output remains active until the end of the period if $(TxCMPR)_{dn}$ is also zero.

The length of the active phase (the output pulse width) is zero when the value of TxCMPR is greater than that of TxPR for up-counting modes. For the up-/down-counting mode, the first transition is lost when $(TxCMPR)_{up}$ is greater than or equal to (TxPR). Similarly, the second transition is lost when $(TxCMPR)_{dn}$ is greater than or equal to (TxPR). The GP timer compare output is inactive for the entire period if both $(TxCMPR)_{up}$ and $TxCMPR)_{dn}$ are greater than or equal to (TxPR) for the up-/down-counting mode.

Figure 1−7, *GP Timer Compare/PWM Output in Up-Counting Mode* (page 1-28) shows the compare operation of a GP timer in the up-counting mode. Figure 1−8, *GP Timer Compare/PWM Output in Up-/Down-Counting Modes* (page 1-29) shows the compare operation of a GP timer in the up-/down-counting mode.

1.5 Generation of PWM Outputs Using the GP Timers

Each GP timer can independently be used to provide a PWM output channel. Thus, up to two PWM outputs may be generated by the GP timers.

1.5.1 PWM Operation

To generate a PWM output with a GP timer, a continuous up- or up-/down-counting mode can be selected. Edge-triggered or asymmetric PWM waveforms are generated when a continuous-up count mode is selected. Centered or symmetric PWM waveforms are generated when a continuous-up/-down mode is selected. To set up the GP timer for the PWM operation, do the following:

- ❑ Set up TxPR according to the desired PWM (carrier) period
- ❑ Set up TxCON to specify the counting mode and clock source, and start the operation
- ❑ Load TxCMPR with values corresponding to the on-line calculated widths (duty cycles) of PWM pulses

The period value is obtained by dividing the desired PWM period by the period of the GP timer input clock, and subtracting one from the resulting number when the continuous up-counting mode is selected to generate asymmetric PWM waveforms. When the continuous up-/down-counting mode is selected to generate symmetric PWM waveforms, this value is obtained by dividing the desired PWM period by two times the period of the GP timer input clock.

The GP timer can be initialized the same way as in the previous example. During run time, the GP timer compare register is constantly updated with newly determined compare values corresponding to the newly determined duty cycles.

1.5.2 GP Timer Reset

When any RESET event occurs, the following happens:

- ❑ All GP timer register bits, except for the counting direction indication bits in GPTCONA/B, are reset to 0; thus, the operation of all GP timers is disabled. The counting direction indication bits are all set to 1
- ❑ All timer interrupt flags are reset to 0

- All timer interrupt mask bits are reset to 0, except for $\overline{\text{PDPINTx}}$; thus, all GP timer interrupts are masked except for $\overline{\text{PDPINTx}}$
- All GP timer compare outputs are put in the high-impedance state

1.6 Compare Units

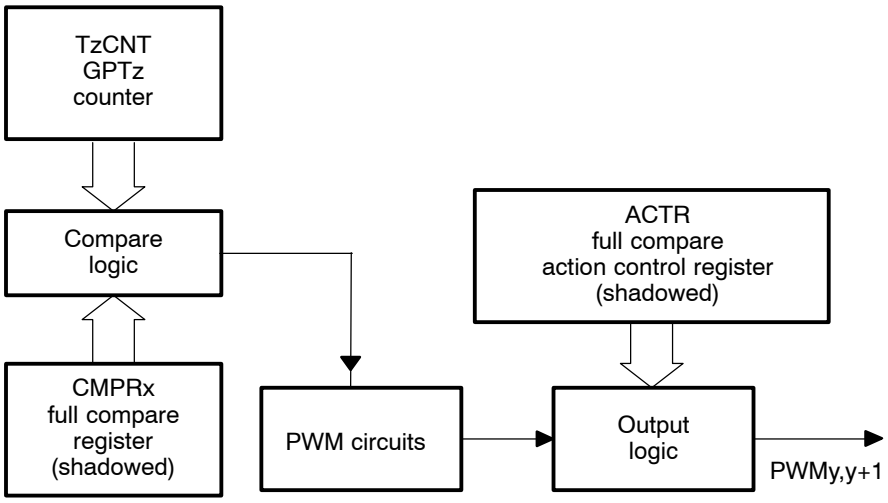
There are three (full) compare units (1, 2, and 3) in the EVA module and three (full) compare units (4, 5, and 6) in the EVB module. Each compare unit has two associated PWM outputs. The time base for the compare units is provided by GP timer 1 (for EVA) and by GP timer 3 (for EVB)

The compare units in each EV module include:

- Three 16-bit compare registers (CMPR1, CMPR2, and CMPR3 for EVA; and CMPR4, CMPR5, and CMPR6 for EVB), all with an associated shadow register, (RW)
- One 16-bit compare control register (COMCONA for EVA, and COMCONB for EVB), (RW)
- One 16-bit action control register (ACTRA for EVA, and ACTRB for EVB), with an associated shadow register, (RW)
- Six PWM (3-state) output (compare output) pins (PWMy, y = 1, 2, 3, 4, 5, 6 for EVA and PWMz, z = 7, 8, 9, 10, 11, 12 for EVB)
- Control and interrupt logic

The functional block diagram of a compare unit is shown in Figure 1–9.

Figure 1–9. Compare Unit Block Diagram
(For EVA: x = 1, 2, 3; y = 1, 3, 5; z = 1.
For EVB: x = 4, 5, 6; y = 7, 9, 11; z = 3.)



The time base for the compare units and the associated PWM circuits is provided by GP timer 1 (for EVA) or GP timer 3 (for EVB), which can be in any of

its counting modes when the compare operation is enabled. Transitions occur on the compare outputs.

Compare Inputs/Outputs

The inputs to a compare unit include:

- Control signals from control registers
- GP timer 1/3 (T1CNT/T3CNT) and its underflow and period match signals
- RESET

The output of a compare unit is a compare match signal. If the compare operation is enabled, this match signal sets the interrupt flag and causes transitions on the two output pins associated with the compare unit.

Compare Operation Modes

The operation mode of the compare units is determined by the bits in COMCONx. These bits determine:

- Whether the compare operation is enabled
- Whether the compare outputs are enabled
- The condition on which the compare registers are updated with the values in their shadow registers
- Whether space vector PWM mode is enabled

Operation

The following paragraph describes the operation of the EVA compare unit. The operation of the EVB compare unit is identical. For EVB, GP timer 3 and ACTRB are used.

The value of the GP timer 1 counter is continuously compared with that of the compare register. When a match is made, a transition appears on the two outputs of the compare unit according to the bits in the action control register (ACTRA). The bits in ACTRA can individually specify each output to be toggle active high or toggle active-low (if not forced high or low) on a compare match. The compare interrupt flag associated with a compare unit is set when a compare match is made between GP timer 1 and the compare register of this compare unit, if compare is enabled. A peripheral interrupt request is generated by the flag if the interrupt is unmasked. The timing of output transitions,

setting of interrupt flags, and generation of interrupt requests are the same as that of the GP timer compare operation. The outputs of the compare units in compare mode are subject to modification by the output logic, dead band units, and the space vector PWM logic.

1.6.1 Register Setup for Compare Unit Operation

The register setup sequence for compare unit operation requires:

For EVA	For EVB
Setting up T1PR	Setting up T3PR
Setting up ACTRA	Setting up ACTRB
Initializing CMPRx	Initializing CMPRx
Setting up COMCONA	Setting up COMCONB
Setting up T1CON	Setting up T3CON

1.6.2 Compare Units Registers

The addresses of registers associated with compare units and associated PWM circuits are shown in Table 1–6, *Addresses of EVA Compare Control Registers* on page 1-36, and in Table 1–7, *Addresses of EVB Compare Control Registers* on page 1-37. These registers are discussed in the subsections that follow.

Compare Control Registers (COMCONA and COMCONB)

The operation of the compare units is controlled by the compare control registers (COMCONA and COMCONB). The bit definition of COMCONA is summarized in Figure 5–7 and that of COMCONB is summarized in Figure 5–8. COMCONA and COMCONB are readable and writable.

Table 1–6. Addresses of EVA Compare Control Registers

Address	Register	Name
7411h	COMCONA	Compare control register
7413h	ACTRA	Compare action control register
7415h	DBTCONA	Dead-band timer control register
7417h	CMPR1	Compare register 1
7418h	CMPR2	Compare register 2
7419h	CMPR3	Compare register 3

Table 1−7. Addresses of EVB Compare Control Registers

Address	Register	Name
7511h	COMCONB	Compare control register
7513h	ACTRB	Compare action control register
7515h	DBTCONB	Dead-band timer control register
7517h	CMPR4	Compare register 4
7518h	CMPR5	Compare register 5
7519h	CMPR6	Compare register 6

1.6.3 Compare Unit Interrupts

There is a maskable interrupt flag in EVxIFRA and EVxIFRB for each compare unit. The interrupt flag of a compare unit is set one clock cycle after a compare match, if a compare operation is enabled. A peripheral interrupt request is generated by the flag if it is unmasked.

1.6.4 Compare Unit Reset

When any reset event occurs, all register bits associated with the compare units are reset to zero and all compare output pins are put in the high-impedance state.

Chapter 2

PWM Circuits

The pulse-width modulation (PWM) circuits associated with compare units make it possible to generate six PWM output channels (per EV) with programmable dead-band and output polarity.

Topic **Page**

2.1 PWM Circuits Associated With Compare Units

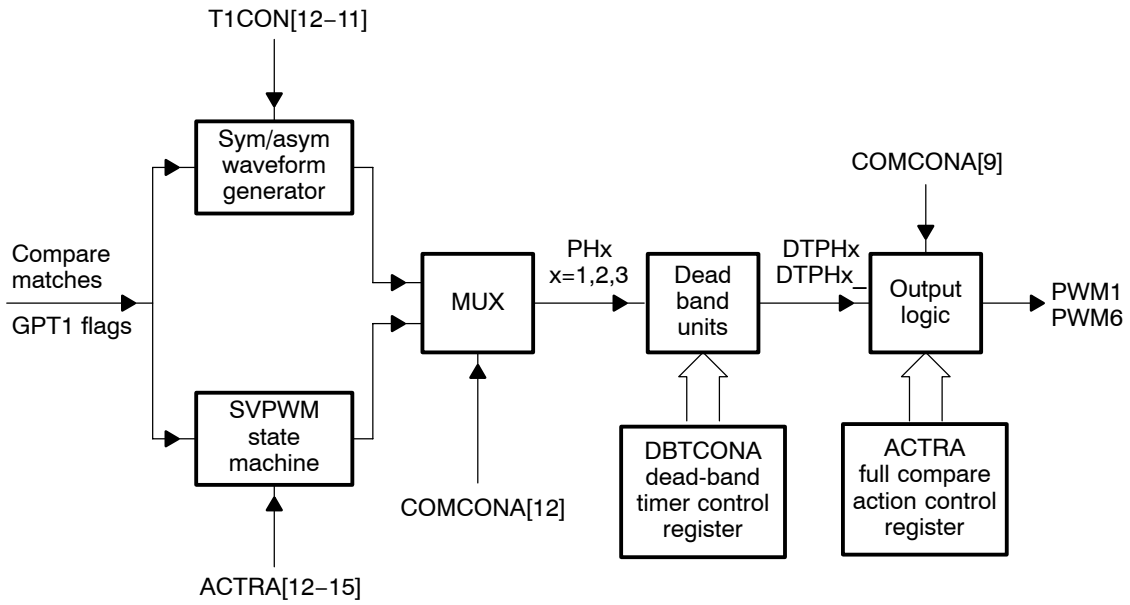
The EVA PWM circuits functional block diagram is shown in Figure 2−1. It includes the following functional units:

- Asymmetric/Symmetric Waveform Generators
- Programmable Dead-Band Unit (DBU)
- Output Logic
- Space Vector (SV) PWM State Machine

The EVB PWM circuits functional block diagram is identical to that of the EVA with the corresponding change of configuration registers.

The asymmetric/symmetric waveform generators are the same as those of the GP timers. The dead-band units and output logic are discussed in sections 2.1.2 and 2.1.5, respectively. The space vector PWM state machine and the space vector PWM technique are described later in this chapter.

Figure 2−1. PWM Circuits Block Diagram



The PWM circuits are designed to minimize CPU overhead and user intervention when generating pulse width modulated waveforms used in motor control and motion control applications. PWM generation with compare units and associated PWM circuits are controlled by the following control registers: T1CON, COMCONA, ACTRA, and DBTCONA (in case of EVA); and T3CON, COMCONB, ACTRB, and DBTCONB (in case of EVB).

2.1.1 PWM Generation Capability of Event Manager

The PWM waveform generation capability of each event manager module (A and B) is summarized as follows:

- ❏ Five independent PWM outputs, three of which are generated by the compare units; the other two are generated by the GP timer compares – plus three additional PWM outputs, dependent on the three compare unit PWM outputs
- ❏ Programmable dead-band for the PWM output pairs associated with the compare units
- ❏ Minimum dead-band duration of one device clock cycle
- ❏ Minimum PWM pulse width and pulse width increment/decrement of one clock cycle
- ❏ 16-bit maximum PWM resolution
- ❏ On-the-fly change of PWM carrier frequency (double buffered period registers)
- ❏ On-the-fly change of PWM pulse widths (double buffered compare registers)
- ❏ Power Drive Protection Interrupt
- ❏ Programmable generation of asymmetric, symmetric, and space vector PWM waveforms
- ❏ Minimum CPU overhead because of the auto-reloading of the compare and period registers

2.1.2 Programmable Dead-Band (Dead-Time) Unit

EVA and EVB have their own programmable dead-band units (DBTCONA and DBTCONB, respectively). The programmable dead-band unit features:

- One 16-bit dead-band control register, DBTCONx (RW)
- One input clock prescaler: x/1, x/2, x/4, etc., to x/32
- Device (CPU) clock input
- Three 4-bit down-counting timers
- Control logic

2.1.3 Dead-Band Timer Control Registers A and B (DBTCONA and DBTCONB)

The operation of the dead-band unit is controlled by the dead-band timer control registers (DBTCONA and DBTCONB). The bit description of DBTCONA is given in Figure 5−15 and that of DBTCONB is given in Figure 5−16.

2.1.4 Inputs and Outputs of Dead-Band Unit

The inputs to the dead-band unit are PH1, PH2, and PH3 from the asymmetric/ symmetric waveform generators of compare units 1, 2, and 3, respectively.

The outputs of the dead-band unit are DTPH1, DTPH1_, DTPH2, DTPH2_, DTPH3, and DTPH3_, corresponding to PH1, PH2, and PH3, respectively.

Dead Band Generation

For each input signal PHx, two output signals, DTPHx and DTPHx_, are generated. When dead-band is not enabled for the compare unit and its associated outputs, the two signals are exactly the same. When the dead-band unit is enabled for the compare unit, the transition edges of the two signals are separated by a time interval called dead-band. This time interval is determined by the DBTCONx bits. If you assume that the value in DBTCONx[11−8] is *m,* and that the value in DBTCONx[4−2] corresponds to prescaler *x/p,* then the dead-band value is (*p***m*) device clock cycles.

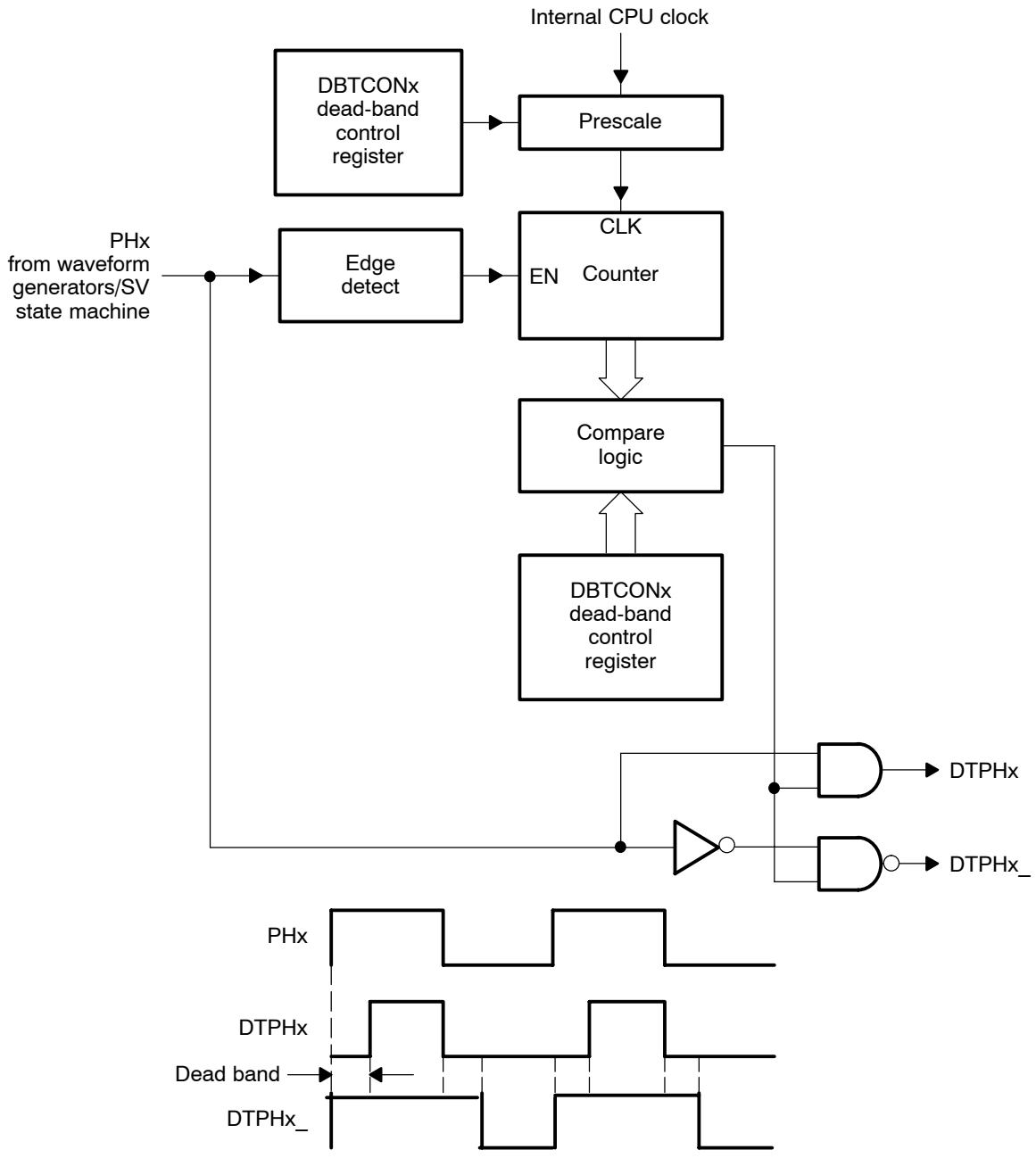
Table 2−1, on page 2-5, shows the dead-band generated by typical bit combinations in DBTCONx. The values are based on a 25-ns HSPCLK. Figure 2−2, on page 2-6, shows the block diagram of the dead-band logic for one compare unit.

Table 2–1. Dead-Band Generation Examples

DBT3–DBT0 (*m*) (DBTCONx[11–8])	DBTPS2–DBTPS0 (*p*) (DBTCONx[4–2])					
	110 and 1x1 (P=32)	100 (P=16)	011 (P=8)	010 (P=4)	001 (P=2)	000 (P=1)
0	0	0	0	0	0	0
1	0.8	0.4	0.2	0.1	0.05	0.025
2	1.6	0.8	0.4	0.2	0.1	0.05
3	2.4	1.2	0.6	0.3	0.15	0.075
4	3.2	1.6	0.8	0.4	0.2	0.1
5	4	2	1	0.5	0.25	0.125
6	4.8	2.4	1.2	0.6	0.3	0.15
7	5.6	2.8	1.4	0.7	0.35	0.175
8	6.4	3.2	1.6	0.8	0.4	0.2
9	7.2	3.6	1.8	0.9	0.45	0.225
A	8	4	2	1	0.5	0.25
B	8.8	4.4	2.2	1.1	0.55	0.275
C	9.6	4.8	2.4	1.2	0.6	0.3
D	10.4	5.2	2.6	1.3	0.65	0.325
E	11.2	5.6	2.8	1.4	0.7	0.35
F	12	6	3	1.5	0.75	0.375

Note: Table values are given in μs.

Figure 2–2. Dead-Band Unit Block Diagram (x = 1, 2, or 3)



Note: Signals such as PHx, DTPHx, and $\overline{\text{DTPHx}}$ are internal to the device, and as such, external monitoring/control of these signals is not possible.

Other Important Features of Dead-Band Units

The dead-band unit is designed to prevent an overlap under any operating situation between the turn-on period of the upper and lower devices controlled by the two PWM outputs associated with each compare unit. This includes those situations where you have loaded a dead-band value greater than that of the duty cycle, and when the duty cycle is 100% or 0%. As a result, the PWM outputs associated with a compare unit do not reset to an inactive state at the end of a period when dead band is enabled for the compare unit.

2.1.5 Output Logic

The output logic circuit determines the polarity and/or the action that must be taken on a compare match for outputs PWMx, for x = 1–6. The outputs associated with each compare unit can be specified active low, active high, forced low, or forced high. The polarity and/or the action of the PWM outputs can be programmed by proper configuration of bits in the ACTR register. The PWM output pins can all be put in the high-impedance state by any of the following:

- Software clearing the COMCONx[9] bit
- Hardware pulling $\overline{\text{PDPINTx}}$ low when $\overline{\text{PDPINTx}}$ is unmasked
- The occurrence of any reset event

Active $\overline{\text{PDPINTx}}$ (when enabled) and system reset override the bits in COMCONx and ACTRx

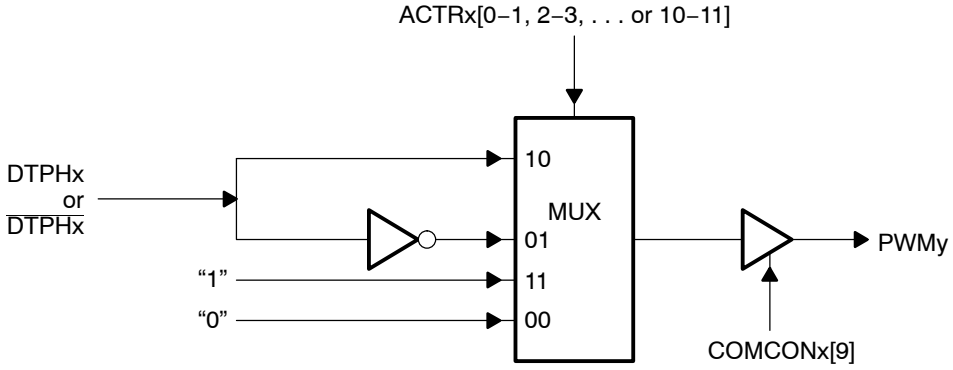
Figure 2–3, on page 2-8, shows a block diagram of the output logic circuit (OLC). The inputs of output logic for the compare units are:

- DTPH1, $\overline{\text{DTPH1}}$, DTPH2, $\overline{\text{DTPH2}}$, DTPH3, and $\overline{\text{DTPH3}}$ from the dead-band unit and compare match signals
- The control bits of ACTRx
- $\overline{\text{PDPINTx}}$ and RESET

The outputs of the Output Logic for the compare units are:

- PWMx, x = 1–6 (for EVA)
- PWMy, y = 7–12 (for EVB)

Figure 2–3. Output Logic Block Diagram (x = 1, 2, or 3; y = 1, 2, 3, 4, 5, or 6)



Output logic for PWM mode

2.2 PWM Waveform Generation

A PWM signal is a sequence of pulses with changing pulse widths. The pulses are spread over a number of fixed-length periods so that there is one pulse in each period. The fixed period is called the PWM (carrier) period and its inverse is called the PWM (carrier) frequency. The widths of the PWM pulses are determined, or modulated, from pulse to pulse according to another sequence of desired values, the modulating signal.

In a motor control system, PWM signals are used to control the on and off time of switching power devices that deliver the desired current and energy to the motor windings (see Figure 2–6 on page 2-14). The shape and frequency of the phase currents and the amount of energy delivered to the motor windings control the required speed and torque of the motor. In this case, the command voltage or current to be applied to the motor is the modulating signal. The frequency of the modulating signal is typically much lower than the PWM carrier frequency.

2.2.1 PWM Signal Generation

To generate a PWM signal, an appropriate timer is needed to repeat a counting period that is the same as the PWM period. A compare register is used to hold the modulating values. The value of the compare register is constantly compared with the value of the timer counter. When the values match, a transition (from low to high, or high to low) happens on the associated output. When a second match is made between the values, or when the end of a timer period is reached, another transition (from high to low, or low to high) happens on the associated output. In this way, an output pulse is generated whose on (or off) duration is proportional to the value in the compare register. This process is repeated for each timer period with different (modulating) values in the compare register. As a result, a PWM signal is generated at the associated output.

Dead Band

In many motion/motor and power electronics applications, two power devices, an upper and a lower, are placed in series on one power converter leg. The turn-on periods of the two devices must not overlap with each other in order to avoid a shoot-through fault. Thus, a pair of non-overlapping PWM outputs is often required to properly turn on and off the two devices. A dead time (deadband) is often inserted between the turning-off of one transistor and the turning-on of the other transistor. This delay allows complete turning-off of one transistor before the turning-on of the other transistor. The required time delay is specified by the turning-on and turning-off characteristics of the power transistors and the load characteristics in a specific application.

2.2.2 Generation of PWM Outputs With Event Manager

Each of the three compare units, together with GP timer 1 (in the case of EVA) or GP timer 3 (in the case of EVB), the dead-band unit, and the output logic in the event manager module, can be used to generate a pair of PWM outputs with programmable dead-band and output polarity on two dedicated device pins. There are six such dedicated PWM output pins associated with the three compare units in each EV module. These six dedicated output pins can be used to conveniently control 3-phase ac induction or brushless dc motors. The flexibility of output behavior control by the compare action control register (ACTRx) also makes it easy to control switched reluctance and synchronous reluctance motors in a wide range of applications. The PWM circuits can also be used to conveniently control other types of motors such as dc brush and stepper motors in single or multi-axis control applications. Each GP timer compare unit, if desired, can also generate a PWM output based on its own timer.

2.2.3 Asymmetric and Symmetric PWM Generation

Both asymmetric and symmetric PWM waveforms can be generated by every compare unit on the EV module. In addition, the three compare units together can be used to generate 3-phase symmetric space vector PWM outputs. PWM generation with GP timer compare units has been described in the GP timer sections. Generation of PWM outputs with the compare units is discussed in this section.

2.2.4 Register Setup for PWM Generation

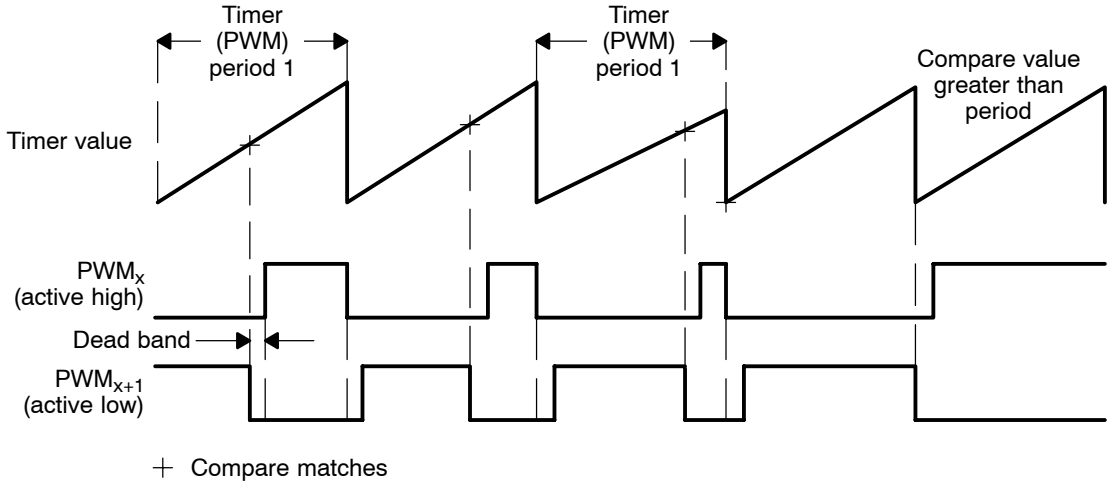
All three kinds of PWM waveform generations with compare units and associated circuits require configuration of the same Event Manager registers. The setup process for PWM generation includes the following steps:

- ❑ Setup and load ACTRx
- ❑ Setup and load DBTCONx, if dead-band is to be used
- ❑ Initialize CMPRx
- ❑ Setup and load COMCONx
- ❑ Setup and load T1CON (for EVA) or T3CON (for EVB) to start the operation
- ❑ Rewrite CMPRx with newly determined values

2.2.5 Asymmetric PWM Waveform Generation

The edge-triggered or asymmetric PWM signal is characterized by modulated pulses which are not centered with respect to the PWM period, as shown in Figure 2–4. The width of each pulse can only be changed from one side of the pulse.

Figure 2–4. Asymmetric PWM Waveform Generation With Compare Unit and PWM Circuits (x = 1, 3, or 5)



To generate an asymmetric PWM signal, GP timer 1 is put in the continuous up-counting mode and its period register is loaded with a value corresponding to the desired PWM carrier period. The COMCONx is configured to enable the compare operation, set the selected output pins to be PWM outputs, and enable the outputs. If dead-band is enabled, the value corresponding to the required dead-band time should be written by software into the DBT(3:0) bits in DBTCONx(11:8). This is the period for the 4-bit dead-band timers. One dead-band value is used for all PWM output channels.

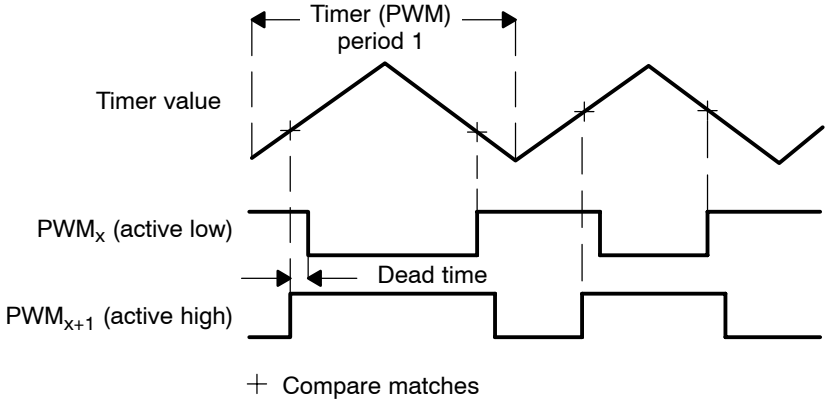
By proper configuration of ACTRx with software, a normal PWM signal can be generated on one output associated with a compare unit while the other is held low (or off) or high (or on), at the beginning, middle, or end of a PWM period. Such software controlled flexibility of PWM outputs is particularly useful in switched reluctance motor control applications.

After GP timer 1 (or GP timer 3) is started, the compare registers are rewritten every PWM period with newly determined compare values to adjust the width (the duty cycle) of PWM outputs that control the switch-on and -off duration of the power devices. Since the compare registers are shadowed, a new value can be written to them at any time during a period. For the same reason, new values can be written to the action and period registers at any time during a period to change the PWM period or to force changes in the PWM output definition.

2.2.6 Symmetric PWM Waveform Generation

A centered or symmetric PWM signal is characterized by modulated pulses which are centered with respect to each PWM period. The advantage of a symmetric PWM signal over an asymmetric PWM signal is that it has two inactive zones of the same duration: at the beginning and at the end of each PWM period. This symmetry has been shown to cause less harmonics than an asymmetric PWM signal in the phase currents of an ac motor, such as induction and dc brushless motors, when sinusoidal modulation is used. Figure 2–5 shows two examples of symmetric PWM waveforms.

Figure 2–5. Symmetric PWM Waveform Generation With Compare Units and PWM Circuits (x = 1, 3, or 5)



The generation of a symmetric PWM waveform with a compare unit is similar to the generation of an asymmetric PWM waveform. The only exception is that GP timer 1 (or GP timer 3) now needs to be put in continuous up-/down-counting mode.

There are usually two compare matches in a PWM period in symmetric PWM waveform generation, one during the upward counting before period match, and another during downward counting after period match. A new compare value becomes effective after the period match (reload on period) because it makes it possible to advance or delay the second edge of a PWM pulse. An application of this feature is when a PWM waveform modification compensates for current errors caused by the dead-band in ac motor control.

Because the compare registers are shadowed, a new value can be written to them at any time during a period. For the same reason, new values can be written to the action and period registers at any time during a period to change the PWM period or to force changes in the PWM output definition.

2.2.7 Double Update PWM Mode

The 28x Event Manager supports “Double Update PWM Mode.” This mode refers to a PWM operation mode in which the position of the leading edge and the position of the trailing edge of a PWM pulse are independently modifiable in each PWM period. To support this mode, the compare register that determines the position of the edges of a PWM pulse must allow (buffered) compare value update once at the beginning of a PWM period and another time in the middle of a PWM period.

The compare registers in 28x Event Managers are all buffered and support three compare value reload/update (value in buffer becoming active) modes. These modes have earlier been documented as compare value reload conditions. The reload condition that supports double update PWM mode is reloaded on Underflow (beginning of PWM period) OR Period (middle of PWM period). Double update PWM mode can be achieved by using this condition for compare value reload.

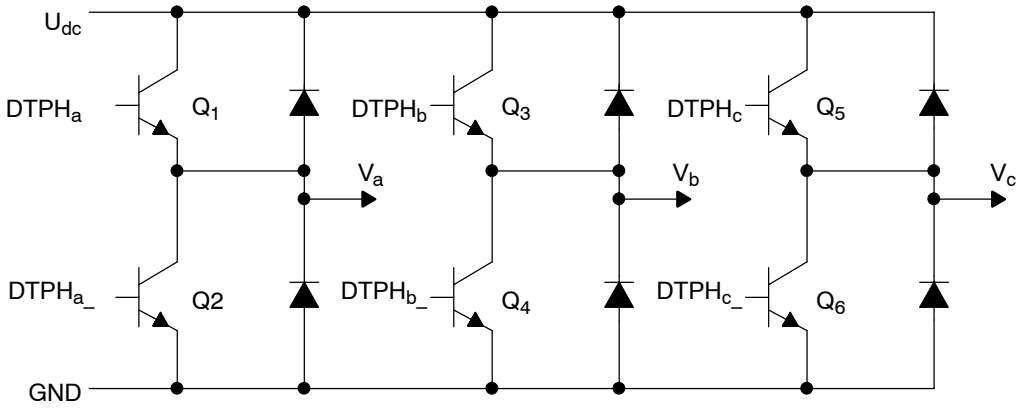
2.3 Space Vector PWM

Space vector PWM refers to a special switching scheme of the six power transistors of a 3-phase power converter. It generates minimum harmonic distortion to the currents in the windings of a 3-phase ac motor. It also provides more efficient use of supply voltage in comparison with the sinusoidal modulation method.

2.3.1 3-Phase Power Inverter

The structure of a typical 3-phase power inverter is shown in Figure 2–6, where V_a, V_b, and V_c are the voltages applied to the motor windings. The six power transistors are controlled by $DTPH_x$ and $DTPH_{x\_}$ (x = a, b, and c). When an upper transistor is switched on ($DTPH_x$ = 1), the lower transistor is switched off ($DTPH_{x\_}$ = 0). Thus, the on and off states of the upper transistors (Q1, Q3, and Q5) or, equivalently, the state of DTPHx (x = a, b, and c) are sufficient to evaluate the applied motor voltage U_{out}.

Figure 2–6. 3-Phase Power Inverter Schematic Diagram



Power Inverter Switching Patterns and the Basic Space Vectors

When an upper transistor of a leg is on, the voltage V_x (x = a, b, or c) applied by the leg to the corresponding motor winding is equal to the voltage supply U_{dc}. When it is off, the voltage applied is zero. The on and off switching of the upper transistors ($DTPH_x$, x = a, b, or c) have eight possible combinations. The eight combinations and the derived motor line-to-line and phase voltage in terms of dc supply voltage U_{dc} are shown in Table 2–2, on page 2-15, where a, b, and c represent the values of $DTPH_a$, $DTPH_b$, and $DTPH_c$, respectively.

Table 2–2. Switching Patterns of a 3-Phase Power Inverter

a	b	c	$V_{a0}(U_{dc})$	$V_{b0}(U_{dc})$	$V_{c0}(U_{dc})$	$V_{ab}(U_{dc})$	$V_{bc}(U_{dc})$	$V_{ca}(U_{dc})$
0	0	0	0	0	0	0	0	0
0	0	1	–1/3	–1/3	2/3	0	–1	1
0	1	0	–1/3	2/3	–1/3	–1	1	0
0	1	1	–2/3	1/3	1/3	–1	0	1
1	0	0	2/3	–1/3	–1/3	1	0	–1
1	0	1	1/3	–2/3	1/3	1	–1	0
1	1	0	1/3	1/3	–2/3	0	1	–1
1	1	1	0	0	0	0	0	0

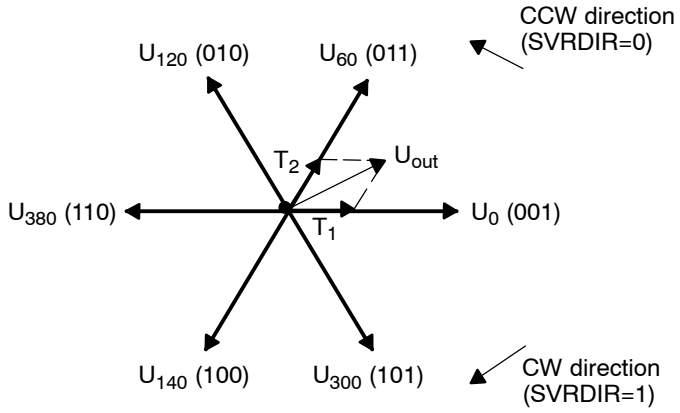
Note: 0 = off, 1 = on

Mapping the phase voltages corresponding to the eight combinations onto the d-q plane by performing a d-q transformation (which is equivalent to an orthogonal projection of the 3-vectors (a b c) onto the two dimensional plane perpendicular to the vector (1,1,1), the d-q plane), results in six nonzero vectors and two zero vectors. The nonzero vectors form the axes of a hexagonal. The angle between two adjacent vectors is 60 degrees. The two zero vectors are at the origin. These eight vectors are called the basic space vectors and are denoted by U_0, U_{60}, U_{120}, U_{180}, U_{240}, U_{300}, O_{000}, and O_{111}. The same transformation can be applied to the demanded voltage vector U_{out} to be applied to a motor. Figure 2–7 shows the projected vectors and the projected desired motor voltage vector U_{out}.

The d axis and q axis of a d-q plane correspond here to the horizontal and vertical geometrical axes of the stator of an ac machine.

The objective of the space vector PWM method is to approximate the motor voltage vector U_{out} by a combination of these eight switching patterns of the six power transistors.

Figure 2−7. Basic Space Vectors and Switching Patterns



The binary representations of two adjacent basic vectors are different in only one bit; that is, only one of the upper transistors switches when the switching pattern switches from U_x to U_{x+60} or from U_{x+60} to U_x. Also, the zero vectors O_{000} and O_{111} apply no voltage to the motor.

2.3.2 Approximation of Motor Voltage With Basic Space Vectors

The projected motor voltage vector U_{out}, at any given time, falls into one of the six sectors. Thus, for any PWM period, it can be approximated by the vector sum of two vector components lying on the two adjacent basic vectors:

$$U_{out} = T_1 U_x + T_2 U_{x+60} + T_0 (O_{000} \text{ or } O_{111})$$

where T_0 is given by $T_p - T_1 - T_2$ and T_p is the PWM carrier period. The third term on the right side of the equation does not affect the vector sum U_{out}. The generation of U_{out} is beyond the scope of this context. For more details on space vector PWM and motor control theory, see *The Field Orientation Principle in Control of Induction Motors* by Andrzej M. Trzynadlowski (The Kluwer International Series in Engineering and Computer Science, Vol. 258:Power).

The above approximation means that the upper transistors must have the on and off pattern corresponding to U_x and U_{x+60} for the time duration of T_1 and T_2, respectively, in order to apply voltage U_{out} to the motor. The inclusion of zero basic vectors helps to balance the turn on and off periods of the transistors, and thus their power dissipation.

2.3.3 Space Vector PWM Waveform Generation With Event Manager

The EV module has built-in hardware to greatly simplify the generation of symmetric space vector PWM waveforms. Software is used to generate space vector PWM outputs.

2.3.4 Software

To generate space vector PWM outputs, the user software must:

- ❑ Configure ACTRx to define the polarity of the compare output pins
- ❑ Configure COMCONx to enable compare operation and space vector PWM mode, and set the reload condition for CMPRx to be underflow
- ❑ Put GP timer 1 (or GP timer 3) in continuous up-/down-counting mode to start the operation

The user software then needs to determine the voltage U_{out} to be applied to the motor phases in the two dimensional d-q plane, decompose U_{out}, and perform the following for each PWM period:

- ❑ Determine the two adjacent vectors, U_x and U_{x+60}
- ❑ Determine the parameters T_1, T_2, and T_0
- ❑ Write the switching pattern corresponding to U_x in ACTRx[14–12] and 1 in ACTRx[15], or the switching pattern of U_{x+60} in ACTRx[14–12] and 0 in ACTRx[15]
- ❑ Put (1/2 T1) in CMPR1 and (1/2 T1 + 1/2 T2) in CMPR2

2.3.5 Space Vector PWM Hardware

The space vector PWM hardware in the EV module does the following to complete a space vector PWM period:

- ❑ At the beginning of each period, sets the PWM outputs to the (new) pattern U_y defined by ACTRx[14–12]
- ❑ On the first compare match during up-counting between CMPR1 and GP timer 1 at (1/2 T1), switches the PWM outputs to the pattern of U_{y+60} if ACTRx[15] is 1, or to the pattern of U_y if ACTRx[15] is 0 ($U_{0-60} = U_{300}$, $U_{360+60} = U_{60}$)
- ❑ On the second compare match during up-counting between CMPR2 and GP timer 1 at (1/2 T1 + 1/2 T2), switches the PWM outputs to the pattern (000) or (111), whichever differs from the second pattern by one bit
- ❑ On the first compare match during down-counting between CMPR2 and GP timer 1 at (1/2 T1 + 1/2 T2), switches the PWM outputs back to the second output pattern
- ❑ On the second compare match during down-counting between CMPR1 and GP timer 1 at (1/2 T1), switches the PWM outputs back to the first pattern

2.3.6 Space Vector PWM Waveforms

The space vector PWM waveforms generated are symmetric with respect to the middle of each PWM period; and for this reason, it is called the symmetric space vector PWM generation method. Figure 2–8 shows examples of the symmetric space vector PWM waveforms.

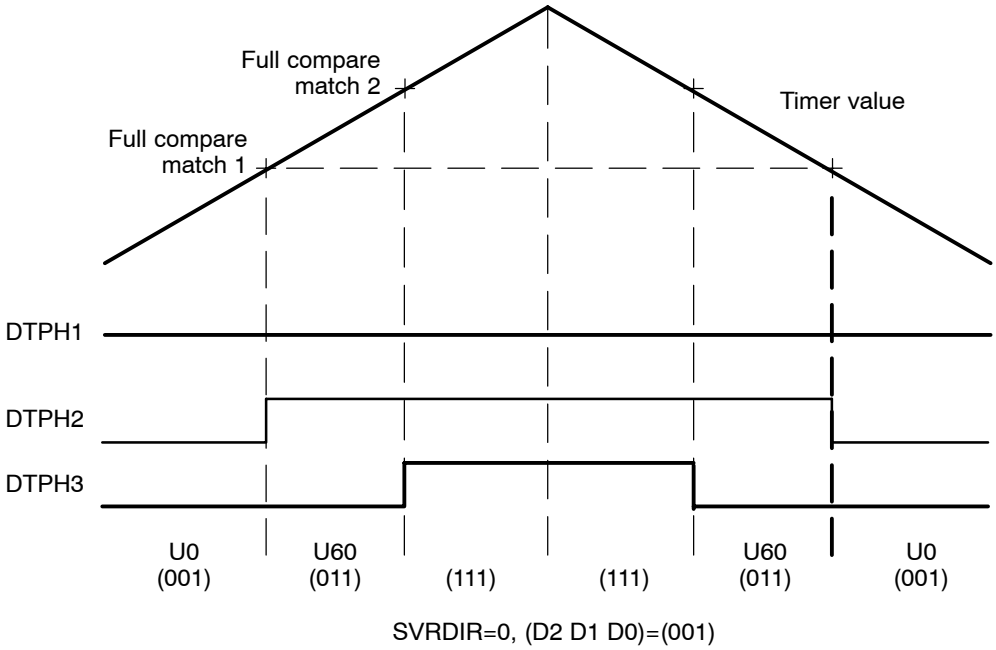
2.3.7 The Unused Compare Register

Only two compare registers are used in space vector PWM output generation. The third compare register, however, is still constantly compared with GP timer 1. When a compare match happens, the corresponding compare interrupt flag remains set and a peripheral interrupt request is generated, if the flag is unmasked. Therefore, the compare register that is not used in space vector PWM output generation can still be used to time events happening in a specific application. Also, because of the extra delay introduced by the state machine, the compare output transitions are delayed by one clock cycle in space vector PWM mode.

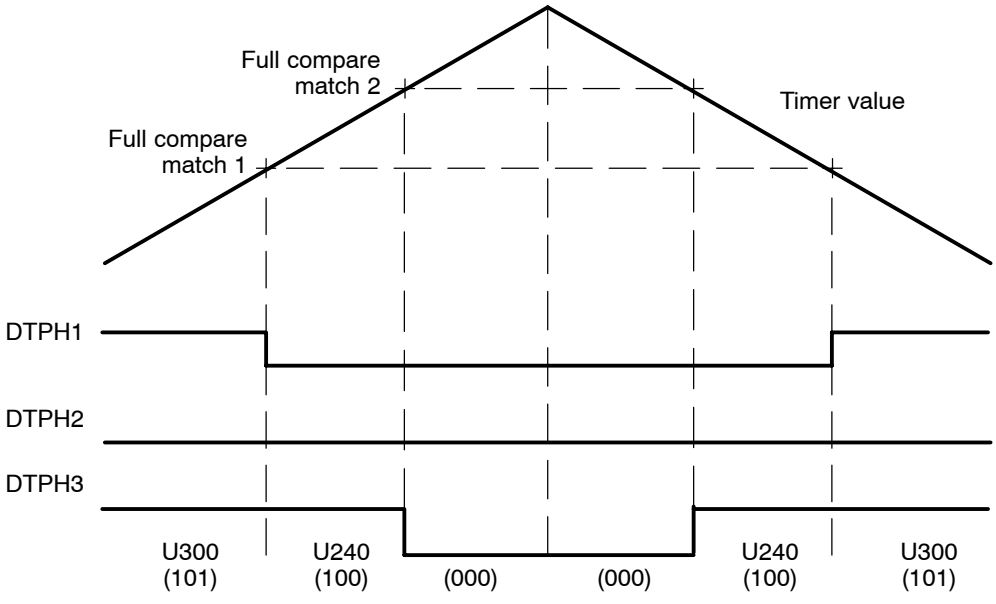
2.3.8 Space Vector PWM Boundary Conditions

All three compare outputs become inactive when both compare registers (CMPR1 and CMPR2) are loaded with a zero value in space vector PWM mode. It is the user's responsibility to assure that (CMPR1) $\leq$ (CMPR2) $\leq$ (T1PR) in the space vector PWM mode; otherwise, unpredictable behavior may result.

Figure 2–8. Symmetric Space Vector PWM Waveforms



SVRDIR=0, (D2 D1 D0)=(001)



SVRDIR=1, (D2 D1 D0)=(101)

Chapter 3

Capture Units

Capture units enable logging of transitions on capture input pins. There are six capture units, three in each EV module. Capture Units 1, 2, and 3 are associated with EVA and Capture Units 4, 5, and 6 are associated with EVB. Each capture unit is associated with a capture input pin.

Topic **Page**

3.1 Capture Unit Overview

Each EVA capture unit can choose GP timer 2 or 1 as its time base; however, CAP1 and CAP2 cannot choose a different timer between themselves as their timebase. Each EVB capture unit can choose GP timer 4 or 3 as its time base; however, CAP4 and CAP5 cannot choose a different timer between themselves as their timebase.

The value of the GP timer is captured and stored in the corresponding 2-level-deep FIFO stack when a specified transition is detected on a capture input pin (CAPx). Figure 3–1 shows a block diagram of an EVA capture unit and Figure 3–2 shows a block diagram of an EVB capture unit.

3.1.1 Capture Unit Features

Capture units have the following features:

- One 16-bit capture control register (CAPCONA for EVA, CAPCONB for EVB), (RW)
- One 16-bit capture FIFO status register (CAPFIFOA for EVA, CAPFIFOB for EVB)
- Selection of GP timer 1 or 2 (for EVA) and GP timer 3 or 4 (for EVB) as the time base
- Three 16-bit 2-level-deep FIFO stacks, one for each capture unit
- Six Schmitt-triggered capture input pins, CAP1 through CAP6, one input pin for each capture unit. (All inputs are synchronized with the device/CPU clock: in order for a transition to be captured, the input must hold at its current level to meet the two rising edges of the device clock. If the input qualifier circuit is used, then the pulse width requirement warranted by the qualification circuitry must be met as well. Input pins CAP1 and CAP2 (CAP4 and CAP5 in case of EVB) can also be used as QEP inputs to QEP circuit).
- User-specified transition detection (rising edge, falling edge, or both edges)
- Six maskable interrupt flags, one for each capture unit

Figure 3–1. Capture Units Block Diagram (EVA)

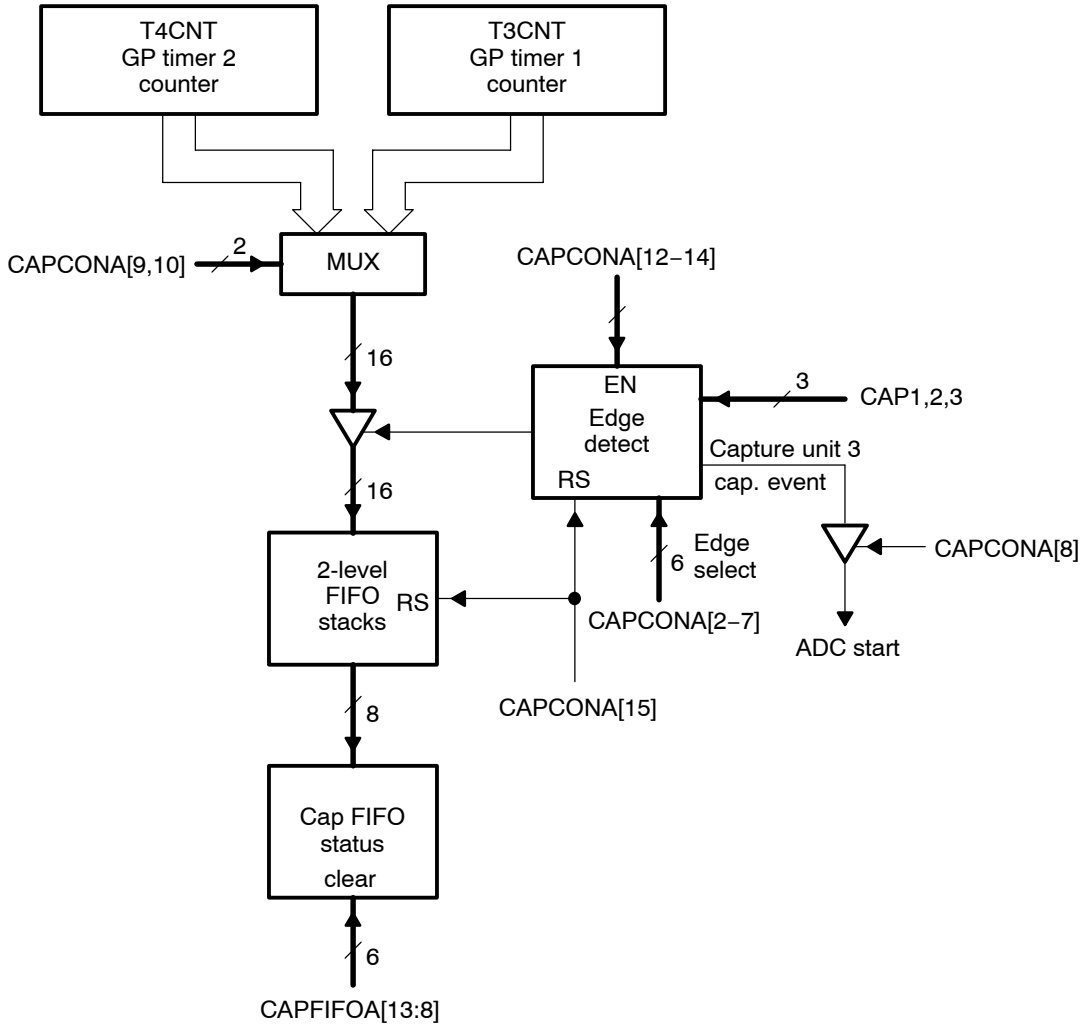
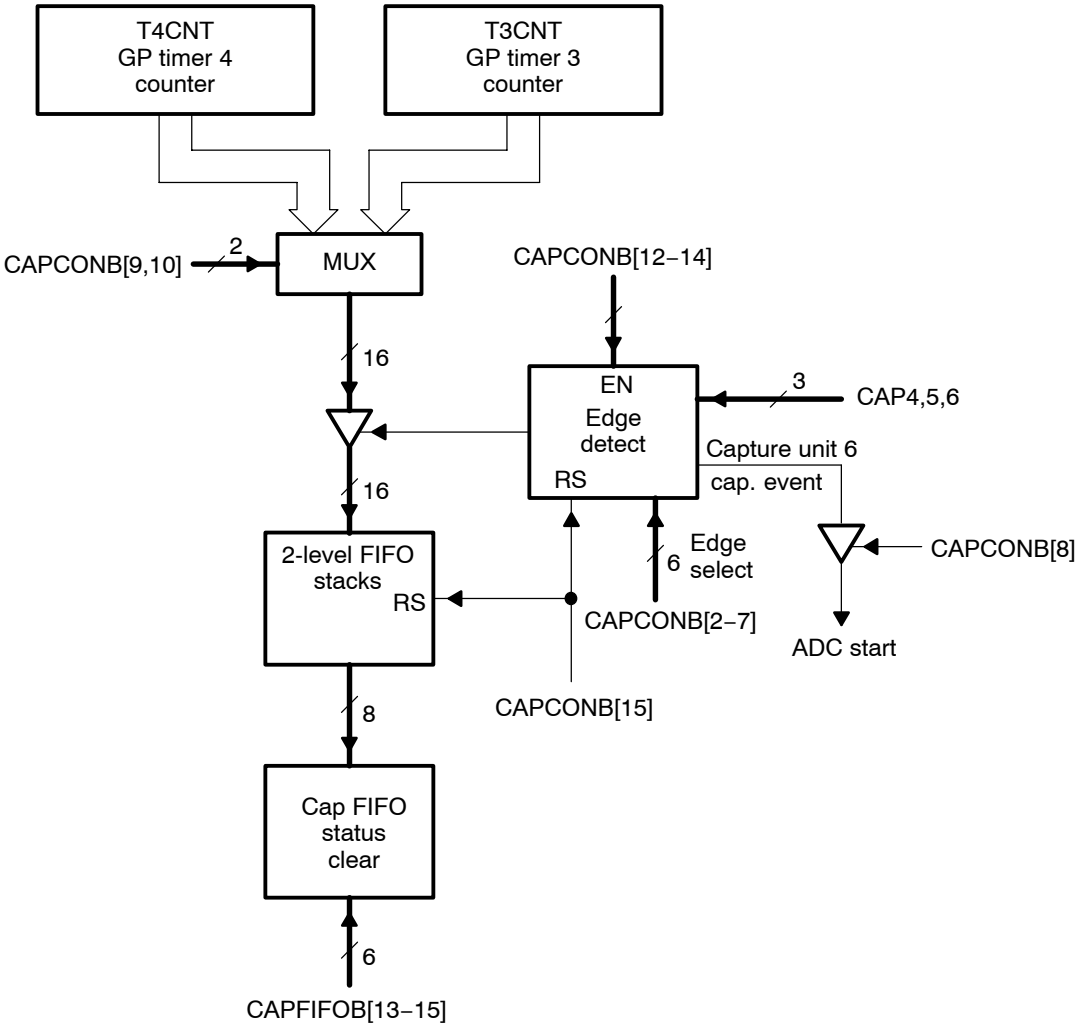


Figure 3–2. Capture Units Block Diagram (EVB)



3.2 Operation of Capture Units

After a capture unit is enabled, a specified transition on the associated input pin causes the counter value of the selected GP timer to be loaded into the corresponding FIFO stack. At the same time, if there are already one or more valid capture values stored in the FIFO stack (CAPxFIFO bits not equal to zero), the corresponding interrupt flag is set. If the flag is unmasked, a peripheral interrupt request is generated. The corresponding status bits in CAPFIFOx are adjusted to reflect the new status of the FIFO stack each time a new counter value is captured in a FIFO stack. The latency from the time a transition happens in a capture input to the time the counter value of the selected GP timer is locked is two clock cycles. This does not include any additional latency due to the input qualifier circuitry.

All capture unit registers are cleared to zero by a RESET condition.

3.2.1 Capture Unit Time Base Selection

For EVA, Capture Unit 3 has a separate time base selection bit from Capture Units 1 and 2. This allows the two GP timers to be used at the same time, one for Capture Units 1 and 2, and the other for Capture Unit 3. For EVB, Capture Unit 6 has a separate time-base selection bit.

Capture operation does not affect the operation of any GP timer or the compare/PWM operations associated with any GP timer.

3.2.2 Capture Unit Setup

For a capture unit to function properly, the following register setup must be performed:

1) Initialize the CAPFIFOx and clear the appropriate status bits.

2) Set the selected GP timer in one of its operating modes.

3) Set the associated GP timer compare register or GP timer period register, if necessary.

4) Set up CAPCONA or CAPCONB as appropriate.

3.3 Capture Unit FIFO Stacks

Each capture unit has a dedicated 2-level-deep FIFO stack. The top stack consists of CAP1FIFO, CAP2FIFO, and CAP3FIFO (in the case of EVA) or CAP4FIFO, CAP5FIFO, and CAP6FIFO (in the case of EVB). The bottom stack consists of CAP1FBOT, CAP2FBOT, and CAP3FBOT (in the case of EVA) or CAP4FBOT, CAP5FBOT, and CAP6FBOT (in the case of EVB). The top-level register of any of the FIFO stacks is a read-only register that always contains the oldest counter value captured by the corresponding capture unit. Therefore, a read access to the FIFO stack of a capture unit always returns the oldest counter value stored in the stack. When the oldest counter value in the top register of the FIFO stack is read, the newer counter value in the bottom register of the stack, if any, is pushed into the top register.

If desired, the bottom register of the FIFO stack can be read. Reading the bottom register of the FIFO stack causes the FIFO status bits to change to 01 (has one entry) if they were previously 10 or 11. If the FIFO status bits were previously 01 when the bottom FIFO register is read, they will change to 00 (empty).

3.3.1 First Capture

The counter value of the selected GP timer (captured by a capture unit when a specified transition happens on its input pin) is written into the top register of the FIFO stack, if the stack is empty. At the same time, the corresponding status bits are set to 01. The status bits are reset to 00 if a read access is made to the FIFO stack before another capture is made.

3.3.2 Second Capture

If another capture occurs before the previously captured counter value is read, the newly captured counter value goes to the bottom register. In the meantime, the corresponding status bits are set to 10. When the FIFO stack is read before another capture happens, the older counter value in the top register is read out, the newer counter value in the bottom register is pushed up into the top register, and the corresponding status bits are set to 01.

The appropriate capture interrupt flag is set by the second capture. A peripheral interrupt request is generated if the interrupt is not masked.

3.3.3 Third Capture

If a capture happens when there are already two counter values captured in the FIFO stack, the oldest counter value in the top register of the stack is pushed out and lost, the counter value in the bottom register of the stack is pushed up into the top register, the newly captured counter value is written into the bottom register, and the status bits are set to 11 to indicate that one or more older captured counter values have been lost.

The appropriate capture interrupt flag is also set by the third capture. A peripheral interrupt request is generated if the interrupt is not masked.

3.4 Capture Interrupt

When a capture is made by a capture unit and there is already at least one valid value in the FIFO (indicated by CAPxFIFO bits not equal to zero), the corresponding interrupt flag is set, and if unmasked, a peripheral interrupt request is generated. Thus, a pair of captured counter values can be read by an interrupt service routine if the interrupt is used. If an interrupt is not desired, either the interrupt flag or the status bits can be polled to determine if two captures have occurred allowing the captured counter values to be read.

3.5 Quadrature Encoder Pulse (QEP) Circuit

Each Event Manager module has a quadrature encoder pulse (QEP) circuit. The QEP circuit, when enabled, decodes and counts the quadrature encoded input pulses on pins CAP1/QEP1 and CAP2/QEP2 (in case of EVA) or CAP4/QEP3 and CAP5/QEP4 (in case of EVB). The QEP circuit can be used to interface with an optical encoder to get position and speed information from a rotating machine. When the QEP circuit is enabled, the capture function on CAP1/CAP2 and CAP4/CAP5 pins is disabled.

3.5.1 QEP Pins

The three QEP input pins are shared between capture units 1, 2, and 3 (or 3, 4, and 5, for EVB), and the QEP circuit.

3.5.2 QEP Circuit Time Base

The time base for the QEP circuit is provided by GP timer 2 (GP timer 4, in case of EVB). The GP timer must be put in directional-up/down count mode with the QEP circuit as the clock source. Figure 3–3 shows the block diagram of the QEP circuit for EVA and Figure 3–4 shows the block diagram of the QEP circuit for EVB.

Figure 3–3. Quadrature Encoder Pulse (QEP) Circuit Block Diagram for EVA

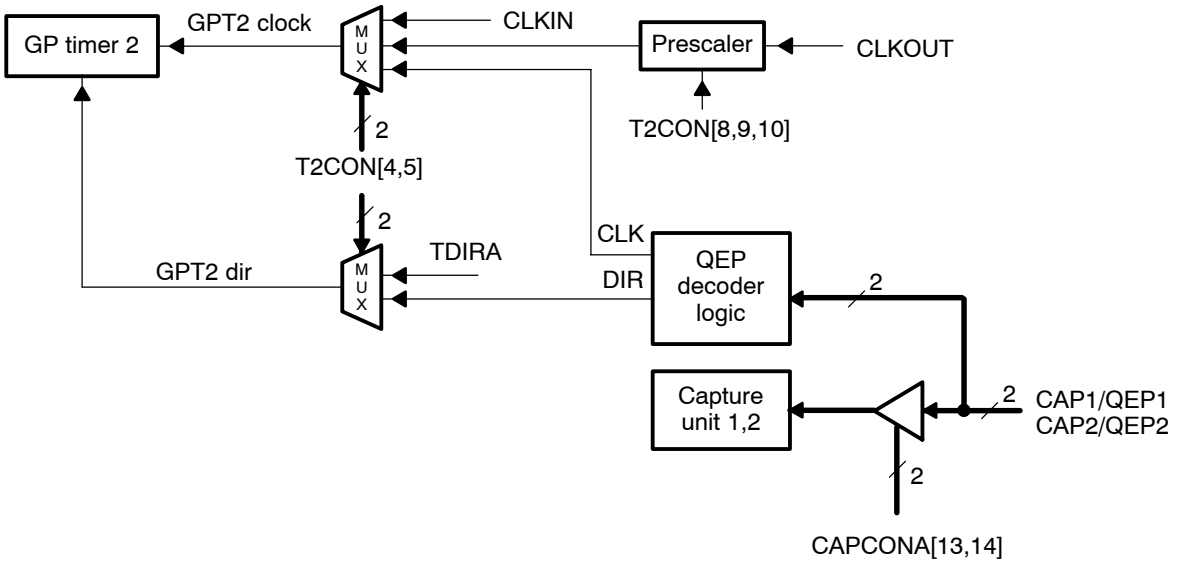
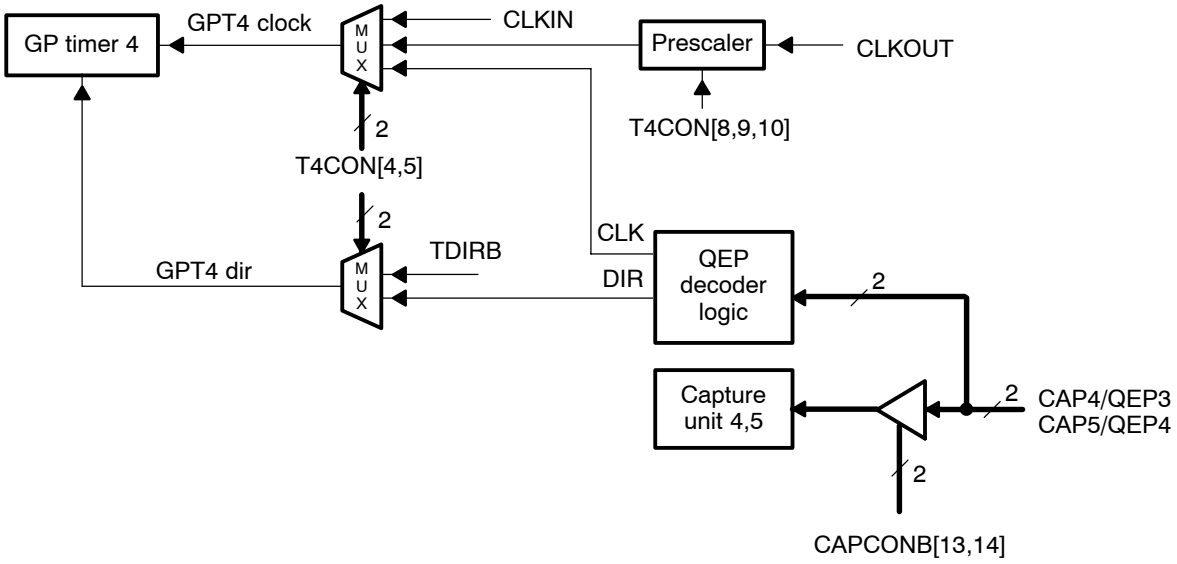


Figure 3–4. Quadrature Encoder Pulse (QEP) Circuit Block Diagram for EVB



3.5.3 Decoding

Quadrature encoded pulses are two sequences of pulses with a variable frequency and a fixed phase shift of a quarter of a period (90 degrees). When generated by an optical encoder on a motor shaft, the direction of rotation of the motor can be determined by detecting which of the two sequences is the leading sequence. The angular position and speed can be determined by the pulse count and pulse frequency.

QEP Circuit

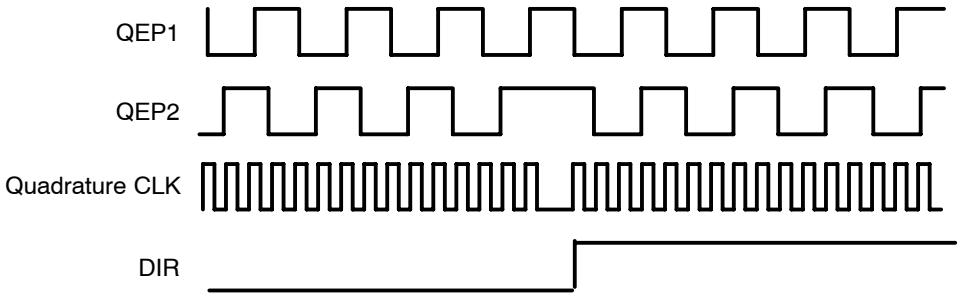
The direction detection logic of the QEP circuit in the EV module determines which one of the sequences is the leading sequence. It then generates a direction signal as the direction input to GP timer 2 (or 4). The timer counts up if CAP1/QEP1 (CAP4/QEP3 for EVB) input is the leading sequence, and counts down if CAP2/QEP2 (CAP5/QEP4 for EVB) is the leading sequence.

Both edges of the pulses of the two quadrature encoded inputs are counted by the QEP circuit. Therefore, the frequency of the clock generated by the QEP logic to GP timer 2 (or 4) is four times that of each input sequence. This quadrature clock is connected to the clock input of GP timer 2 (or 4).

Quadrature Encoded Pulse Decoding Example

Figure 3–5 shows an example of quadrature encoded pulses and the derived clock and counting direction.

Figure 3–5. Quadrature Encoded Pulses and Decoded Timer Clock and Direction



3.5.4 QEP Counting

GP timer 2 (or 4) always starts counting from its current value. A desired value can be loaded to the GP timer's counter prior to enabling the QEP mode. When the QEP circuit is selected as the clock source, the timer ignores the TDIRA/B and TCLKINA/B input pins.

GP Timer Interrupt and Associated Compare Outputs in QEP Operation

Period, underflow, overflow, and compare interrupt flags for a GP timer with a QEP circuit clock are generated on respective matches. A peripheral interrupt request can be generated by an interrupt flag, if the interrupt is unmasked.

3.5.5 Register Setup for the QEP Circuit

To start the operation of the QEP circuit in EVA:

1) Load GP timer 2's counter, period, and compare registers with desired values, if necessary

2) Configure T2CON to set GP timer 2 in directional-up/down mode with the QEP circuits as clock source, and enable the selected timer

To start the operation of the QEP circuit in EVB:

1) Load GP timer 4s counter, period, and compare registers with desired values, if necessary

2) Configure T4CON to set GP timer 4 in directional-up/down mode with the QEP circuits as clock source, and enable the selected timer

This page intentionally left blank

Chapter 4

EV Interrupts

This chapter explains the organization of interrupts and describes how to request them.

Topic **Page**

4.1 Event Manager (EV) Interrupt Overview

EV interrupt events are organized into three groups: A, B, and C. Each group is associated with a different interrupt flag and interrupt enable register. There are several event manager peripheral interrupt requests in each EV interrupt group. Table 4−2 shows all EVA interrupts, their priority, and grouping; and Table 4−3 shows all EVB interrupts, their priority, and grouping. There is an interrupt flag register and a corresponding interrupt mask register for each EV interrupt group, as shown in Table 4−1. A flag in EVAIFRx (x = A, B, or C) is masked (will not generate a peripheral interrupt request) if the corresponding bit in EVAIMRx is zero.

Table 4−1. Interrupt Flag Register and Corresponding Interrupt Mask Register

Flag Register	Mask Register	EV Module
EVAIFRA	EVAIMRA	EVA
EVAIFRB	EVAIMRB	
EVAIFRC	EVAIMRC	
EVBIFRA	EVBIMRA	EVB
EVBIFRB	EVBIMRB	
EVBIFRC	EVBIMRC	

4.2 EV Interrupt Request and Service

When a peripheral interrupt request is acknowledged, the appropriate peripheral interrupt vector is loaded into the peripheral interrupt vector register (PIVR) by the PIE controller. The vector loaded into the PIVR is the vector for the highest priority pending enabled event. The vector register can be read by the interrupt service routine (ISR).

Table 4–2. Event Manager A (EVA) Interrupts

Group	Interrupt	Priority within group	Vector (ID)†	Description/Source	INT
A	$\overline{\text{PDPINTA}}$	1 (highest)	0020h	Power Drive Protection Interrupt A	1
A	CMP1INT	2	0021h	Compare Unit 1 compare interrupt	2
	CMP2INT	3	0022h	Compare Unit 2 compare interrupt	
	CMP3INT	4	0023h	Compare Unit 3 compare interrupt	
	T1PINT	5	0027h	GP timer 1 period interrupt	
	T1CINT	6	0028h	GP timer 1 compare interrupt	
	T1UFINT	7	0029h	GP timer 1 underflow interrupt	
	T1OFINT	8	002Ah	GP timer 1 overflow interrupt	
B	T2PINT	1	002Bh	GP timer 2 period interrupt	3
	T2CINT	2	002Ch	GP timer 2 compare interrupt	
	T2UFINT	3	002Dh	GP timer 2 underflow interrupt	
	T2OFINT	4	002Eh	GP timer 2 overflow interrupt	
C	CAP1INT	1	0033h	Capture Unit 1 interrupt	3
	CAP2INT	2	0034h	Capture Unit 2 interrupt	
	CAP3INT	3 (lowest)	0035h	Capture Unit 3 interrupt	

† The Vector ID is used by DSP/BIOS.

Table 4−3. Event Manager B (EVB) Interrupts

Group	Interrupt	Priority within group	Vector (ID)†	Description/Source	INT
A	PDPINTB	1 (highest)	0019h	Power Drive Protection Interrupt B	1
A	CMP4INT	2	0024h	Compare Unit 4 compare interrupt	
	CMP5INT	3	0025h	Compare Unit 5 compare interrupt	
	CMP6INT	4	0026h	Compare Unit 6 compare interrupt	
	T3PINT	5	002Fh	GP timer 3 period interrupt	4
	T3CINT	6	0030h	GP timer 3 compare interrupt	
	T3UFINT	7	0031h	GP timer 3 underflow interrupt	
	T3OFINT	8	0032h	GP timer 3 overflow interrupt	
B	T4PINT	1	0039h	GP timer 4 period interrupt	
	T4CINT	2	003Ah	GP timer 4 compare interrupt	5
	T4UFINT	3	003Bh	GP timer 4 underflow interrupt	
	T4OFINT	4	003Ch	GP timer 4 overflow interrupt	
C	CAP4INT	1	0036h	Capture Unit 4 interrupt	
	CAP5INT	2	0037h	Capture Unit 5 interrupt	5
	CAP6INT	3 (lowest)	0038h	Capture Unit 6 interrupt	

† The Vector ID is used by DSP/BIOS.

Table 4−4. Conditions for Interrupt Generation

Interrupt	Condition For Generation
Underflow	When the counter reaches 0000h
Overflow	When the counter reaches FFFFh
Compare	When the counter register contents match that of the compare register
Period	When the counter register contents match that of the period register

4.2.1 Interrupt Generation

When an interrupt event occurs in the EV module, the corresponding interrupt flag in one of the EV interrupt flag registers is set to one. A peripheral interrupt request is generated to the Peripheral Interrupt Expansion controller, if the flag is locally unmasked (the corresponding bit in EVAIMRx is set to one).

4.2.2 Interrupt Vector

The peripheral interrupt vector corresponding to the interrupt flag that has the highest priority among the flags that are set and enabled is loaded into the PIVR when an interrupt request is acknowledged (this is all done in the peripheral interrupt controller, external to the event manager peripheral).

Note: Failure to Clear the Interrupt Flag Bit

The interrupt flag bit in the peripheral register must be cleared by software writing a 1 to the bit in the ISR. Failure to clear this bit will prevent future interrupt requests by that source.

Chapter 5

EV Registers

This chapter includes all of the event manager (EV) registers, grouped by function.

Topic **Page**

5.1 Register Overview

All EV-A registers are listed in Table 1–2 and EV-B registers are listed in Table 1–3.

5.2 Timer Registers

The timer registers include the following:

- Timer 1 Counter Register (T1CNT) — Address 7401h
- Timer 1 Compare Register (T1CMPR) — Address 7402h
- Timer 1 Period Register (T1PR) — Address 7403h
- Timer 2 Counter Register (T2CNT) — Address 7405h
- Timer 2 Compare Register (T2CMPR) — Address 7406h
- Timer 2 Period Register (T2PR) — Address 7407h
- Timer 3 Counter Register (T3CNT) — Address 7501h
- Timer 3 Compare Register (T3CMPR) — Address 7502h
- Timer 3 Period Register (T3PR) — Address 7503h
- Timer 4 Counter Register (T4CNT) — Address 7505h
- Timer 4 Compare Register (T4CMPR) — Address 7506h
- Timer 4 Period Register (T4PR) — Address 7507h
- Timer 1 Control Register (T1CON) — Address 7404h
- Timer 2 Control Register (T2CON) — Address 7408h
- Timer 3 Control Register (T3CON) — Address 7504h
- Timer 4 Control Register (T4CON) — Address 7508h

Note:

All of these registers are separate and, therefore, independently configurable.

The generic form of each of these registers is shown in Figure 5–1 through Figure 5–6.

Figure 5–1. Timer x Counter Register (TxCNT, where x = 1, 2, 3, or 4)

15	0
T1CNT	
R/W-x	

Legend: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15:0	T1CNT	Holds the instantaneous value of Timer 1 counter

Figure 5–2. Timer x Compare Register (TxCMPR, where x = 1, 2, 3, or 4)

15	0
T1CMPR	
R/W-x	

Legend: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15:0	T1CMPR	Holds the compare value of Timer 1 counter

Figure 5–3. Timer x Period Register (TxPR, where x = 1, 2, 3, or 4)

15	0
T1PR	
R/W-x	

Legend: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15:0	T1PR	Holds the period value of Timer 1 counter

The bit definition of the individual GP timer control registers, TxCON, is shown in Figure 5–4. The bit definition of the overall GP timer control registers, GPTCONA and GPTCONB, are shown in Figure 5–5 (on page 5-5) and Figure 5–6 (on page 5-8), respectively.

Note:

Each Timer Control Register (TxCON) is independently configurable.

Figure 5–4. Timer x Control Register (TxCON; x = 1, 2, 3, or 4)

15	14	13	12	11	10	9	8
Free	Soft	Reserved	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T2SWT1/ T4SWT3†	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR/ SELT3PR†
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read access, W = Write access, -0 = value after reset
† Reserved in T1CON and in T3CON

Figure 5−4. Timer x Control Register (TxCON; x = 1, 2, 3, or 4) (Continued)

Bit(s)	Name	Description
15:14	FREE, SOFT	Emulation control bits 00 Stop immediately on emulation suspend 01 Stop after current timer period is complete on emulation suspend 10 Operation is not affected by emulation suspend 11 Operation is not affected by emulation suspend
13	Reserved	Reads return zero, writes have no effect.
12−11	TMODE1−TMODE0	Count mode selection 00 Stop/Hold 01 Continuous-Up/-Down Count Mode 10 Continuous-Up Count Mode 11 Directional-Up/-Down Count Mode
10−8	TPS2−TPS0	Input clock prescaler 000 x/1 001 x/2 010 x/4 011 x/8 100 x/16 101 x/32 110 x/64 111 x/128 (x = HSPCLK)
7	T2SWT1 T4SWT3	T2SWT1. For EVA, this bit is T2SWT1. (GP timer 2 starts with GP timer 1.) Start GP timer 2 with GP timer 1's timer enable bit. This bit is reserved in T1CON. T4SWT3. For EVB, this bit is T4SWT3. (GP timer 4 starts with GP timer 3.) Start GP timer 4 with GP timer 3's timer-enable bit. This bit is reserved in T3CON. 0 Use own TENABLE bit 1 Use TENABLE bit of T1CON (in case of EVA) or T3CON (in case of EVB) to enable and disable operation ignoring own TENABLE bit
6	TENABLE	Timer enable 0 Disable timer operation (the timer is put in hold and the prescaler counter is reset) 1 Enable timer operations

Figure 5–4. Timer x Control Register (TxCON; x = 1, 2, 3, or 4) (Continued)

Bit(s)	Name	Description
5–4	TCLKS(1,0)	Clock source 00 Internal (i.e., HSPCLK) 01 External (i.e., TCLKINx) 10 Reserved 11 QEP circuit
3–2	TCLD(1,0)	Timer compare register reload condition 00 When counter is 0 01 When counter value is 0 or equals period register value 10 Immediately 11 Reserved
1	TECMPR	Timer compare enable 0 Disable timer compare operation 1 Enable timer compare operation
0	SELT1PR, SELT3PR	SELT1PR. In the case of EVA, this bit is SELT1PR (Period register select). When set to 1 in T2CON, the period register of Timer 1 is chosen for Timer 2 also, ignoring the period register of Timer 2. This bit is a reserved bit in T1CON. SELT3PR. In the case of EVB, this bit is SELT3PR (Period register select). When set to 1 in T4CON, the period register of Timer 3 is chosen for Timer 4 also, ignoring the period register of Timer 4. This bit is a reserved bit in T3CON. 0 Use own period register 1 Use T1PR (in case of EVA) or T3PR (in case of EVB) as period register ignoring own period register

Figure 5–5. GP Timer Control Register A (GPTCONA) — Address 7400h

15	14	13	12	11	10	9	8
Reserved	T2STAT	T1STAT	T2CTRIPE	T1CTRIPE	T2TOADC		T1TOADC
R-0	R-1	R-1	R/W-1	R/W-1	R/W-0		R/W-0

7	6	5	4	3	2	1	0
T1TOADC	TCMPOE	T2CMPOE	T1CMPOE	T2PIN		T1PIN	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

Note: R = Read access, W = Write access, -n = value after reset

Figure 5−5.GP Timer Control Register A (GPTCONA) — Address 7400h (Continued)

Bit(s)	Name	Description
15	Reserved	Reads return zero; writes have no effect.
14	T2STAT	GP timer 2 Status. Read only 0 Counting downward 1 Counting upward
13	T1STAT	GP timer 1 Status. Read only 0 Counting downward 1 Counting upward
12	T2CTRIPE	T2CTRIP Enable. This bit, when active, enables and disables Timer 2 Compare Trip (T2CTRIP). This bit is active only when EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. 0 T2CTRIP is disabled. T2CTRIP does not affect Timer 2 compare output, GPTCON(5), or PDPINT flag (EVIFRA(0)). 1 T2CTRIP is enabled. When T2CTRIP is low, Timer 2 compare output goes into HI-Z state, GPTCON(5) is reset to zero, and PDPINT flag [EVIFRA(0)] is set to one.
11	T1CTRIPE	T1CTRIP Enable. This bit, when active, enables and disables Timer 1 Compare Trip (T1CTRIP) input. This bit is active only when EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. 0 T1CTRIP is disabled. T1CTRIP does not affect Timer 1 compare output, GPTCON(4), or PDPINT flag (EVIFRA(0)). 1 T1CTRIP is enabled. When T1CTRIP is low, Timer 1 compare output goes into HI-Z state, GPTCON(4) is reset to zero, and PDPINT flag (EVIFRA(0)) is set to one.
10−9	T2TOADC	Start ADC with timer 2 event 00 No event starts ADC 01 Setting of underflow interrupt flag starts ADC 10 Setting of period interrupt flag starts ADC 11 Setting of compare interrupt flag starts ADC

Notes: 1) Both GPTCON[12] and GPTCON[11] default to 1 when EXTCON[0] is first set to 1.

2) MUXs replace GPTCON[6] and (EVIMRA(0) | PDPINT) to drive the enabling and disabling of T1PWM_T1CMP and T2PWM_T2CMP outputs separately. Both MUXs are controlled by EXTCON(0):
− When EXTCON(0) = 0, both MUXs select GPTCON(6) and (!EVIMRA(0) | PDPINT).
− When EXTCON(1) = 1, the MUX for T1PWM_T1CMP selects GPTCON(4), and the MUX for T2PWM_T2CMP selects GPTCON(5).

3) (!EVIMRA(0) | PDPINT) represents the asynchronous path of PDPINT pin to the compare output buffers existing in the 240x™ design.

Figure 5−5.GP Timer Control Register A (GPTCONA) — Address 7400h (Continued)

Bit(s)	Name	Description
8−7	T1TOADC	Start ADC with timer 1 event 00 No event starts ADC 01 Setting of underflow interrupt flag starts ADC 10 Setting of period interrupt flag starts ADC 11 Setting of compare interrupt flag starts ADC
6	TCMPOE	Timer compare output enable. This bit, when active, enables and disables timer compare outputs. This bit is active only if EXTCON(0) = 0. This bit is reserved when EXTCON(0) = 1. This bit, when active, is reset to zero when both PDPINT/T1CTRIP are low and EVIMRA(0) = 1. 0 Timer compare outputs, T1/2PWM_T1/2CMP, are in high-impedance state. 1 Timer compare outputs, T1/2PWM_T1/2CMP, are driven by individual timer compare logic.
5	T2CMPOE	Timer 2 compare output enable. This bit, when active, enables and disables EV Timer 2 compare output, T2PWM_T1CMP. This bit is active only if EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. This bit, when active, is reset to zero when T2CTRIP is low and is also enabled. 0 Timer 2 compare output, T2PWM_T2CMP, is in high-impedance state. 1 Timer 2 compare outputs T2PWM_T2CMP, is driven by individual timer 2 compare logic.
4	T1CMPOE	Timer 1 Compare Output Enable. This bit, when active, enables or disables EV Timer 1 compare output T1PWM_T1CMP. This bit is active only when EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. This bit, when active is reset to zero when T1CTRIP is low and is also enabled. 0 Timer 1 compare output, T1PWM_T1CMP, is in HI−z state. 1 Timer 1 compare output, T1PWM_T1CMP, is driven by Timer 1 compare logic.
3−2	T2PIN	Polarity of GP timer 2 compare output 00 Forced low 01 Active low 10 Active high 11 Forced high

Notes: 1) Both GPTCON[12] and GPTCON[11] default to 1 when EXTCON[0] is first set to 1.

2) MUXs replace GPTCON[6] and (EVIMRA(0) | PDPINT) to drive the enabling and disabling of T1PWM_T1CMP and T2PWM_T2CMP outputs separately. Both MUXs are controlled by EXTCON(0):
− When EXTCON(0) = 0, both MUXs select GPTCON(6) and (!EVIMRA(0) | PDPINT).
− When EXTCON(1) = 1, the MUX for T1PWM_T1CMP selects GPTCON(4), and the MUX for T2PWM_T2CMP selects GPTCON(5).

3) (!EVIMRA(0) | PDPINT) represents the asynchronous path of PDPINT pin to the compare output buffers existing in the 240x™ design.

Figure 5−5.GP Timer Control Register A (GPTCONA) — Address 7400h (Continued)

Bit(s)	Name	Description
1−0	T1PIN	Polarity of GP timer 1 compare output 00 Forced low 01 Active low 10 Active high 11 Forced high

Notes: 1) Both GPTCON[12] and GPTCON[11] default to 1 when EXTCON[0] is first set to 1.

2) MUXs replace GPTCON[6] and (EVIMRA(0) | PDPINT) to drive the enabling and disabling of T1PWM_T1CMP and T2PWM_T2CMP outputs separately. Both MUXs are controlled by EXTCON(0):
− When EXTCON(0) = 0, both MUXs select GPTCON(6) and (!EVIMRA(0) | PDPINT).
− When EXTCON(1) = 1, the MUX for T1PWM_T1CMP selects GPTCON(4), and the MUX for T2PWM_T2CMP selects GPTCON(5).

3) (!EVIMRA(0) | PDPINT) represents the asynchronous path of PDPINT pin to the compare output buffers existing in the 240x™ design.

Figure 5−6. GP Timer Control Register B (GPTCONB) — Address 7500h

15	14	13	12	11	10	9	8
Reserved	T4STAT	T3STAT	T4CTRIPE	T3CTRIPE	T4TOADC		T3TOADC
R/W-0	R-1	R-1	R/W-1	R/W-1	R/W-0		R/W-0

7	6	5	4	3	2	1	0
T3TOADC	TCMPOE	T4CMPOE	T3CMPOE	T4PIN		T3PIN	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

Note: R = Read access, W = Write access, -n = value after reset

Bit(s)	Name	Description
15	Reserved	Reads return zero; writes have no effect.
14	T4STAT	GP timer 4 Status. Read only 0 Counting downward 1 Counting upward
13	T3STAT	GP timer 3 Status. Read only 0 Counting downward 1 Counting upward
12	T4CTRIPE	T4CTRIP Enable. This bit, when active, enables and disables Timer 4 Compare Trip (T4CTRIP). This bit is active only when EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. 0 T4CTRIP is disabled. T4CTRIP does not affect Timer 4 compare output, GPTCON(5), or PDPINT flag (EVIFRA(0)). 1 T4CTRIP is enabled. When T4CTRIP is low, Timer 4 compare output goes into HI-Z state, GPTCON(5) is reset to zero, and PDPINT flag [EVIFRA(0)] is set to one.

Figure 5−6.GP Timer Control Register B (GPTCONB) — Address 7500h (Continued)

Bit(s)	Name	Description
11	T3CTRIPE	T3CTRIP Enable. This bit, when active, enables and disables Timer 3 Compare Trip (T3CTRIP) input. This bit is active only when EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. 0 T3CTRIP is disabled. T3CTRIP does not affect Timer 3 compare output, GPTCON(4), or PDPINT flag (EVIFRA(0)). 1 T3CTRIP is enabled. When T3CTRIP is low, Timer 3 compare output goes into HI-Z state, GPTCON(4) is reset to zero, and PDPINT flag (EVIFRA(0)) is set to one.
10−9	T4TOADC	Start ADC with timer 4 event 00 No event starts ADC 01 Setting of underflow interrupt flag starts ADC 10 Setting of period interrupt flag starts ADC 11 Setting of compare interrupt flag starts ADC
8−7	T3TOADC	Start ADC with timer 3 event 00 No event starts ADC 01 Setting of underflow interrupt flag starts ADC 10 Setting of period interrupt flag starts ADC 11 Setting of compare interrupt flag starts ADC
6	TCMPOE	Compare output enable. If $\overline{\text{PDPINTx}}$ is active, this bit it set to zero. 0 Disable all GP timer compare outputs (all compare outputs are put in the high-impedance state) 1 Enable all GP timer compare outputs
5	T4CMPOE	Timer 4 compare output enable. This bit, when active, enables and disables EV Timer 4 compare output, T4PWM_T4CMP. This bit is active only if EXTCON(0) = 0. This bit is reserved when EXTCON(0) = 1. This bit, when active, is reset to zero when T4CTRIP is low and is also enabled. 0 Timer 4 compare output, T4PWM_T4CMP, is in high-impedance state. 1 Timer 4 compare outputs T4PWM_T4CMP, is driven by individual timer 4 compare logic.
4	T3CMPOE	Timer 3 Compare Output Enable. This bit, when active, enables or disables EV Timer 1 compare output T3PWM_T3CMP. This bit is active only when EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. This bit, when active is reset to zero when T3CTRIP is low and is also enabled. 0 Timer 3 compare output, T3PWM_T3CMP, is in HI−z state. 1 Timer 3 compare output, T3PWM_T3CMP, is driven by Timer 3 compare logic.

Figure 5−6.GP Timer Control Register B (GPTCONB) — Address 7500h (Continued)

Bit(s)	Name	Description
3−2	T4PIN	Polarity of GP timer 4 compare output
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high
1−0	T3PIN	Polarity of GP timer 3 compare output
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high

5.3 Compare Control Register

Figure 5–7. Compare Control A (COMCONA) Register — Address 7411h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCMPOE	$\overline{\text{PDPINTA}}$ Status
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
FCMP3OE	FCMP2OE	FCMP1OE	Reserved		C3TRIPE	C2TRIPE	C1TRIPE
R/W-0	R/W-0	R/W-0	R-0		R/W-1	R/W-1	R/W -1

Legend: R = Read, W = Write, -n = reset value

Note: Shaded areas indicate that the bit is active only when the EXTCONA bit 0 = 1.

Bit(s)	Name	Description
15	CENABLE	Compare enable
		0 Disables compare operation. All shadowed registers (CMPRx, ACTRB) become transparent
		1 Enables compare operation
14–13	CLD1, CLD0	
		00 When T3CNT = 0 (that is, underflow)
		01 When T3CNT = 0 or T3CNT = T3PR (that is, on underflow or period match)
		10 Immediately
		11 Reserved; result is unpredictable
12	SVENABLE	Space vector PWM mode enable
		0 Disables space vector PWM mode
		1 Enables space vector PWM mode
11–10	ACTRLD1, ACTRLD0	Action control register reload condition
		00 When T3CNT = 0 (that is, underflow)
		01 When T3CNT = 0 or T3CNT = T3PR (that is, on underflow or period match)
		10 Immediately
		11 Reserved; result is unpredictable
9	FCMPOE	Full Compare Output Enable: This bit, when active, enables and disables all full compare outputs at the same time. This bit is active only if EXTCONA(0) = 0. This bit is reserved when EXTCONA(0) = 1. This bit, when active, is reset to zero when both PDPINTA/T1CTRIP is low and EVAIFRA(0) = 1.
		0 Full compare outputs, PWM1/2/3/4/5/6, are in Hi-Z state.
		1 Full compare outputs, PWM1/2/3/4/5/6, are driven by corresponding compare logic.
8	$\overline{\text{PDPINTA}}$ Status	This bit reflects the current status of the $\overline{\text{PDPINTA}}$ pin.

Figure 5−7. Compare Control A — Address 7411h (COMCONA) Register (Continued)

Bit(s)	Name	Description
7	FCMP3OE	Full Compare 3 Output Enable: This bit, when active, enables or disables Full Compare 3 outputs, PWM5/6. This bit is active only if EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. This bit, when active, is reset to zero when C3TRIP is low and is also enabled. 0 Full Compare 3 outputs, PWM5/6, are in high-impedance state. 1 Full Compare 3 outputs, PWM5/6, are driven by Full Compare 3 logic.
6	FCMP2OE	Full Compare 2 Output Enable: This bit, when active, enables or disables Full Compare 2 outputs, PWM3/4. This bit is active only if EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. This bit, when active, is reset to zero when C2TRIP is low and is also enabled. 0 Full Compare 2 outputs, PWM3/4, are in high-impedance state. 1 Full Compare 2 outputs, PWM3/4, are driven by Full Compare 2 logic.
5	FCMP1OE	Full Compare 1 Output Enable: This bit, when active, enables or disables Full Compare 1 outputs, PWM1/2. This bit is active only if EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. This bit, when active, is reset to zero when C1TRIP is low and is also enabled. 0 Full Compare 1 outputs, PWM1/2, are in high-impedance state. 1 Full Compare 1 outputs, PWM1/2, are driven by Full Compare 1 logic.
4−3	Reserved	
2	C3TRIPE	C3TRIP Enable: This bit, when active, enables or disables Full Compare 3 trip (C3TRIP). This bit is active only if EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. 0 C3TRIP is disabled. C3TRIP does not affect Full Compare 3 outputs, COMCONA(8), or PDPINT flag (EVAIFRA(0)). 1 C3TRIP is enabled. When C3TRIP is low, both Full Compare 3 outputs go into high-impedance state, COMCONA(8) is reset to zero, and PDPINTA flag (EVAIFRA(0)) is set to one.
1	C2TRIPE	C2TRIP Enable: This bit, when active, enables or disables Full Compare 2 trip (C2TRIP). This bit is active only if EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. 0 C2TRIP is disabled. C2TRIP does not affect Full Compare 2 outputs, COMCONA(7), or PDPINTA flag (EVAIFRA(0)). 1 C2TRIP is enabled. When C2TRIP is low, both Full Compare 2 outputs go into high-impedance state, COMCONA(7) is reset to zero, and PDPINTA flag (EVAIFRA(0)) is set to one.

Figure 5–7. Compare Control A — Address 7411h (COMCONA) Register (Continued)

Bit(s)	Name	Description
0	C1TRIPE	C1TRIP Enable: This bit, when active, enables or disables Full Compare 1 trip (C1TRIP). This bit is active only if EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. 0 C1TRIP is disabled. C1TRIP does not affect Full Compare 1 outputs, COMCONA(6), or PDPINTA flag (EVAIFRA(0)). 1 C1TRIP is enabled. When C1TRIP is low, both Full Compare 1 outputs go into high-impedance state, COMCONA(6) is reset to zero, and PDPINTA flag (EVAIFRA(0)) is set to one.

Figure 5–8. Compare Control B (COMCONB) Register — Address 7511h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCMPOE	$\overline{\text{PDPINTB}}$ Status
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0
7	6	5	4	3	2	1	0
FCMP6OE	FCMP5OE	FCMP4OE	Reserved		C6TRIPE	C5TRIPE	C4TRIPE
R/W-0	R/W-0	R/W-0	R-0		R/W-1	R/W-1	R/W -1

Legend: R = Read, W = Write, -n = reset value

Note: Shaded areas indicate that the bit is active only when the EXTCONA bit 0 = 1.

Bit(s)	Name	Description
15	CENABLE	Compare enable 0 Disables compare operation. All shadowed registers (CMPRx, ACTRB) become transparent 1 Enables compare operation
14–13	CLD1, CLD0	 00 When T3CNT = 0 (that is, underflow) 01 When T3CNT = 0 or T3CNT = T3PR (that is, on underflow or period match) 10 Immediately 11 Reserved; result is unpredictable
12	SVENABLE	Space vector PWM mode enable 0 Disables space vector PWM mode 1 Enables space vector PWM mode
11–10	ACTRLD1, ACTRLD0	Action control register reload condition 00 When T3CNT = 0 (that is, underflow) 01 When T3CNT = 0 or T3CNT = T3PR (that is, on underflow or period match) 10 Immediately 11 Reserved; result is unpredictable

Figure 5–8.Compare Control B (COMCONB) Register — Address 7511h (Continued)

9	FCMPOE	Full Compare Output Enable: This bit, when active, enables and disables all full compare outputs at the same time. This bit is active only if EXTCONB(0) = 0. This bit is reserved when EXTCONB(0) = 1. This bit, when active, is reset to zero when both PDPINTB/T3CTRIP is low and EVBIFRA(0) = 1. 0 Full compare outputs, PWM7/8/9/10/11/12, are in high-impedance state. 1 Full compare outputs, PWM7/8/9/10/11/12, are driven by corresponding compare logic.
8	$\overline{\text{PDPINTB}}$ Status	This bit reflects the current status of the $\overline{\text{PDPINTB}}$ pin.
7	FCMP6OE	Full Compare 6 Output Enable: This bit, when active, enables or disables Full Compare 6 outputs, PWM11/12. This bit is active only if EXTCONB(0) = 1. This bit is reserved when EXTCONB(0) = 0. This bit, when active, is reset to zero when C6TRIP is low and is also enabled. 0 Full Compare 6 outputs, PWM11/12, are in high-impedance state. 1 Full Compare 6 outputs, PWM11/12, are driven by Full Compare 6 logic.
6	FCMP5OE	Full Compare 5 Output Enable: This bit, when active, enables or disables Full Compare 5 outputs, PWM9/10. This bit is active only if EXTCONB(0) = 1. This bit is reserved when EXTCONB(0) = 0. This bit, when active, is reset to zero when C5TRIP is low and is also enabled. 0 Full Compare 5 outputs, PWM9/10, are in high-impedance state. 1 Full Compare 5 outputs, PWM9/10, are driven by Full Compare 2 logic.
5	FCMP4OE	Full Compare 4 Output Enable: This bit, when active, enables or disables Full Compare 4 outputs, PWM7/8. This bit is active only if EXTCONB(0) = 1. This bit is reserved when EXTCONB(0) = 0. This bit, when active, is reset to zero when C4TRIP is low and is also enabled. 0 Full Compare 4 outputs, PWM7/8, are in high-impedance state. 1 Full Compare 4 outputs, PWM7/8, are driven by Full Compare 4 logic.
4–3	Reserved	
2	C6TRIPE	C6TRIP Enable: This bit, when active, enables or disables Full Compare 6 trip (C6TRIP). This bit is active only if EXTCONB(0) = 1. This bit is reserved when EXTCONB(0) = 0. 0 C6TRIP is disabled. C6TRIP does not affect Full Compare 6 outputs, COMCON(8), or PDPINTB flag (EVBIFRA(0)). 1 C6TRIP is enabled. When C6TRIP is low, both Full Compare 6 outputs go into high-impedance state, COMCONB(8) is reset to zero, and PDPINTB flag (EVBIFRA(0)) is set to one.

Figure 5–8.Compare Control B (COMCONB) Register — Address 7511h *(Continued)*

Bit(s)	Name	Description
1	C5TRIPE	C5TRIP Enable: This bit, when active, enables or disables Full Compare 5 trip (C5TRIP). This bit is active only if EXTCONB(0) = 1. This bit is reserved when EXTCONB(0) = 0. 0 C5TRIP is disabled. C5TRIP does not affect Full Compare 5 outputs, COMCON(7), or PDPINT flag (EVBIFRA(0)). 1 C5TRIP is enabled. When C5TRIP is low, both Full Compare 5 outputs go into high-impedance state, COMCONB(7) is reset to 0, and PDPINTB flag (EVBIFRA(0)) is set to 1.
0	C4TRIPE	C4TRIP Enable: This bit, when active, enables or disables Full Compare 4 trip (C4TRIP). This bit is active only if EXTCONB(0) = 1. This bit is reserved when EXTCONB(0) = 0. 0 C4TRIP is disabled. C4TRIP does not affect Full Compare 4 outputs, COMCONB(6), or PDPINTB flag (EVBIFRA(0)). 1 C4TRIP is enabled. When C4TRIP is low, both Full Compare 4 outputs go into high-impedance state, COMCONB(6) is reset to zero, and PDPINTB flag (EVBIFRA(0)) is set to one.

Note:

If the CxTRIPE bits are used as GPIO bits, then the compare-trip functionality must be disabled in the COMCONx registers. Otherwise, the corresponding PWM pin(s) might be inadvertently driven into high impedance, when the CxTRIPE/GPIO bit is driven low.

5.4 Compare Action Control Registers

The compare action control registers (ACTRA and ACTRB) control the action that takes place on each of the six compare output pins (PWMx, where x = 1–6 for ACTRA, and x = 7–12 for ACTRB) on a compare event, if the compare operation is enabled by COMCONx[15]. ACTRA and ACTRB are double-buffered. The condition on which ACTRA and ACTRB is reloaded is defined by bits in COMCONx. ACTRA and ACTRB also contain the SVRDIR, D2, D1, and D0 bits needed for space vector PWM operation. The bit configuration of ACTRA is described in Figure 5–9 and that of ACTRB is described in Figure 5–10.

Figure 5–9. Compare Action Control Register A (ACTRA) — Address 7413h

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP6ACT1	CMP6ACT0	CMP5ACT1	CMP5ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
CMP4ACT1	CMP4ACT0	CMP3ACT1	CMP3ACT0	CMP2ACT1	CMP2ACT0	CMP1ACT1	CMP1ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15	SVRDIR	Space vector PWM rotation direction. Used only in space vector PWM output generation. 0 Positive (CCW) 1 Negative (CW)
14–12	D2–D0	Basic space vector bits. Used only in space vector PWM output generation.
11–10	CMP6ACT1–0	Action on compare output pin 6, CMP6. 00 Forced low 01 Active low 10 Active high 11 Forced high
9–8	CMP5ACT1–0	Action on compare output pin 5, CMP5. 00 Forced low 01 Active low 10 Active high 11 Forced high

Figure 5−9.Compare Action Control Register A (ACTRA) — Address 7413h (Continued)

Bit(s)	Name	Description
7−6	CMP4ACT1−0	Action on compare output pin 4, CMP4. 00 Forced low 01 Active low 10 Active high 11 Forced high
5−4	CMP3ACT1−0	Action on compare output pin 3, CMP3 00 Forced low 01 Active low 10 Active high 11 Forced high
3−2	CMP2ACT1−0	Action on compare output pin 2, CMP2 00 Forced low 01 Active low 10 Active high 11 Forced high
1−0	CMP1ACT1−0	Action on compare output pin 1, CMP1 00 Forced low 01 Active low 10 Active high 11 Forced high

Figure 5−10. Compare Action Control Register B (ACTRB) — Address 7513h

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP12ACT1	CMP12ACT0	CMP11ACT1	CMP11ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
CMP10ACT1	CMP10ACT0	CMP9ACT1	CMP9ACT0	CMP8ACT1	CMP8ACT0	CMP7ACT1	CMP7ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15	SVRDIR	Space vector PWM rotation direction. Used only in space vector PWM output generation. 0 Positive (CCW) 1 Negative (CW)
14−12	D2−D0	Basic space vector bits. Used only in space vector PWM output generation.

Figure 5−10. Compare Action Control Register B (ACTRB) — Address 7513h (Continued)

Bit(s)	Name	Description
11−10	CMP12ACT1−0	Action on compare output pin 12, CMP12.
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high
9−8	CMP11ACT1−0	Action on compare output pin 11, CMP11.
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high
7−6	CMP10ACT1−0	Action on compare output pin 10, CMP10.
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high
5−4	CMP9ACT1−0	Action on compare output pin 9, CMP9
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high
3−2	CMP8ACT1−0	Action on compare output pin 8, CMP8
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high
1−0	CMP7ACT1−0	Action on compare output pin 7, CMP7
		00 Forced low
		01 Active low
		10 Active high
		11 Forced high

5.5 Capture Unit Registers

The operation of the capture units is controlled by four 16-bit control registers, CAPCONA/B and CAPFIFOA/B. TxCON (x = 1, 2, 3, or 4) registers are also used to control the operation of the capture units since the time base for capture circuits can be provided by any of these timers.

Figure 5–11. Capture Control Register A (CAPCONA) — Address 7420h

15	14	13	12	11	10	9	8
CAPRES	CAP12EN		CAP3EN	Reserved	CAP3TSEL	CAP12TSEL	CAP3TOADC
RW-0	RW-0		RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
CAP1EDGE		CAP2EDGE		CAP3EDGE		Reserved	
RW-0		RW-0		RW-0		RW-0	

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15	CAPRES	Capture reset. Always reads zero.
		0 Clear all registers of capture units to 0
		1 No action
14–13	CAP12EN	Captures 1 and 2 Enable:
		00 Disable captures 1 and 2. FIFO stacks retain their contents.
		01 Enable captures 1 and 2.
		10 Reserved
		11 Reserved
12	CAP3EN	Capture 3 Enable:
		0 Disables Capture Unit 3; FIFO stack of Capture Unit 3 retains its contents
		1 Enable capture 3.
11	Reserved	Reads return zero; writes have no effect.
10	CAP3TSEL	GP timer selection for capture unit 3.
		0 Selects GP timer 2
		1 Selects GP timer 1
9	CAP12TSEL	GP timer selection for capture units 1 and 2.
		0 Selects GP timer 2
		1 Selects GP timer 1
8	CAP3TOADC	Capture unit 3 event starts ADC.
		0 No action
		1 Starts ADC when the CAP3INT flag is set

Figure 5−11. Capture Control Register A (CAPCONA) — Address 7420h (Continued)

Bit(s)	Name	Description
7−6	CAP1EDGE	Edge detection control for Capture Unit 1.
		00 No detection
		01 Detects rising edge
		10 Detects falling edge
		11 Detects both edges
5−4	CAP2EDGE	Edge detection control for Capture Unit 2.
		00 No detection
		01 Detects rising edge
		10 Detects falling edge
		11 Detects both edges
3−2	CAP3EDGE	Edge detection control for Capture Unit 3.
		00 No detection
		01 Detects rising edge
		10 Detects falling edge
		11 Detects both edges
1−0	Reserved	Reads return zero; writes have no effect.

Figure 5−12. Capture Control Register B (CAPCONB) — Address 7520h

15	14	13	12	11	10	9	8
CAPRES	CAP45EN		CAP6EN	Reserved	CAP6TSEL	CAP45TSEL	CAP6TOADC
R/W-0	R/W-0		R/W-0	R-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
CAP4EDGE		CAP5EDGE		CAP6EDGE		Reserved	
R/W-0		R/W-0		R/W-0		R/W-0	

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15	CAPRES	This bit is not implemented as a register bit. Writing a 0 simply clears the capture registers.
		0 Clear all registers of capture units and QEP circuit to 0
		1 No action
14−13	CAP45EN	Capture Units 4 and 5 and QEP circuit control.
		00 Disables Capture Units 4 and 5. FIFO stacks retain their contents
		01 Enables Capture Units 4 and 5
		10 Reserved
		11 Reserved

Figure 5−12. Capture Control Register B (CAPCONB) — Address 7520h (Continued)

12	CAP6EN	Capture unit 6 control
		0 Disables Capture Unit 6; FIFO stack of Capture Unit 6 retains its contents
		1 Enables Capture Unit 6
11	Reserved	Reads return zero; writes have no effect.
10	CAP6TSEL	GP timer selection for Capture Unit 6
		0 Selects GP timer 4
		1 Selects GP timer 3
9	CAP45TSEL	GP timer selection for Capture Units 4 and 5
		0 Selects GP timer 4
		1 Selects GP timer 3
8	CAP6TOADC	Capture Unit 6 event starts ADC.
		0 No action
		1 Starts ADC when the CAP6INT flag is set.
7−6	CAP4EDGE	Edge detection control for Capture Unit 4.
		00 No detection
		01 Detects rising edge
		10 Detects falling edge
		11 Detects both edges
5−4	CAP5EDGE	Edge detection control for Capture Unit 5.
		00 No detection
		01 Detects rising edge
		10 Detects falling edge
		11 Detects both edges
3−2	CAP6EDGE	Edge detection control for Capture Unit 6.
		00 No detection
		01 Detects rising edge
		10 Detects falling edge
		11 Detects both edges
1−0	Reserved	

5.5.1 Capture FIFO Status Register A (CAPFIFOA)

CAPFIFOA contains the status bits for each of the three FIFO stacks of the capture units. The bit description of CAPFIFOA is given in Figure 5−13. If a write occurs to the CAPnFIFOA status bits at the same time as they are being updated (because of a capture event), the write data takes precedence.

The write operation to the CAPFIFOx registers can be used as a programming advantage. For example, if 01 is written into the CAPnFIFO bits, the EV module is led to “believe” that there is already an entry in the FIFO. Subsequently, every time the FIFO gets a new value, a capture interrupt will be generated.

Figure 5−13. Capture FIFO Status Register A (CAPFIFOA) — Address 7422h

15 14	13 12	11 10	9 8	7 0
Reserved	CAP3FIFO	CAP2FIFO	CAP1FIFO	Reserved
R-0	R/W-0	R/W-0	R/W-0	R-0

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15−14	Reserved	Reads return zero; writes have no effect.
13−12	CAP3FIFO	CAP3FIFO status
		00 Empty
		01 Has one entry
		10 Has two entries
		11 Had two entries and captured another one; first entry has been lost
11−10	CAP2FIFO	CAP2FIFO status
		00 Empty
		01 Has one entry
		10 Has two entries
		11 Had two entries and captured another one; first entry has been lost
9−8	CAP1FIFO	CAP1FIFO status
		00 Empty
		01 Has one entry
		10 Has two entries
		11 Had two entries and captured another one; first entry has been lost
7−0	Reserved	Reads return zero; writes have no effect.

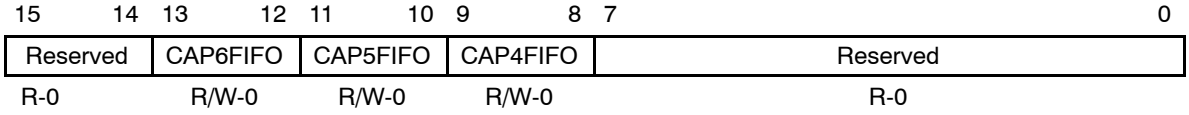
5.5.2 Capture FIFO Status Register B (CAPFIFOB)

CAPFIFOB contains the status bits for each of the three FIFO stacks of the capture units. The bit description of CAPFIFOB is given in Figure 5−14. If a

write occurs to the CAPnFIFOB status bits at the same time as they are being updated (because of a capture event), the write data takes precedence.

The write operation to the CAPFIFOx registers can be used as a programming advantage. For example, if 01 is written into the CAPnFIFO bits, the EV module is led to “believe” that there is already an entry in the FIFO. Subsequently, every time the FIFO gets a new value, a capture interrupt is generated.

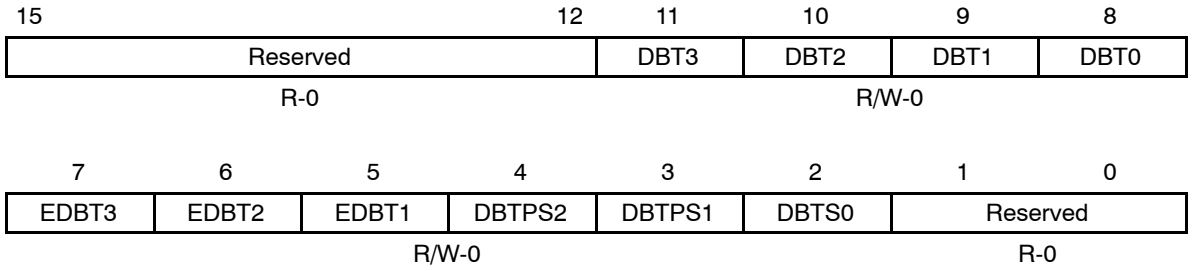
Figure 5−14. Capture FIFO Status Register B (CAPFIFOB) — Address 7522h



Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15−14	Reserved	Reads return zero; writes have no effect.
13−12	CAP6FIFO	CAP6FIFO Status
		00 Empty
		01 Has one entry
		10 Has two entries
		11 Had two entries and captured another one; first entry has been lost
11−10	CAP5FIFO	CAP5FIFO Status
		00 Empty
		01 Has one entry
		10 Has two entries
		11 Had two entries and captured another one; first entry has been lost
9−8	CAP4FIFO	CAP4FIFO Status
		00 Empty
		01 Has one entry
		10 Has two entries
		11 Had two entries and captured another one; first entry has been lost
7−0	Reserved	Reads return zero; writes have no effect.

Figure 5−15. Dead-Band Timer Control Register A (DBTCONA) — Address xx15h



Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15−12	Reserved	
11−8	DBT3 (MSB) − DBT0 (LSB)	Dead-band timer period. These bits define the period value of the three 4-bit dead-band timers.
7	EDBT3	Dead-band timer 3 enable (for pins PWM5 and PWM6 of Compare Unit 3). 0 Disable 1 Enable
6	EDBT2	Dead-band timer 2 enable (for pins PWM3 and PWM4 of Compare Unit 2). 0 Disable 1 Enable
5	EDBT1	Dead-band timer 1 enable (for pins PWM1 and PWM2 of Compare Unit 1). 0 Disable 1 Enable
4−2	DBTPS2 − DBTPS0	Dead-band timer prescaler 000 x/1 001 x/2 010 x/4 011 x/8 100 x/16 101 x/32 110 x/32 111 x/32 111 x = Device (CPU) clock frequency
1−0	Reserved	

Figure 5−16. Dead-Band Timer Control Register B (DBTCONB) — Address xx15h

15	14	13	12	11	10	9	8
Reserved				DBT3	DBT2	DBT1	DBT0
R-0				R/W-0			

7	6	5	4	3	2	1	0
EDBT3	EDBT2	EDBT1	DBTPS2	DBTPS1	DBTS0	Reserved	
R/W-0						R-0	

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15−12	Reserved	
11−8	DBT3 (MSB) − DBT0 (LSB)	Dead-band timer period. These bits define the period value of the three 4-bit dead-band timers.
7	EDBT3	Dead-band timer 3 enable (for pins PWM11 and PWM12 of Compare Unit 3). 0 Disable 1 Enable
6	EDBT2	Dead-band timer 2 enable (for pins PWM9 and PWM10 of Compare Unit 2). 0 Disable 1 Enable
5	EDBT1	Dead-band timer 1 enable (for pins PWM7 and PWM8 of Compare Unit 1). 0 Disable 1 Enable
4−2	DBTPS2 − DBTPS0	Dead-band timer prescaler 000 x/1 001 x/2 010 x/4 011 x/8 100 x/16 101 x/32 110 x/32 111 x/32 111 x = Device (CPU) clock frequency
1−0	Reserved	

5.6 EV Interrupt Flag Registers

The registers are all treated as 16-bit memory mapped registers. The unused bits all return zero when read by software. Writing to unused bits has no effect. Since EVxIFRx are readable registers, occurrence of an interrupt event can be monitored by software polling the appropriate bit in EVxIFRx when the interrupt is masked.

Figure 5–17. EVA Interrupt Flag Register A (EVAIFRA) — Address 742Fh

15						10	9	8
Reserved						TIOFINT FLAG	T1UFINT FLAG	T1CINT FLAG
R-0						R/W-0	R/W-0	R/W-0
7	6	5	4	3	2	1	0	
T1PINT FLAG	Reserved			CMP3INT FLAG	CMP2INT FLAG	CMP1INT FLAG	PDPINTA FLAG	
R/W-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0	

Note: R = Read access, W = Write access, -n = value after reset

Bit(s)	Name	Description
15–11	Reserved	Reserved. Reads return 0; writes have no effect.
10	T1OFINT FLAG	GP timer 1 overflow interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
9	T1UFINT FLAG	GP timer 1 underflow interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
8	T1CINT FLAG	GP timer 1 compare interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag

Figure 5−17. EVA Interrupt Flag Register A (EVAIFRA) — Address 742Fh (Continued)

7	T1PINT FLAG	GP timer 1 compare interrupt		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
6−4	Reserved	Reads return zero; writes have no effect		
3	CMP3INT FLAG	Compare 3 interrupt		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
2	CMP2INT FLAG	Compare 2 interrupt		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
1	CMP1INT FLAG	Compare 1 interrupt		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
0	PDPINTA FLAG	Power Drive Protection Interrupt Flag: The definition of this bit depends on EXTCONA(0). When EXTCONA(0) = 0, the definition remains the same as 240x. When EXTCONA(0) = 1, this bit is set when any compare trip is low and is also enabled.		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag

Figure 5−18. EVA Interrupt Flag Register B (EVAIFRB) — Address 7430h

15	14	13	12	11	10	9	8
Reserved							
R/W-0							

7	6	5	4	3	2	1	0
Reserved				T2OFINT FLAG	T2UFINT FLAG	T2CINT FLAG	T2PINT FLAG
R-0				RW1C-0	RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Bit(s)	Name	Description
15−14	Reserved	Reads return 0; writes have no effect.
3	T2OFINT FLAG	GP timer 2 overflow interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
2	T2UFINT FLAG	GP timer 2 underflow interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
1	T2CINT FLAG	GP timer 2 compare interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
0	T2PINT FLAG	GP timer 2 period interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag

Figure 5−19. EVA Interrupt Flag Register C (EVAIFRC) — Address 7431h

15	8
Reserved	
R-0	

7	3	2	1	0
Reserved		CAP3FINT FLAG	CAP2FINT FLAG	CAP1FINT FLAG
R-0		RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Bit(s)	Name	Description
15−3	Reserved	Reads return 0; writes have no effect.
2	CAP3FINT FLAG	Capture 3 interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
1	CAP2FINT FLAG	Capture 2 interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
0	CAP1FINT FLAG	Capture 1 interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag

Figure 5−20. EVA Interrupt Mask Register A (EVAIMRA) — Address 742Ch

15	11	10	9	8
Reserved		T1OFINT	T1UFINT	T1CINT
R-0		R/W-0	R/W-0	R/W-0

7	6	4	3	2	1	0
T1PINT	Reserved		CMP3INT	CMP2INT	CMP1INT	PDPINTA
R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-1

Note: R = Read access, W = write access, -n = value after reset

Bit(s)	Name	Description
15−11	Reserved	
10	T1OFINT	T1OFINT enable
		0 Disable
		1 Enable
9	T1UFINT	T1UFINT enable
		0 Disable
		1 Enable
8	T1CINT	T1CINT enable
		0 Disable
		1 Enable
7	T1PINT	T1PINT enable
		0 Disable
		1 Enable
6−4	Reserved	
3	CMP3INT	CMP3INT enable
		0 Disable
		1 Enable
2	CMP2INT	CMP2INT enable
		0 Disable
		1 Enable

Figure 5−20. EVA Interrupt Mask Register A (EVAIMRA) — Address 742Ch (Continued)

Bit(s)	Name	Description
1	CMP1INT	CMP1INT enable 0 Disable 1 Enable
0	PDPINTA	PDPINTA ENABLE. The definition of this bit depends on EXTCONA(0). When EXTCONA(0) = 0, the definition remains the same as 240x, i.e., this bit enables and disables both PDP interrupt and the direct path of PDPINT pin to compare output buffers. When EXTCONA(0) = 1, this bit becomes just a PDP interrupt enable and disable bit. 0 Disable 1 Enable

Figure 5−21. EVA Interrupt Mask Register B (EVAIMRB) — Address 742Dh

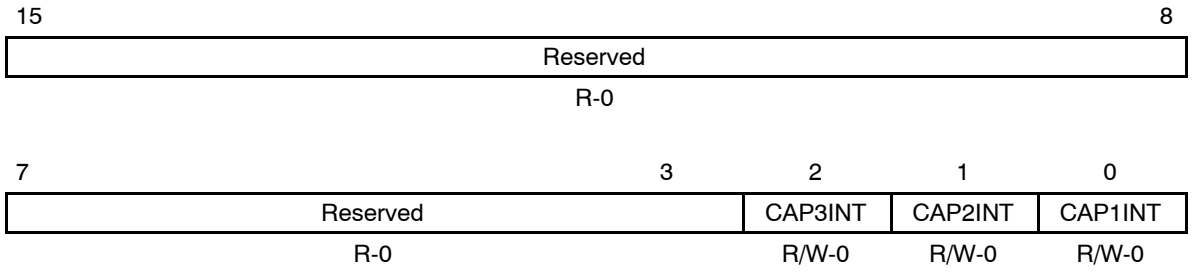
15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		T2OFINT	T2UFINT	T2CINT	T2PINT
R-0		R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15−4	Reserved	
3	T2OFINT	T2OFINT enable 0 Disable 1 Enable
2	T2UFINT	T2UFINT enable 0 Disable 1 Enable
1	T2CINT	T2CINT enable 0 Disable 1 Enable
0	T2PINT	T2PINT enable 0 Disable 1 Enable

Figure 5−22. EVA Interrupt Mask Register C (EVAIMRC) — Address 742Eh



Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description
15−3	Reserved	
2	CAP3INT ENABLE	CAP3INT Enable 0 Disable 1 Enable
1	CAP2INT ENABLE	CAP2INT Enable 0 Disable 1 Enable
0	CAP1INT ENABLE	CAP1INT Enable 0 Disable 1 Enable

Figure 5−23. EVB Interrupt Flag Register A (EVBIFRA) — Address 752Fh

15	11	10	9	8
Reserved		T3OFINT FLAG	T3UFINT FLAG	T3CINT FLAG
R-0		RW1C-0	RW1C-0	RW1C-0

7	6	4	3	2	1	0
T3PINT FLAG	Reserved		CMP6INT	CMP5INT	CMP4INT	PDPINTB
RW1C-0	R-0		RW1C-0	RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Bit(s)	Name	Description
15−11	Reserved	Reads return 0; writes have no effect.
10	T3OFINT	T3OFINT FLAG. GP timer 3 overflow interrupt.
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
9	T3UFINT	T3UFINT FLAG. GP timer 3 underflow interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag
8	T3CINT	T3CINT FLAG. GP timer 3 compare interrupt
		Read: 0 Flag is reset
		1 Flag is set
		Write: 0 No effect
		1 Resets flag

Figure 5−23. EVB Interrupt Flag Register A (EVBIFRA) — Address 752Fh (Continued)

7	T3PINT	T3PINT FLAG. GP timer 3 period interrupt		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
6−4	Reserved			
3	CMP6INT	CMP6INT FLAG. Compare 6 interrupt.		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
2	CMP5INT	CMP6INT FLAG. Compare 5 interrupt.		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
1	CMP4INT	CMP6INT FLAG. Compare 4 interrupt.		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag
0	PDPINTB	PDPINTB FLAG. Power drive protection interrupt.		
		Read:	0	Flag is reset
			1	Flag is set
		Write:	0	No effect
			1	Resets flag

Figure 5–24. EVB Interrupt Flag Register B (EVBIFRB) — Address 7530h

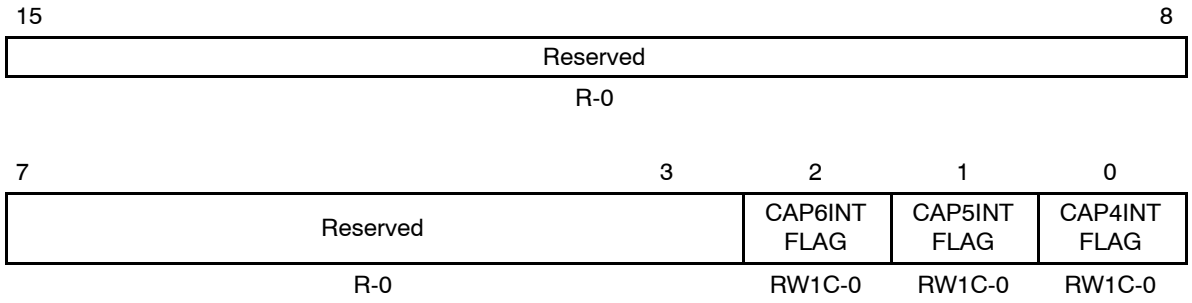
15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved				T4OFINT FLAG	T4UFINT FLAG	T4CINT FLAG	T4PINT FLAG
R-0				RW1C-0	RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Bit(s)	Name	Description
15–4	Reserved	
3	T4OFINT FLAG	GP timer 4 overflow interrupt Read: 0 Flag is reset 1 Flag is set Write: 0 No effect 1 Resets flag
2	T4UFINT FLAG	GP timer 4 underflow interrupt Read: 0 Flag is reset 1 Flag is set Write: 0 No effect 1 Resets flag
1	T4CINT FLAG	GP timer 4 compare interrupt Read: 0 Flag is reset 1 Flag is set Write: 0 No effect 1 Resets flag
0	T4PINT FLAG	GP timer 4 period interrupt Read: 0 Flag is reset 1 Flag is set Write: 0 No effect 1 Resets flag

Figure 5−25. EVB Interrupt Flag Register C (EVBIFRC) — Address 7531h



Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Bit(s)	Name	Description
15−3	Reserved	
2	CAP6INT FLAG	GP timer 4 overflow interrupt. Read: 0 Flag is reset 1 Flag is set Write: 0 No effect 1 Resets flag
1	CAP5INT FLAG	GP timer 4 overflow interrupt. Read: 0 Flag is reset 1 Flag is set Write: 0 No effect 1 Resets flag
0	CAP4INT FLAG	GP timer 4 overflow interrupt. Read: 0 Flag is reset 1 Flag is set Write: 0 No effect 1 Resets flag

Figure 5–26. EVB Interrupt Mask Register A (EVBIMRA) — Address 752Ch

15	11	10	9	8
Reserved		T3OFINT ENABLE	T3UFINT ENABLE	T3CINT ENABLE
R/W-0		R/W-0	R/W-0	R/W-0

7	6	4	3	2	1	0
T3PINT ENABLE	Reserved		CMP6INT ENABLE	CMP5INT ENABLE	CMP4INT ENABLE	PDPINTB ENABLE
R/W-0	R-0		R/W-0	R/W-0	R/W-0	R/W-1

Note: R = Read access, W = Write access, -n = value after reset

Bit(s)	Name	Description
15–11	Reserved	
10	T3OFINT ENABLE	T3OFINT Enable 0 Disable 1 Enable
9	T3UFINT ENABLE	T3UFINT Enable 0 Disable 1 Enable
8	T3CINT ENABLE	T3CINT Enable 0 Disable 1 Enable
7	T3PINT ENABLE	T3PINT Enable 0 Disable 1 Enable
6–4	Reserved	
3	CMP6INT ENABLE	CMP6INT Enable 0 Disable 1 Enable
2	CMP5INT ENABLE	CMP5INT Enable 0 Disable 1 Enable
1	CMP4INT ENABLE	CMP4INT Enable 0 Disable 1 Enable
0	PDPINTB ENABLE	PDPINTB Enable. This is enabled (set to 1) following reset 0 Disable 1 Enable

Figure 5–27. EVB Interrupt Mask Register B (EVBIMRB) — Address 752Dh

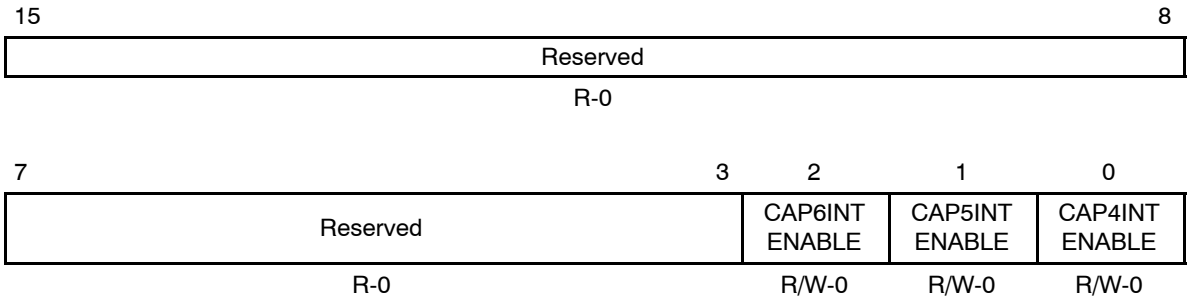
15	8
Reserved	
R-0	

7	4	3	2	1	0
Reserved		T4OFINT ENABLE	T4UFINT ENABLE	T4CINT ENABLE	T4PINT ENABLE
R-0		RW-0	RW-0	RW-0	RW-0

Note: R = Read access, W = Write access, -0 = value after reset

Bit(s)	Name	Description	
15–4	Reserved		
3	T4OFINT ENABLE		
		0	Disable
		1	Enable
2	T4UFINT ENABLE		
		0	Disable
		1	Enable
1	T4CINT ENABLE		
		0	Disable
		1	Enable
0	T4PINT ENABLE		
		0	Disable
		1	Enable

Figure 5–28. EVB Interrupt Mask Register C (EVBIMRC) — Address 752Eh



Note: R = Read access, W = Write access, -n = value after reset

Bit(s)	Name	Description	
15–3	Reserved		
2	CAP6INT ENABLE		
		0	Disable
		1	Enable
1	CAP5INT ENABLE		
		0	Disable
		1	Enable
0	CAP4INT ENABLE		
		0	Disable
		1	Enable

5.7 EV Control Registers

EXTCONA and EXTCONB are added control registers to enable and disable the added/modified features. The EXTCONx registers are required for compatibility with 240x EV. EXTCONx enables and disables the additions and modifications in features. All additions and modifications are disabled by default to keep compatibility with 240x EV. The description applies to EXTCONA. EXTCONB is identical to this register except that it controls the EVB register set.

Figure 5–29. EV Extension Control Register A (EXTCONA) — Address 7409h

15	14	13	12	11	10	9	8
Reserved							
R-0							

7	6	5	4	3	2	1	0
Reserved				EVSOCE	QEPIE	QEPIQUAL	INDCOE
R-0				R/W-0	R/W-0	R/W-0	R/W-0

Bit(s)	Name	Description
15:4	Reserved	
3	EVSOCE	EV Start-of-Conversion Output Enable. This bit enables and disables the ADC start-of-conversion output of EV (i.e., EVASOCn for EVA and EVBSOCn for EVB). When enabled, a negative (active-low) pulse of 32 x HSPCLK is generated on selected EV ADC start-of-conversion event. This bit does not affect the EVTOADC signal routed to the ADC module as optional SOC trigger. 0 Disables $\overline{\text{EVSOC}}$ output. $\overline{\text{EVSOC}}$ is in high-impedance state. 1 Enables $\overline{\text{EVSOC}}$ output.
2	QEPIE	QEP Index Enable. This bit enables and disables the CAP3_QEPI1 as index input. CAP3_QEPII, when enabled as index input, can cause the timer configured as the QEP counter to reset. 0 Disables CAP3_QEPI1 as index input. Transitions on CAP3_QEPI1 do not affect the timer configured as the QEP counter. 1 Enables CAP3_QPII as index input. Either a zero-to-one transition on CAP3_QEPI1 alone (when EXTCONA[1] = 0), or a zero-to-one transition plus CAP1_QEP1 and CAP2_QEP2 are both high (when EXTCON[1] = 1), causes the timer configured as QEP counter to reset to zero.
1	QEPIQUAL	CAP3_QEPI1 Index Qualification Mode. This bit turns on and off QEP index qualifier. 0 CAP3_QEPI1 qualification mode is off. CAP3_QEPI1 is allowed to pass the qualifier unaffected. 1 CAP3_QEPI1 qualification mode is on. A zero-to-one transition is allowed to pass the qualifier only when both CAP1_QEP1 and CAP2_QEP2 are high. Otherwise the output of the qualifier stays low.

Figure 5−29. EV Control Register (EXTCONA) — Address 7409h *(Continued)*

Bit(s)	Name	Description
0	INDCOE	Independent compare output enable mode. This bit, when set to one, allows compare outputs to be enabled and disabled independently. 0 Independent compare output enable mode is disabled. Time 1 and 2 compare outputs are enabled and disabled at the same time by GPTCONA(6). Full Compare 1, 2, and 3 outputs are enabled and disabled at the same time by COMCONA(9). GPTCONA(12,11,5,4) and COMCONA(7:5, 2:0) are reserved. EVIFRA(0) enables and disables all the compare outputs at the same time. EVIMR(0) enables and disables PDP interrupt and the direct path of $\overline{\text{PDPINT}}$ signal at the same time. 1 Independent compare output enable mode is enabled. Compare outputs are enabled and disabled respectively by GPTCON(5,4) and COMCON(7:5). Compare trips are enabled and disabled respectively by GPTCON(12,11) and COMCON(2:0). GPTCON(6) and COMCON(9) are reserved. EVIFRA[0] is set to one when any trip input is low and is also enabled. EVIMRA(0) functions only as interrupt enable and disable.

5.8 Differences in Register Bit Definitions

The changes described here are for one EV. The same changes must be implemented in both EVA and EVB. This includes the addition of the EXTCONx control register, i.e., an EXTCONx register is added to each EV instance, one in EVA, and another in EVB.

Changes are introduced to registers as shown in Table 5−1. Only the bits that changed are shown; all others are the same as they were in the 240x EV. See the individual registers in this chapter for complete bit descriptions.

Table 5−1. Register Bit Changes

Bit(s)	Name	Description
TXCON Register Bit Changes		
5,4	TCLKS(1,0)	Timer 2 Clock Source 00 Internal, i.e., HSPCLK 01 External, i.e., TCLKIN 10 Reserved 11 QEP circuit After the change, both Timers 1 and 2 (and, similarly, both Timers 3 and 4) are allowed to use QEP circuit as clock source.
GPTCON Register Bit Changes		
12	T2CTRIPE	T2CTRIP Enable: This bit, when active, enables and disables Timer 2 Compare Trip (T2CTRIP). This bit is active only when EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0: 0 T2CTRIP is disabled. T2CTRIP does not affect Timer 2 compare output, GPTCON(5), or PDPINT flag (EVIFRA(0)). 1 T2CTRIP is enabled. When T2CTRIP is low, Timer 2 compare output goes into high-impedance state, GPTCON(5) is reset to zero, and PDPINT flag (EVIFRA(0)) is set to one.
11	T1CTRIPE	T1CTRIP Enable: This bit, when active, enables and disables Timer 1 Compare Trip (T1CTRIP) input. This bit is active only when EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0: 0 T1CTRIP is disabled. T1CTRIP does not affect Timer 1 compare output, GPTCON(4), or PDPINT flag (EVIFRA(0)). 1 T1CTRIP is enabled. When T1CTRIP is low, Timer 1 compare output goes into high-impedance state, GPTCON(4) is reset to zero, and PDPINT flag (EVIFRA(0)) is set to one.

Table 5–1. Register Bit Changes (Continued)

Bit(s)	Name	Description
6	TCMPOE	Timer Compare Output Enable: This bit, when active, enables and disables timer compare outputs. This bit is active only if EXTCONA(0) = 0. This bit is reserved when EXTCONA(0) = 1. This bit, when active, is reset to zero when both PDPINT/T1CTRIP is low and EVIMRA(0) = 1 are true: 0 Timer compare outputs, T1/2PWM_T1/2CMP, are in high-impedance state. 1 Timer compare outputs, T1/2PWM_T1/2CMP, are driven by individual timer compare logic.
5	T2CMPOE	Timer 2 Compare Output Enable: This bit, when active, enables or disables EV Timer 2 compare output, T2PWM_T2CMP. This bit is active only if EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. This bit, when active, is reset to zero when T2CTRIP is low and is also enabled: 0 Timer 2 compare output, T2PWM_T2CMP, is in high-impedance state. 1 Timer 2 compare output, T2PWM_T2CMP, is driven by Timer 2 compare logic.
4	T1CMPOE	Timer 1 Compare Output Enable: This bit, when active, enables or disables EV Timer 1 compare output, T1PWM_T1CMP. This bit is active only EXTCONA(0) = 1. This bit is reserved when EXTCONA(0) = 0. This bit, when active, is reset to zero when T1CTRIP is low and is also enabled: 0 Timer 1 compare output, T1PWM_T1CMP, is in high-impedance state. 1 Timer 1 compare output, T1PWM_T1CMP, is driven by Timer 1 compare logic.
		COMCON Register Bit Changes
9	FCMPOE	Full Compare Output Enable: This bit, when active, enables and disables all full compare outputs at the same time. This bit is active only if EXTCONA(0) = 0. This bit is reserved when EXTCONA(0) = 1. This bit, when active is reset to zero when both PDPINT/T1CTRIP is low and EVIFRA(0) = 1: 0 Full compare outputs, PWM1/2/3/4/5/6, are in high-impedance state. 1 Full compare outputs, PWM1/2/3/4/5/6, are driven by corresponding compare logic.
8	PDPINT	Status of $\overline{\text{PDPINT}}$ pin

Table 5−1. Register Bit Changes (Continued)

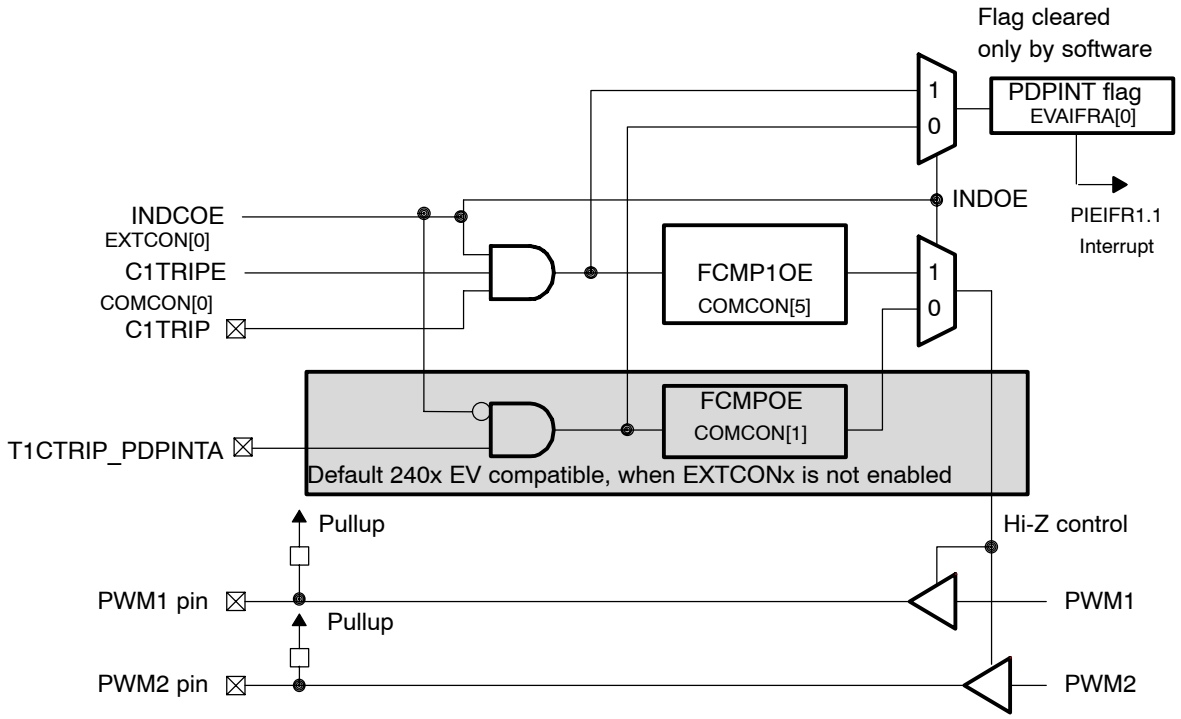
Bit(s)	Name	Description
7	FCMP3OE	Full Compare 3 Output Enable: This bit, when active, enables or disables Full Compare 3 outputs, PWM5/6. This bit is active only if EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. This bit, when active, is reset to zero when C3TRIP is low and is also enabled: 0 Full Compare 3 outputs, PWM5/6, are in high-impedance state. 1 Full Compare 3 outputs, PWM5/6, are driven by Full Compare 3 logic.
6	FCMP2OE	Full Compare 2 Output Enable: This bit, when active, enables or disables Full Compare 2 outputs, PWM4/5. This bit is active only if EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. This bit, when active, is reset to zero when C2TRIP is low and is also enabled: 0 Full Compare 2 outputs, PWM4/5, are in high-impedance state. 1 Full Compare 2 outputs, PWM4/5, are driven by Full Compare 2 logic.
5	FCMP1OE	Full Compare 1 Output Enable: This bit, when active, enables or disables Full Compare 1 outputs, PWM1/2. This bit is active only if EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0. This bit, when active, is reset to zero when C1TRIP is low and is also enabled: 0 Full Compare 1 outputs, PWM1/2, are in high-impedance state. 1 Full Compare 1 outputs, PWM1/2, are driven by Full Compare 1 logic.
4:3	reserved	
2	C3TRIPE	C3TRIP Enable: This bit, when active, enables or disables Full Compare 3 trip (C3TRIP). This bit is active only if EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0: 0 C3TRIP is disabled. C3TRIP does not affect Full Compare 3 outputs, COMCON(8), or PDPINT flag (EVIFRA(0)). 1 C3TRIP is enabled. When C3TRIP is low, both Full Compare 3 outputs go into high-impedance state, COMCON(8) is reset to zero, and PDPINT flag (EVIFRA(0)) is set to one.
1	C2TRIPE	C2TRIP Enable: This bit, when active, enables or disables Full Compare 2 trip (C2TRIP). This bit is active only if EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0: 0 C2TRIP is disabled. C2TRIP does not affect Full Compare 2 outputs, COMCON(7), or PDPINT flag (EVIFRA(0)). 1 C2TRIP is enabled. When C2TRIP is low, both Full Compare 2 outputs go into high-impedance state, COMCON(7) is reset to zero, and PDPINT flag (EVIFRA(0)) is set to one.

Table 5−1. Register Bit Changes (Continued)

Bit(s)	Name	Description
0	C1TRIPE	C1TRIP Enable: This bit, when active, enables or disables Full Compare 1 trip (C1TRIP). This bit is active only if EXTCON(0) = 1. This bit is reserved when EXTCON(0) = 0: 0 C1TRIP is disabled. C1TRIP does not affect Full Compare 1 outputs, COMCON(6), or PDPINT flag (EVIFRA(0)). 1 C1TRIP is enabled. When C1TRIP is low, both Full Compare 1 outputs go into high-impedance state, COMCON(6) is reset to zero, and PDPINT flag (EVIFRA(0)) is set to one.
CAPCON Register Bit Changes		
13:14	CAP12EN	Captures 1 and 2 Enable: 00 Disable Captures 1 and 2. FIFO stacks retain their contents. 01 Enable Captures 1 and 2 10 Reserved 11 Reserved An early version of the 240x User's Guide wrongly assumed that CAPCON(13:14) also controls the enabling and disabling of QEP circuit.
EVIFRA Register Bit Changes		
0	PDPINT	Power Drive Protection Interrupt Flag: The definition of this bit depends on EXTCON(0). When EXTCON(0) = 0, the definition remains the same as '240x. When EXTCON(0) = 1, this bit is set when any compare trip is low and is also enabled.
EVIMRA Register Bit Changes		
0	PDPINT	PDPINT Enable: The definition of this bit depends on EXTCON(0). When EXTCON(0) = 0, the definition remains the same as 240x, i.e., this bit enables and disables both PDP interrupt and the direct path of PDPINT pin to compare output buffers. When EXTCON(0) = 1, this bit becomes just a PDP interrupt enable and disable bit.

EXTCONx is an added control register to enable and disable the added/modified features; therefore, the entire set of bit descriptions is new. See the register in Figure 5−29 on page 5-40 for the descriptions. Figure 5−30 and Figure 5−31 show the Hi-Z control using the EXTCONx register.

Figure 5−30. EXTCONx Register Bit Controls for PWM Hi-Z Control



Note: This diagram is a logical representation of Hi-Z control and does not reflect the actual circuit in a specific device.

Control Sequence	INDOE	C1TRIPE	C1TRIP Pin	FCMP1OE Bit	PDPINT Flag Only	Hi-Z Control	PWMs
EXTCONx bits enabled for individual PWMs control	1	1	1	1	0	1	PWM signal
Low pulse on the C1TRIP pin†	1	1	⊓				
FCMP1OE is cleared for Hi-Z enable	1	1	1	0	1	0	Hi-Z
Set FCMP1OE = 1 to remove Hi-Z control	1	1	1	1	0	1	PWM signal

Notes: 1) The shaded cells in the table show the changes affected due to low pulse on the T1CTRIP pin.

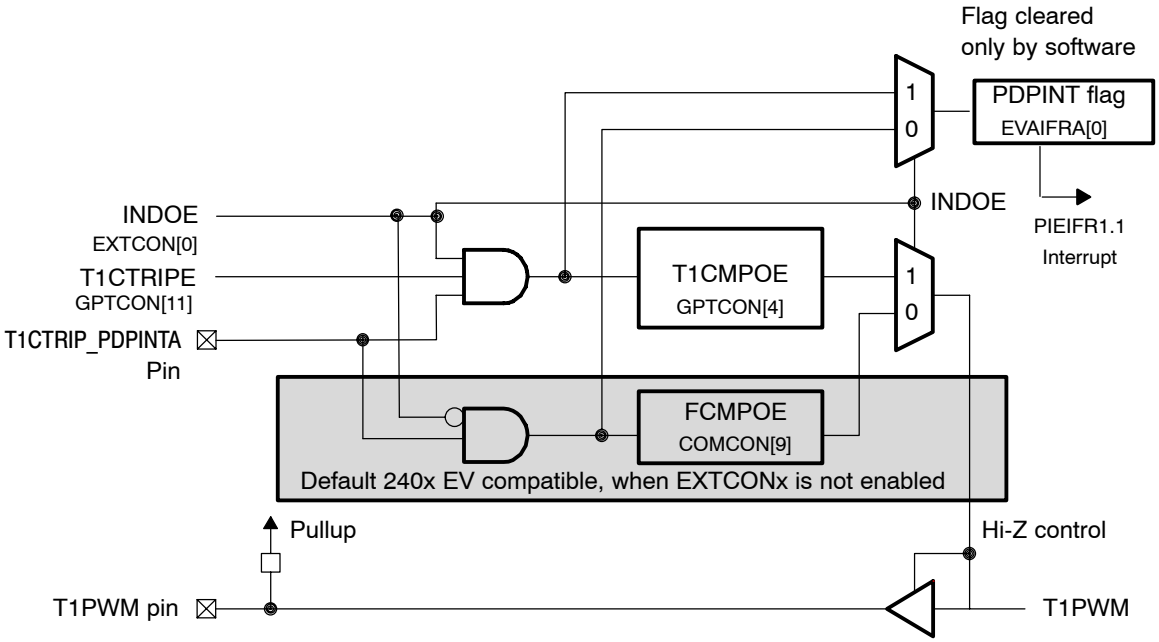
2) FCMPOE This is active in 240x EV-compatible mode when EXTCON_bit0_INDOE = 0
This is a single bit that controls high-impedance (Hi-Z) mode for all the PWM pairs:
EVA − PWM1/2, PWM 3/4, PWM 5/6, T1/T2 PWM.

3) FCMP1OE This is active in the enhanced mode for the EV when EXTCON_bit0_INDOE = 1
This bit controls high-impedance mode only for the PWM 1/2 pair
FCMP2OE, FCMP3OE control PWM 3/4, PWM 5/6 pairs
EVB has similar independent PWM high-impedance mode control in its register set.

4) T1CTRIP_PDPINTA trip control alone has the direct control path to PWM Hi-Z control buffers and the FCMPOE bit control logic. C1TRIP/C2TRIP/C3TRIP pins do not have direct control path to the Hi-Z buffers. They all go through to their respective FCMPxOE bits.

† Pulse width is based on the input qualifier on this pin.

Figure 5–31. EXTCONx Register Bit Controls for T1/T2 PWM Hi-Z Control



Note: This diagram is a logical representation of Hi-Z control and does not reflect the actual circuit in a specific device.

Control Sequence	INDOE	T1CTRIPE	T1CTRIP Pin	T1CMPOE Bit	PDPINT Flag Only	Hi-Z Control	PWMs
EXTCONx bits enabled for individual PWMs control	1	1	1	1	0	1	T1PWM signal
Low pulse on the T1CTRIP pin†	1	1	⊓				
T1CMPOE is cleared for high'-impedance (Hi-Z) enable	1	1	1	0	1	0	Hi-Z
Set T1CMPOE = 1 to remove Hi-Z control	1	1	1	1	0	1	T1PWM signal

Notes: 1) The shaded cells in the table show the changes affected due to low pulse on the T1CTRIP pin.

2) FCMPOE This is active in 240x™ EV-compatible mode when EXTCON_bit0_INDOE = 0
This is a single bit that controls high-impedance mode for all the PWM pairs:
EVA – PWM1/2, PWM 3/4, PWM 5/6, T1/T2 PWM.

3) T1CMPOE This is active in the enhanced mode for the EV when EXTCON_bit0_INDOE = 1
This bit controls high-impedance mode only for the T1PWM pin
T1CMPOE, T2CMPOE control T1PWM, T2PWM pins
EVB has similar independent T3PWM/T4PWM high-impedance mode control in its register set.

† Pulse width is based on the input qualifier on this pin.

240x is a trademark of Texas Instruments.

This page intentionally left blank

Appendix A

Revision History

This document was revised to SPRU065B from SPRU065A, which was released in July 2003. The scope of the revisions was limited to technical changes as described in A.1. This appendix lists only revisions made in the most recent version.

A.1 Changes Made in This Revision

The following changes were made in this revision:

Page	Additions/Modifications/Deletions
1-34	Changed T2PR and T2CON to 3 for EVB in Section 1.6.
5-12	Changed definition of bit 6 of the Compare Control A Register (Figure 5−7)
5-20	Changed name of bits 14−13 in the CAPCONB Register (Figure 5−12)
B-1	Updated Registers in the register layout summary in Appendix B

Appendix B

EV Register Summary

Figure B–1. Timer x Counter Register (TxCNT, where x = 1, 2, 3, or 4)

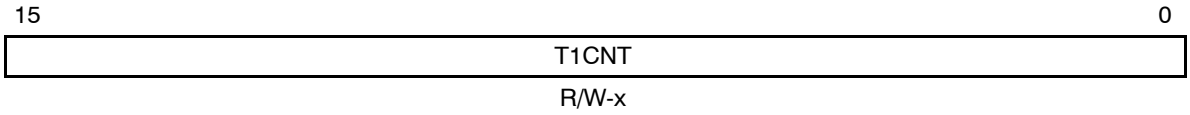


Figure B–2. Timer x Compare Register (TxCMPR, where x = 1, 2, 3, or 4)

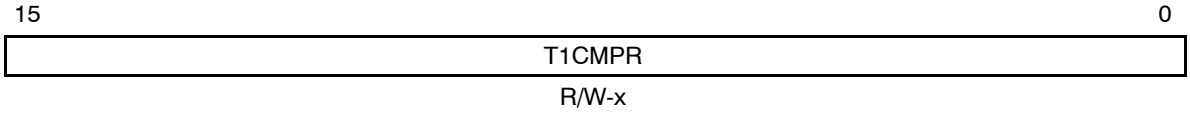


Figure B–3. Timer x Period Register (TxPR, where x = 1, 2, 3, or 4)

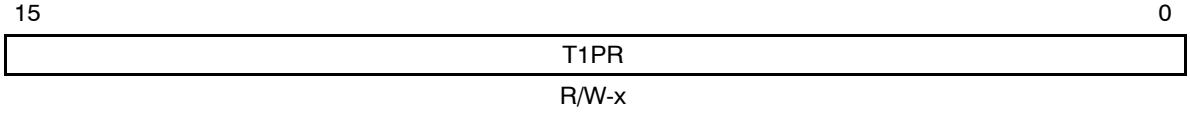


Figure B–4. Timer x Control Register (TxCON; x = 1, 2, 3, or 4)

15	14	13	12	11	10	9	8
Free	Soft	Reserved	TMODE1	TMODE0	TPS2	TPS1	TPS0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
T2SWT1/ T4SWT3†	TENABLE	TCLKS1	TCLKS0	TCLD1	TCLD0	TECMPR	SELT1PR/ SELT3PR†
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Legend: R = Read access, W = Write access, -0 = value after reset
† Reserved in T1CON and in T3CON

Figure B–5. GP Timer Control Register A (GPTCONA) — Address 7400h

15	14	13	12	11	10	9	8
Reserved	T2STAT	T1STAT	T2CTRIPE	T1CTRIPE	T2TOADC		T1TOADC
R-0	R-1	R-1	R/W-1	R/W-1	R/W-0		R/W-0

7	6	5	4	3	2	1	0
T1TOADC	TCMPOE	T2CMPOE	T1CMPOE	T2PIN		T1PIN	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

Note: R = Read access, W = Write access, -n = value after reset

Figure B–6. GP Timer Control Register B (GPTCONB) — Address 7500h

15	14	13	12	11	10	9	8
Reserved	T4STAT	T3STAT	T4CTRIPE	T3CTRIPE	T4TOADC		T3TOADC
R/W-0	R-1	R-1	R/W-1	R/W-1	R/W-0		R/W-0

7	6	5	4	3	2	1	0
T3TOADC	TCMPOE	T4CMPOE	T3CMPOE	T4PIN		T3PIN	
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0		R/W-0	

Figure B–7. Compare Control A (COMCONA) Register — Address 7411h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCMPOE	$\overline{\text{PDPINTA}}$ Status
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
FCMP3OE	FCMP2OE	FCMP1OE	Reserved		C3TRIPE	C2TRIPE	C1TRIPE
R/W-0	R/W-0	R/W-0	R-0		R/W-1	R/W-1	R/W -1

Figure B–8. Compare Control B (COMCONB) Register — Address 7511h

15	14	13	12	11	10	9	8
CENABLE	CLD1	CLD0	SVENABLE	ACTRLD1	ACTRLD0	FCMPOE	$\overline{\text{PDPINTB}}$ Status
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R-0

7	6	5	4	3	2	1	0
FCMP6OE	FCMP5OE	FCMP4OE	Reserved		C6TRIPE	C5TRIPE	C4TRIPE
R/W-0	R/W-0	R/W-0	R-0		R/W-1	R/W-1	R/W -1

Legend: R = Read, W = Write, -n = reset value

Note: Shaded areas indicate that the bit is active only when the EXTCONA bit 0 = 1.

Figure B–9. Compare Action Control Register A (ACTRA) — Address 7413h

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP6ACT1	CMP6ACT0	CMP5ACT1	CMP5ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
CMP4ACT1	CMP4ACT0	CMP3ACT1	CMP3ACT0	CMP2ACT1	CMP2ACT0	CMP1ACT1	CMP1ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Figure B–10. Compare Action Control Register B (ACTRB) — Address 7513h

15	14	13	12	11	10	9	8
SVRDIR	D2	D1	D0	CMP12ACT1	CMP12ACT0	CMP11ACT1	CMP11ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
CMP10ACT1	CMP10ACT0	CMP9ACT1	CMP9ACT0	CMP8ACT1	CMP8ACT0	CMP7ACT1	CMP7ACT0
RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0	RW-0

Note: R = Read access, W = Write access, -0 = value after reset

Figure B–11. Capture Control Register A (CAPCONA) — Address 7420h

15	14	13	12	11	10	9	8
CAPRES	CAP12EN		CAP3EN	Reserved	CAP3TSEL	CAP12TSEL	CAP3TOADC
RW-0	RW-0		RW-0	RW-0	RW-0	RW-0	RW-0

7	6	5	4	3	2	1	0
CAP1EDGE		CAP2EDGE		CAP3EDGE		Reserved	
RW-0		RW-0		RW-0		RW-0	

Note: R = Read access, W = Write access, -0 = value after reset

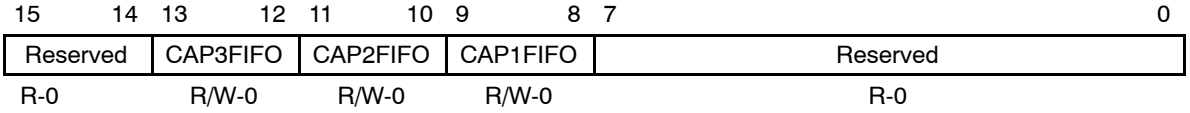
Figure B–12. Capture Control Register B (CAPCONB) — Address 7520h

15	14	13	12	11	10	9	8
CAPRES	CAP45EN		CAP6EN	Reserved	CAP6TSEL	CAP45TSEL	CAP6TOADC
R/W-0	R/W-0		R/W-0	R-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
CAP4EDGE		CAP5EDGE		CAP6EDGE		Reserved	
R/W-0		R/W-0		R/W-0		R/W-0	

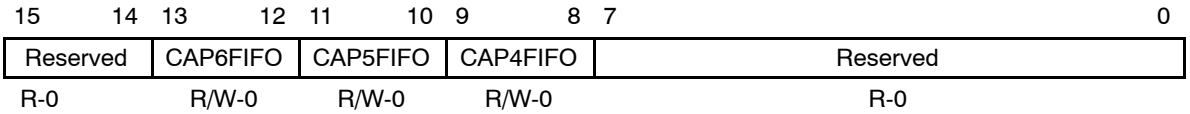
Note: R = Read access, W = Write access, -0 = value after reset

Figure B−13. Capture FIFO Status Register A (CAPFIFOA) — Address 7422h



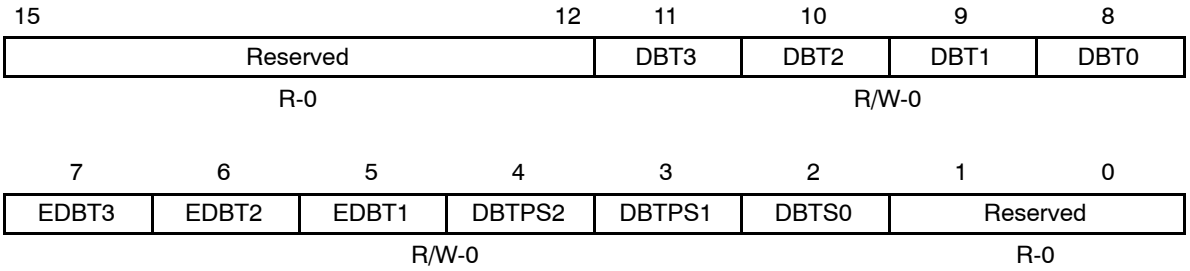
Note: R = Read access, W = Write access, -0 = value after reset

Figure B−14. Capture FIFO Status Register B (CAPFIFOB) — Address 7522h



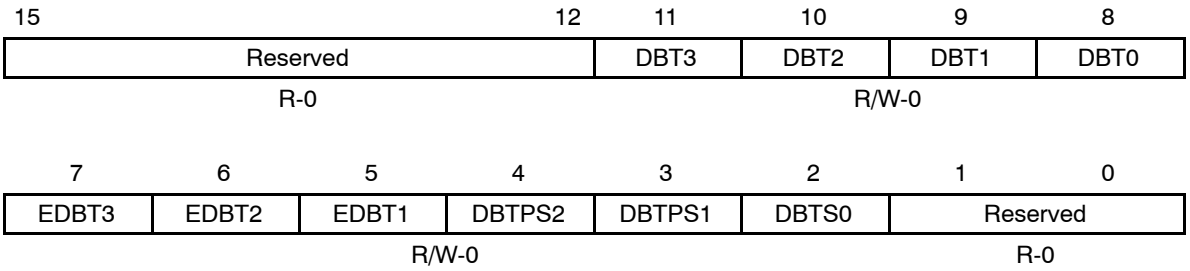
Note: R = Read access, W = Write access, -0 = value after reset

Figure B−15. Dead-Band Timer Control Register A (DBTCONA) — Address xx15h



Note: R = Read access, W = Write access, -0 = value after reset

Figure B−16. Dead-Band Timer Control Register B (DBTCONB) — Address xx15h



Note: R = Read access, W = Write access, -0 = value after reset

Figure B–17. EVA Interrupt Flag Register A (EVAIFRA) — Address 742Fh

15					10	9	8
Reserved					TIOFINT FLAG	T1UFINT FLAG	T1CINT FLAG
R-0					R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
T1PINT FLAG	Reserved			CMP3INT FLAG	CMP2INT FLAG	CMP1INT FLAG	PDPINTA FLAG
R/W-0	R-0			R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, -n = value after reset

Figure B–18. EVA Interrupt Flag Register B (EVAIFRB) — Address 7430h

15							8
Reserved							
R/W-0							

7			4	3	2	1	0
Reserved				T2OFINT FLAG	T2UFINT FLAG	T2CINT FLAG	T2PINT FLAG
R-0				RW1C-0	RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

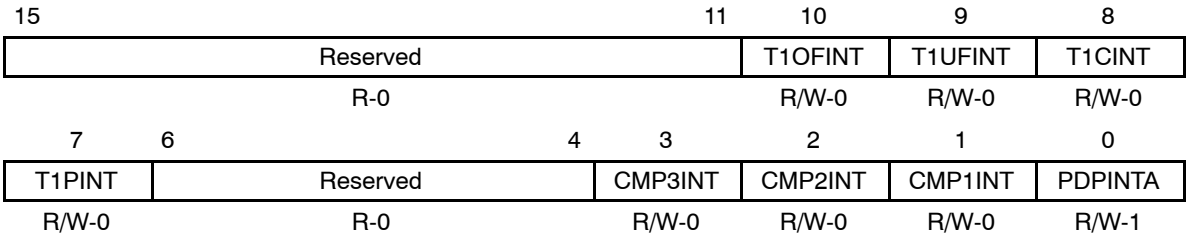
Figure B–19. EVA Interrupt Flag Register C (EVAIFRC) — Address 7431h

15							8
Reserved							
R-0							

7				3	2	1	0
Reserved					CAP3FINT FLAG	CAP2FINT FLAG	CAP1FINT FLAG
R-0					RW1C-0	RW1C-0	RW1C-0

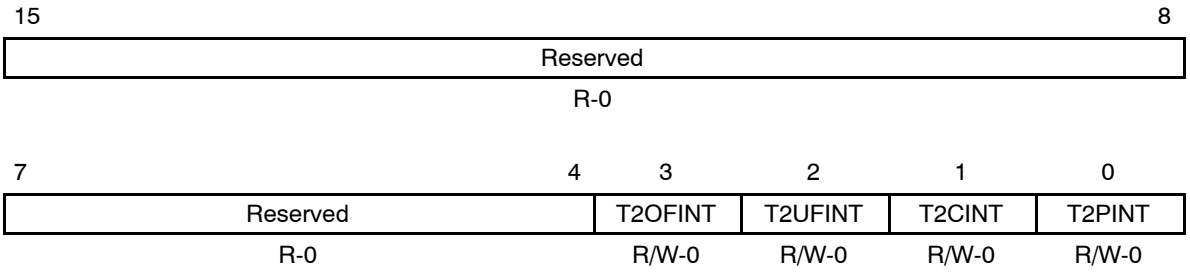
Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Figure B–20. EVA Interrupt Mask Register A (EVAIMRA) — Address 742Ch



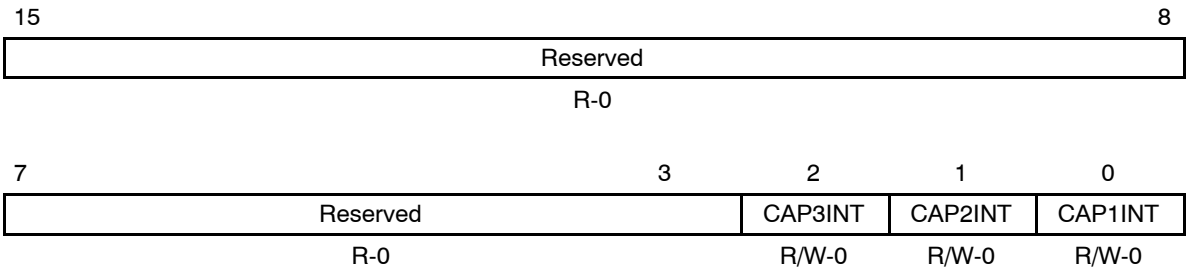
Note: R = Read access, W = write access, -n = value after reset

Figure B–21. EVA Interrupt Mask Register B (EVAIMRB) — Address 742Dh



Note: R = Read access, W = Write access, -0 = value after reset

Figure B–22. EVA Interrupt Mask Register C (EVAIMRC) — Address 742Eh



Note: R = Read access, W = Write access, -0 = value after reset

Figure B–23. EVB Interrupt Flag Register A (EVBIFRA) — Address 752Fh

15 – 11	10	9	8
Reserved	T3OFINT FLAG	T3UFINT FLAG	T3CINT FLAG
R-0	RW1C-0	RW1C-0	RW1C-0

7	6 – 4	3	2	1	0
T3PINT FLAG	Reserved	CMP6INT	CMP5INT	CMP4INT	PDPINTB
RW1C-0	R-0	RW1C-0	RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Figure B–24. EVB Interrupt Flag Register B (EVBIFRB) — Address 7530h

15 – 8
Reserved
R-0

7 – 4	3	2	1	0
Reserved	T4OFINT FLAG	T4UFINT FLAG	T4CINT FLAG	T4PINT FLAG
R-0	RW1C-0	RW1C-0	RW1C-0	RW1C-0

Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

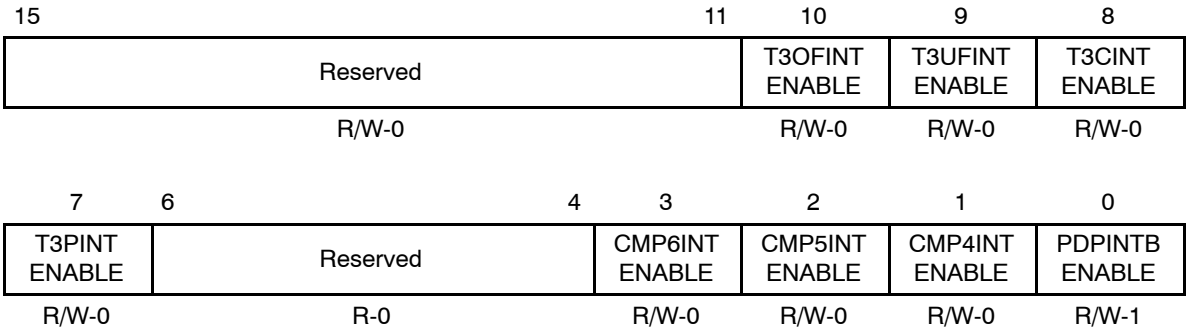
Figure B–25. EVB Interrupt Flag Register C (EVBIFRC) — Address 7531h

15 – 8
Reserved
R-0

7 – 3	2	1	0
Reserved	CAP6INT FLAG	CAP5INT FLAG	CAP4INT FLAG
R-0	RW1C-0	RW1C-0	RW1C-0

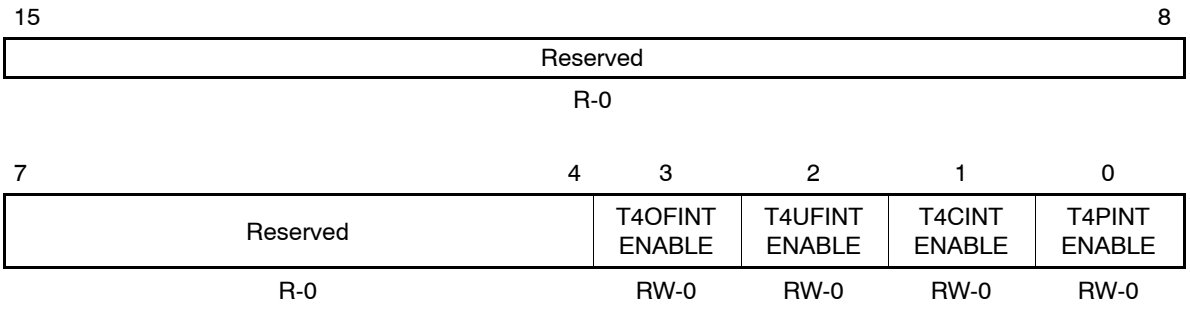
Note: R = Read access, W1C = Write 1 to clear, -0 = value after reset

Figure B–26. EVB Interrupt Mask Register A (EVBIMRA) — Address 752Ch



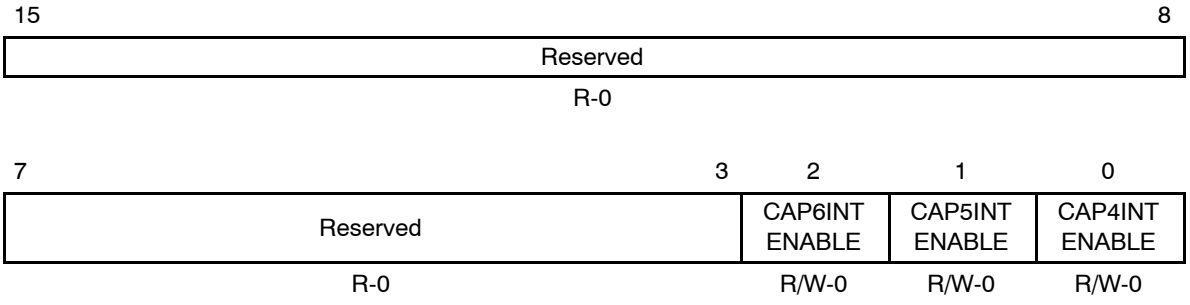
Note: R = Read access, W = Write access, -n = value after reset

Figure B–27. EVB Interrupt Mask Register B (EVBIMRB) — Address 752Dh



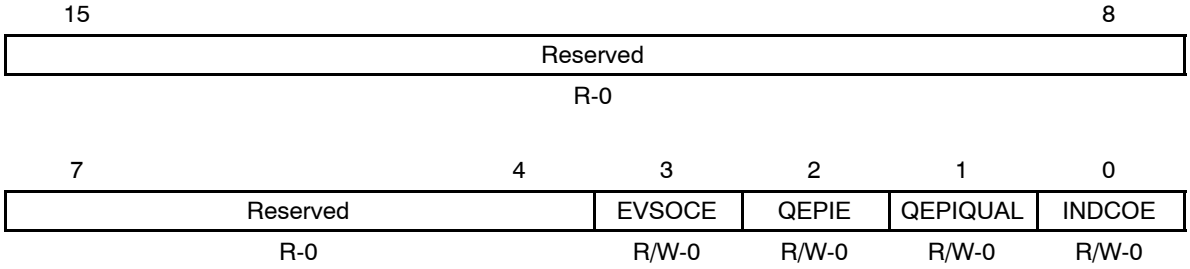
Note: R = Read access, W = Write access, -0 = value after reset

Figure B–28. EVB Interrupt Mask Register C (EVBIMRC) — Address 752Eh



Note: R = Read access, W = Write access, -n = value after reset

Figure B–29. EV Extension Control Register A (EXTCONA) — Address 7409h



This page intentionally left blank.

Index

A

B

C

D

E

Q

R

T

TMS320x281x DSP System Control and Interrupts Reference Guide

Literature Number: SPRU078C
April 2002 – Revised March 2005

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Preface

Read This First

About This Manual

This guide describes how various 281x digital signal processor (DSP) system controls and interrupts work with peripherals. It includes information on the:

- ❑ Flash and one-time programmable (OTP) memories
- ❑ Code security module (CSM), which is a security feature incorporated in 28x devices
- ❑ Clocking mechanisms including the oscillator, PLL, the watchdog function, and the low-power modes
- ❑ GPIO MUX registers used to select the operation of shared pins
- ❑ Peripheral frames and the device emulation registers
- ❑ Peripheral interrupt expansion (PIE) block that multiplexes numerous interrupt sources into a smaller set of interrupt inputs

Related Documentation From Texas Instruments

The following books describe the TMS320x281x and related support tools that are available on the TI website.

TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, and TMS320C2812 Digital Signal Processors (literature number SPRS174) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320R2811 and TMS320R2812 Digital Signal Processors (literature number SPRS257) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C28x DSP CPU and Instruction Set Reference Guide (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x™ fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320x281x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU060) describes the ADC module. The module is a 12−bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample−and−hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x281x Boot ROM Reference Guide (literature number SPRU095) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide (literature number SPRU074) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

TMS320x281x Event Manager (EV) Reference Guide (literature number SPRU065) describes the EV modules that provide a broad range of functions and features that are particularly useful in motion control and motor control applications. The EV modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits.

TMS320x281x External Interface (XINTF) Reference Guide (literature number SPRU067) describes the external interface (XINTF) of the 28x digital signal processors (DSPs).

TMS320x281x Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU061) describes the McBSP) available on the C28x devices. The McBSPs allow direct interface between a DSP and other devices in a system.

TMS320x281x, 280x Peripheral Reference Guide (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide (literature number SPRU051) describes the SCI that is a two-wire

asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

TMS320x281x, 280x Serial Peripheral Interface (SPI) Reference Guide (literature number SPRU059) describes the SPI – a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit–transfer rate. The SPI is used for communications between the DSP controller and external peripherals or another controller.

TMS320x281x System Control and Interrupts Reference Guide (literature number SPRU078) describes the various interrupts and system control features of the 281x digital signal processors (DSPs).

The TMS320C28x Instruction Set Simulator Technical Overview (literature number SPRU608) describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU625) describes development using DSP/BIOS.

3.3 V DSP for Digital Motor Control Application Report (literature number SPRA550). New generations of motor control digital signal processors (DSPs) lower their supply voltages from 5 V to 3.3 V to offer higher performance at lower cost. Replacing traditional 5-V digital control circuitry by 3.3-V designs introduce no additional system cost and no significant complication in interfacing with TTL and CMOS compatible components, as well as with mixed voltage ICs such as power transistor gate drivers. Just like 5-V based designs, good engineering practice should be exercised to minimize noise and EMI effects by proper component layout and PCB design when 3.3-V DSP, ADC, and digital circuitry are used in a mixed signal environment, with high and low voltage analog and switching signals, such as a motor control system. In addition, software techniques such as Random PWM method can be used by special features of the Texas Instruments (TI) TMS320x24xx DSP controllers to significantly reduce noise effects caused by EMI radiation.

This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications. The application report first de-

scribes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V – 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system's noise and EMI effects are summarized in the last section.

Thermo-Electric Cooler Control Using a TMS320F2812 DSP & DRV592 Power Amplifier Application Note (literature number SPRA873). This application report presents a thermoelectric cooler system consisting of a Texas Instruments TMS320F2812 digital signal processor (DSP) and DRV592 power amplifier. The DSP implements a digital proportional-integral-derivative feedback controller using an integrated 12-bit analog-to-digital converter to read the thermistor, and direct output of pulse-width-modulated waveforms to the H-bridge DRV592 power amplifier. The system presented provides up to 6.1 watts of heating or cooling to the laser mount, although the DRV592 amplifier is actually capable of delivering up to 15 watts when configured appropriately. The closed-loop TEC system is seen to achieve ±0.0006°C temperature accuracy, depending on the needed operating temperature range, with a step response settling time of 14 to 16 seconds. A complete description of the experimental system, along with software and software operating instructions, are provided.

Running an Application from Internal Flash Memory on the TMS320F281x DSP Application Report (literature number SPRA958). Several special requirements exist for running an application from on-chip flash memory on the TMS320F28x DSP. These requirements generally do not manifest themselves during development in RAM since the Code Composer Studio™ debugger can mask problems associated with initialized sections and how they are linked to memory. This application report covers the requirements needed to properly configure application software for execution from on-chip flash memory. Requirements for both DSP/BIOS™ and non-DSP/BIOS projects are presented. Some performance considerations and techniques are also discussed. Example code projects are included that run from on-chip flash on the eZdsp™ F2812 development board (or alternately any F2812, F2811, or F2810 DSP board). Code examples that run from internal RAM are also provided for completeness. These code examples provide a starting point for code development, if desired.

Trademarks

Code Composer Studio and C28x are trademarks of Texas Instruments.

This page intentionally left blank.

Contents

Figures

Tables

Chapter 1

Memory

This chapter describes how Flash and one-time programmable (OTP) memories can be used with the 28x digital signal processor (DSP) device and peripherals. It also includes the registers associated with memory.

Topic **Page**

1.1 Flash and OTP Memory

This section describes how to configure two kinds of memory – Flash and one-time programmable (OTP).

1.1.1 Flash Memory

The on-chip Flash in Flash devices is uniformly mapped in both program and data memory space. This Flash memory is always enabled on 28x devices and features:

- ❑ Multiple sectors
- ❑ Code security
- ❑ Low power modes
- ❑ Configurable waitstates that can be adjusted based on CPU frequency
- ❑ Flash pipeline mode for improved performance of linear code execution

1.1.2 OTP Memory

The 1K $\times$ 16 block of one-time programmable (OTP) memory can be used to program data or code. This block can be programmed one time and cannot be erased.

1.2 Flash and OTP Power Modes

The following operating states apply to the Flash and OTP memory:

- Reset or Sleep State: This is the state after a device reset. In this state, the bank and pump are in a sleep state (lowest power). A CPU read or fetch accesses to the Flash/OTP memory map area stalls the CPU. This access automatically initiates a change in power modes to the active or read state.
- Standby State: In this state, the bank and pump are in standby power mode state. A CPU read or fetch accesses to the Flash/OTP memory map area stalls the CPU. This access automatically initiates a change in power modes to the active state.
- Active or Read State: In this state, the bank and pump are in active power mode state (highest power). The CPU read or fetch access wait states to the Flash/OTP memory map area is controlled by the FBANKWAIT and FOTPWAIT registers. A prefetch mechanism called Flash pipeline can also be enabled for improving fetch performance for linear code execution.

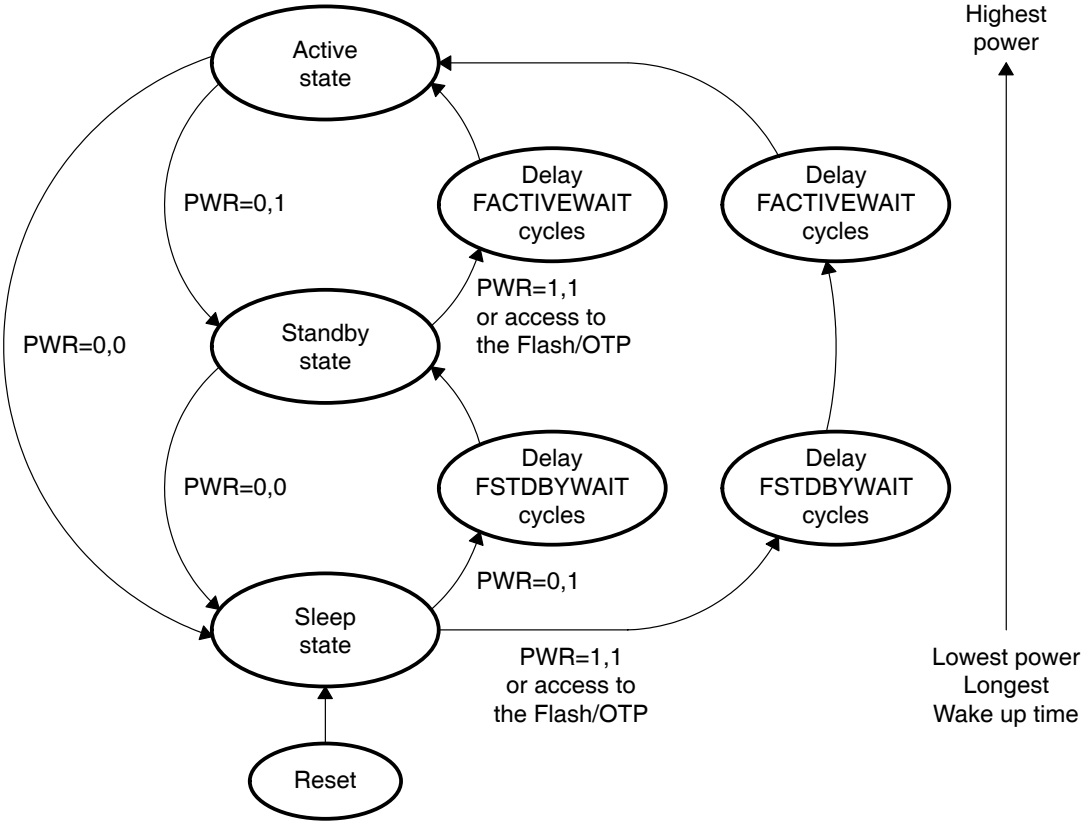
The Flash/OTP bank and pump are always in the same power mode during read or execution operations from the Flash/OTP. See Figure 1–1 for a graphic depiction of the states.

You can change the current Flash/OTP memory power state as follows:

- To move to a lower power state: Change the PWR mode bits from a higher power mode to a lower power mode. This change instantaneously moves the Flash/OTP bank to the lower power state. This register should be accessed only by code running outside the Flash/OTP memory.
- To move from a lower power state to a higher power state, there are two options:
  - Change the FPWR register from a lower state to a higher state. This access brings the Flash/OTP memory to the higher state.
  - Access the Flash or OTP memory by a read access or program fetch access. This access automatically brings the Flash/OTP memory to the active state.

There is a delay when moving from a lower power state to a higher one. See Figure 1–1. This delay is required to allow the Flash to stabilize at the higher power mode. If any access to the Flash/OTP memory occurs during this delay the CPU automatically stalls until the delay is complete.

Figure 1–1. Flash Power Mode State Diagram



The duration of the delay is determined by the FSTDBYWAIT and FACTIVEWAIT registers. Moving from the sleep state to a standby state is delayed by a count determined by the FSTDBYWAIT register. Moving from the standby state to the active state is delayed by a count determined by the FACTIVEWAIT register. Moving from the sleep mode (lowest power) to the active mode (highest power) is delayed by FSTDBYWAIT + FACTIVEWAIT.

1.2.1 Flash and OTP Performance

CPU read or fetch operations to the Flash/OTP can take one of the following forms:

- ❏ 32-bit instruction fetch
- ❏ 16 or 32-bit data space read
- ❏ 16-bit program space read

Once Flash is in the active power state, then a read or fetch access to the bank memory map area can be classified as three types:

- Flash Memory Random Access: The number of wait states, for a random access, is configured by the RANDWAIT bits in the FBANKWAIT register. This register defaults to a worst-case count and the user needs to program the appropriate number of wait states to improve performance based on the CPU clock rate and the access time of the Flash.
- Flash Memory Paged Access: The Flash array is organized into rows and columns. The rows contain 2048 bits of information. The first access to a row is considered a random access. Subsequent accesses within the same row can be made with a faster access time. This is termed a PAGE access.

  The Flash can take advantage of this by the user configuring a lower number of wait states in the PAGEWAIT bits in the FBANKWAIT register. This mode works for data space and program space reads as well as instruction fetches. See the device data sheet for more information on the access time of the Flash.
- OTP Access: Read or fetch accesses to the OTP are controlled by the OTPWAIT register bits in the FOTPWAIT register. Accesses to the OTP take longer than the Flash and there is no paged mode.

Notes: 1) Writes to the Flash/OTP memory map area are ignored. They complete in a single cycle.
2) When the Code Security Module (CSM) is secured, reads to the Flash/OTP memory map area from outside the secure zone take the same number of cycles as a normal access. However, the read operation returns a zero.
3) The Flash supports 0-wait accesses when the PAGEWAIT bits are set to zero. This assumes that the CPU speed is low enough to accommodate the access time.

1.2.2 28x Flash Pipeline Mode

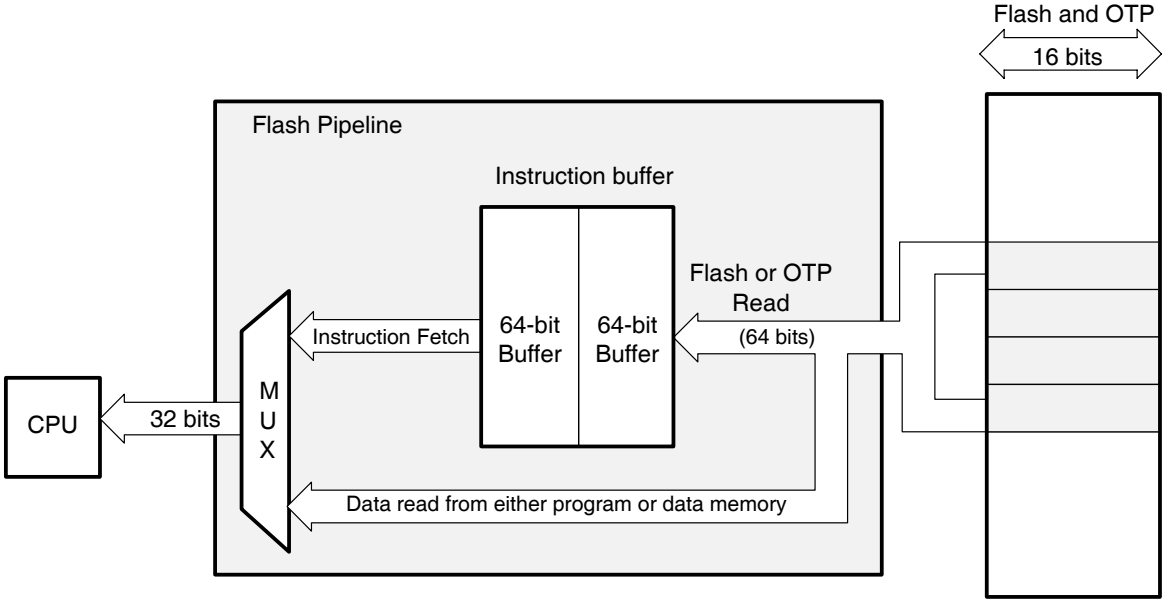
Flash memory is typically used to store application code. During code execution, instructions are fetched from sequential memory addresses, except when a discontinuity occurs. Usually the portion of the code that resides in sequential addresses makes up the majority of the application code and is referred to as linear code. To improve the performance of linear code execution, a Flash pipeline mode has been implemented. Setting the ENPIPE bit in the FOPT register enables this mode. The Flash pipeline mode is independent of the CPU pipeline. To allow you to maintain code timing compatibility between Flash and ROM devices, the Flash pipeline mode has also been implemented on ROM devices.

An instruction fetch from the Flash or OTP reads out 64 bits per access. The starting address of the access from Flash is automatically aligned to a 64-bit

boundary such that the instruction location is within the 64 bits to be fetched. With Flash pipeline mode enabled (see Figure 1–2), the 64 bits read from the instruction fetch are stored in a 64-bit wide by 2-level deep instruction pre-fetch buffer. The contents of this pre-fetch buffer are then sent to the CPU for processing as required.

Up to two 32-bit instructions or up to four 16-bit instructions can reside within a single 64-bit access. The majority of C28x instructions are 16 bits, so for every 64-bit instruction fetch from the Flash bank it is likely that there are up to four instructions in the pre-fetch buffer ready to process through the CPU. During the time it takes to process these instructions, the Flash pipeline automatically initiates another access to the Flash bank to pre-fetch the next 64 bits. In this manner, the Flash pipeline mode works in the background to keep the instruction pre-fetch buffers as full as possible. Using this technique, the overall efficiency of sequential code execution from Flash or OTP is improved significantly.

Figure 1–2. Flash Pipeline



The Flash pipeline pre-fetch is aborted only on a PC discontinuity caused by executing an instruction such as a branch, BANZ, call, or loop. When this occurs, the pre-fetch is aborted and the contents of the pre-fetch buffer are flushed. There are two possible scenario's when this occurs:

- ❑ If the destination address is within the Flash or OTP, the pre-fetch aborts and then resumes at the destination address.

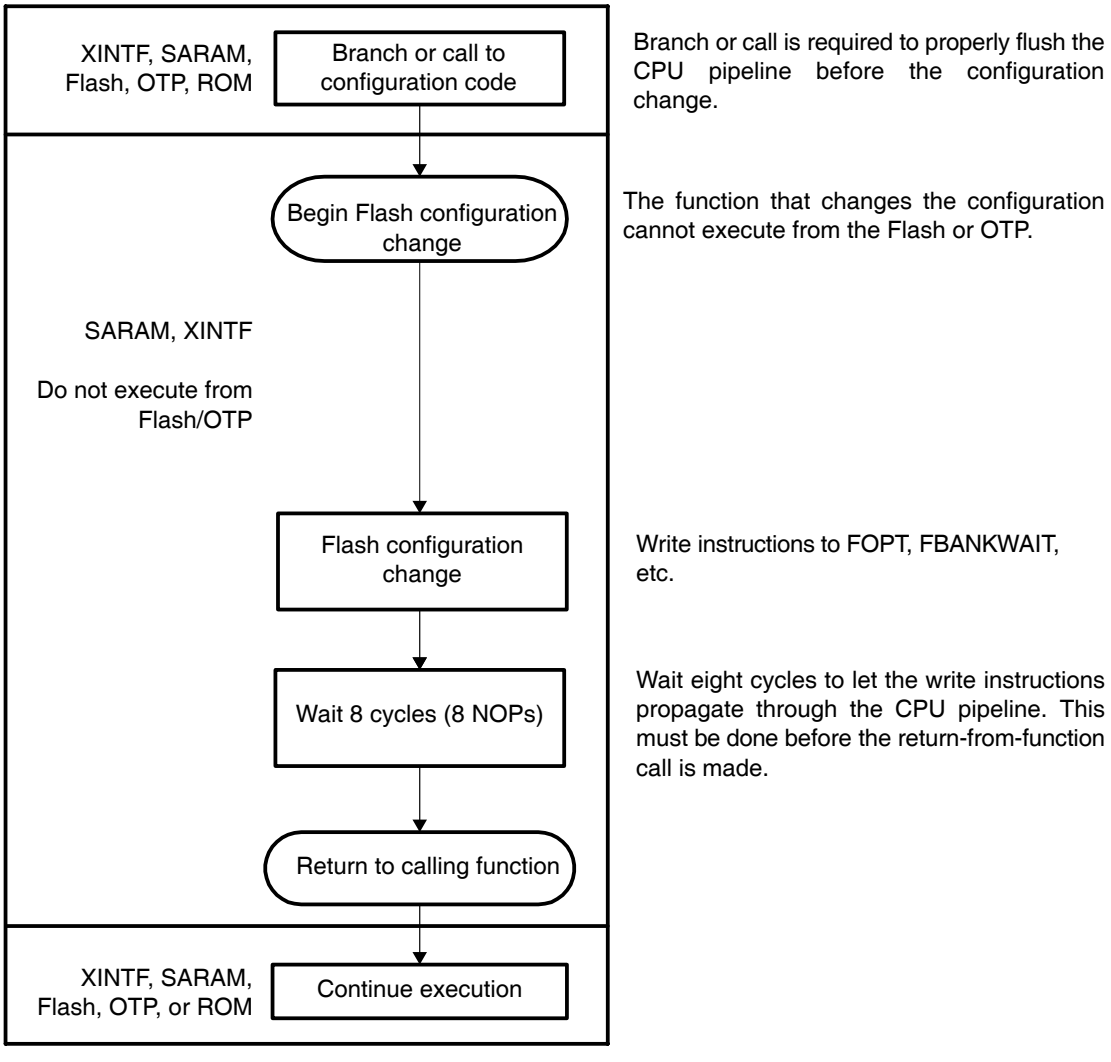
- If the destination address is outside of the Flash and OTP, the pre-fetch is aborted and begins again only when a branch is made back into the Flash or OTP. The Flash pipeline pre-fetch mechanism only applies to instruction fetches from program space. Data reads from data memory and from program memory do not utilize the pre-fetch buffer capability and thus bypass the pre-fetch buffer. For example, instructions such as MAC, DMAC, and PREAD read a data value from program memory. When this read happens, the pre-fetch buffer is bypassed but the buffer is not flushed. If an instruction pre-fetch is already in progress when a data read operation is initiated, then the data read will be stalled until the pre-fetch completes.

1.2.3 Procedure to Change the Flash Configuration Registers

During Flash configuration, no accesses to the Flash or OTP can be in progress. This includes instructions still in the CPU pipeline, data reads, and instruction pre-fetch operations. To be sure that no access takes place during the configuration change, you should follow the procedure shown in Figure 1–3 for any code that modifies the FOPT, FPWR, FBANKWAIT, or FOTPWAIT registers.

This procedure also applies to the ROM on devices where the Flash and OTP have been replaced with ROM.

Figure 1–3. Flash Configuration Access Flow Diagram



1.3 Flash and OTP Registers

The Flash and OTP memory can be configured by the registers shown in Table 1–1. The bit descriptions are in Figure 1–4 through Figure 1–6.

Table 1–1. Flash/OTP Configuration Registers(1)

Name	Address	Size (x16)	Description
		Configuration Registers	
FOPT	0x0000–0A80	1	Flash Option Register
Reserved	0x0000–0A81	1	Reserved
FPWR	0x0000–0A82	1	Flash Power Modes Register
FSTATUS	0x0000–0A83	1	Status Register
FSTDBYWAIT	0x0000–0A84	1	Flash Sleep To Standby Wait Register
FACTIVEWAIT	0x0000–0A85	1	Flash Standby To Active Wait Register
FBANKWAIT	0x0000–0A86	1	Flash Read Access Wait State Register
FOTPWAIT	0x0000–0A87	1	OTP Read Access Wait State Register

1) These registers are EALLOW protected.

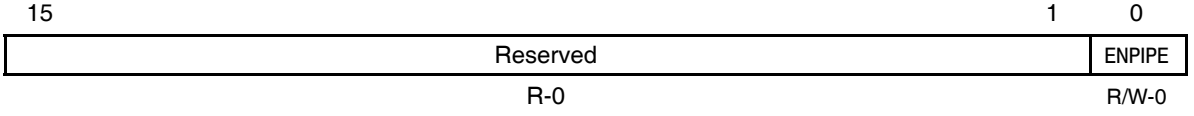
Note: Flash configuration registers should not be accessed while an access is in progress in Flash or OTP memory

The Flash configuration registers should not be accessed from code that is running from OTP or Flash memory or while an access may be in progress. All register accesses to the Flash registers should be made from code executing outside of Flash/OTP memory and an access should not be attempted until all activity on the Flash/OTP has completed. No hardware is included to protect against this.

You can read the Flash registers from code executing in Flash/OTP; however, do not write to the registers.

CPU write access to the registers can be enabled only by executing the EALLOW instruction. Write access is disabled when the EDIS instruction is executed. This protects the registers from spurious accesses. Read access is always available. The registers can be accessed through the JTAG port without the need to execute EALLOW. These registers support both 16-bit and 32-bit accesses.

Figure 1–4. Flash Options (FOPT) Register



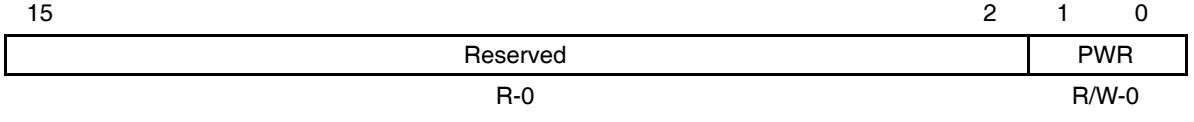
Legend: R = Read access, -0 = value after reset

Note: EALLOW-protected register

Table 1–2. Flash Options (FOPT) Register Field Descriptions

Bits	Field	Description
15–1	Reserved	
0	ENPIPE	Enable Flash Pipeline Mode Bit: Flash pipeline mode is active when this bit is set. The pipeline mode improves performance of instruction fetches by pre-fetching instructions. On Flash devices, ENPIPE affects fetches from Flash and OTP. On ROM devices, this bit affects fetches from the ROM blocks that replaced the Flash and OTP. When pipeline mode is enabled, the Flash waitstates (paged and random) must be greater than zero.

Figure 1–5. Flash Power Register (FPWR)



Legend: R = Read access, -0 = value after reset

Note: EALLOW-protected register

Table 1–3. Flash Power Register (FPWR) Field Descriptions

Bits	Field	Description
15–2	Reserved	
1–0	PWR	Flash Power Mode Bits: Writing to these bits changes the current power mode of the Flash bank and pump. See section 1.2 for more information on changing the Flash bank power mode. On ROM devices, changing PWR has no effect on the power consumption of the ROM. Moving to standby or sleep mode causes the next access from the ROM to be delayed just as on Flash devices. 00 Pump and bank sleep (lowest power) 01 Pump and bank standby 10 Reserved (no effect) 11 Pump and bank active (highest power)

Figure 1–6. Flash Status Register (FSTATUS)

15	9	8
Reserved		3VSTAT
R-0		R/W1C-0

7	4	3	2	1	0
Reserved		ACTIVE-WAITS	STDBY-WAITS	PWRS	
R-0		R-0	R-0	R-0	

Legend: R = read access, -0 = value after reset, W1C = write 1 to clear

Note: EALLOW-protected register

Table 1–4. Flash Status Register (FSTATUS) Field Descriptions

Bits	Field	Description
15–9	Reserved	
8	3VSTAT	V_{DD3V} Status Latch Bit: When set, this bit indicates that the 3 VSTAT signal from the pump module went to a high level. This signal indicates that the 3 V supply went out of allowable range. This bit is cleared by writing a 1, writes of 0 are ignored.
7–4	Reserved	
3	ACTIVEWAITS	Bank and Pump Standby To Active Wait Counter Status Bit: This bit indicates whether the respective wait counter is currently timing out an access. If the bit is set, then the counter is counting. If the bit is 0, then the counter is not counting.

Table 1–4. Flash Status Register (FSTATUS) Field Descriptions (Continued)

Bits	Field	Description
2	STDBYWAITS	Bank and Pump Sleep To Standby Wait Counter Status Bit: This bit indicates whether the respective wait counter is currently timing out an access. If the bit is set, then the counter is counting. If the bit is 0, then the counter is not counting.
1–0	PWRS	Power Modes Status Bits: These bits indicate the current power mode the Flash/OTP is in: 00 Pump and bank sleep (lowest power) 01 Pump and bank standby 10 Reserved 11 Pump and bank active (highest power) **Note:** The above bits only get set to the new power mode once the appropriate timing delays have expired.

Figure 1–7. Flash Standby Wait (FSTDBYWAIT) Register

15	9	8	0
Reserved		STDBYWAIT	
R-0		R/W-1	

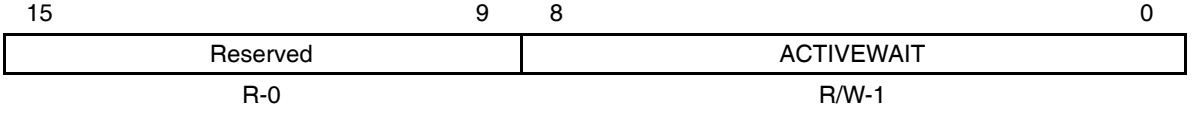
Legend: R = Read access, -0 = value after reset

Note: EALLOW-protected register

Table 1–5. Flash Standby Wait (FSTDBYWAIT) Register Field Descriptions

Bits	Field	Description
15–9	Reserved	
8–0	STDBYWAIT	Bank and Pump Sleep To Standby Wait Count. When the bank and pump modules are in sleep mode and a write is performed to the PWR bits in the FPWR register (to change to a higher default power mode) or a CPU read or fetch access is performed to the Flash bank or OTP, then a counter is initiated with the value specified in these register bits. The power mode for the bank and pump are set for standby mode. The counter then counts down to zero before the PWRS bits are set to standby mode. If a CPU read or fetch access to the Flash bank/OTP initiated the process, the CPU is stalled until the access completes (see ACTIVEWAIT bits). The STDBYWAIT bits specify the number of CPU clock cycles (0..511 SYSCLKOUT cycles) of delay. See the Flash and OTP Power Mode section for more details. This register should be left in its default state.

Figure 1–8. Flash Standby to Active Wait Counter (FACTIVEWAIT) Register



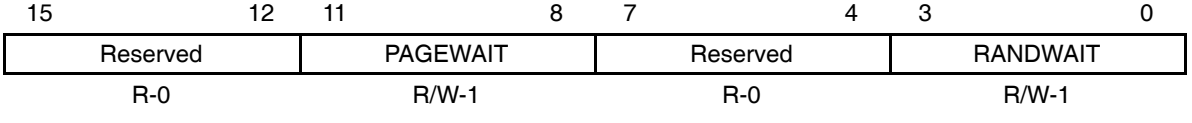
Legend: R = Read access, -0 = value after reset

Note: EALLOW-protected register

Table 1–6. Flash Standby to Active Wait Counter (FACTIVEWAIT) Register Field Descriptions

Bits	Field	Description
15–9	Reserved	
8–0	ACTIVEWAIT	Bank and Pump Standby To Active Wait Count: When the bank and pump modules are in standby mode and a write is performed to the PWR bits in the FPWR register (to change to a higher default power mode) or a CPU read or fetch access is performed to the Flash bank, then a counter is initiated with the value specified in these register bits. The power mode for the bank and pump are set for active mode. The counter then counts down to zero before allowing any CPU access to proceed. If a CPU read or fetch access to the Flash bank initiated the process, the CPU is stalled until the access completes (see PAGEWAIT and RANDWAIT bits). The ACTIVEWAIT bits specify the number of CPU clock cycles (0..511 SYSCLKOUT cycles) of delay. Refer to the Flash and OTP Power Modes section for more details. This register should be left in its default state.

Figure 1–9. Flash Waitstate (FBANKWAIT) Register



Legend: R = Read access, -0 = value after reset

Note: EALLOW-protected register

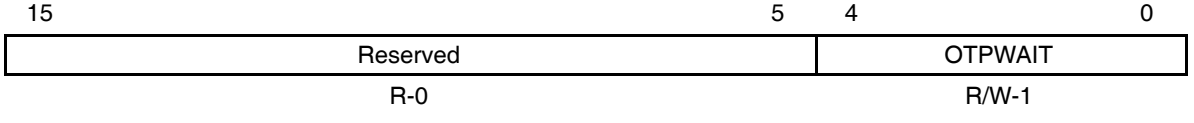
Table 1–7. Flash Waitstate (FBANKWAIT) Register Field Descriptions

Bits	Field	Description
15–12	Reserved	
11–8	PAGEWAIT	Flash Paged Read Wait States: These register bits specify the number of wait states for a paged read operation in CPU clock cycles (0..15 SYSCLKOUT cycles) to the banks. See Notes 1 and 2. On ROM devices, these bits affect the wait states of the ROM block that replaced Flash.

Table 1–7. Flash Waitstate (FBANKWAIT) Register Field Descriptions (Continued)

Bits	Field	Description
7–4	Reserved	
3–0	RANDWAIT	Flash Random Read Wait States: These register bits specify the number of wait states for a random read operation in CPU clock cycles (0..15 SYSCLKOUT cycles) to the banks. See Notes 1 and 2.RANDWAIT must be set greater than 0. That is, at least 1 random waitstate must be used. See the device-specific data sheet for the minimum time required for Flash or ROM access. On ROM devices, these bits affect the wait states of the ROM block that replaced Flash. **Notes:** 1) You must set RANDWAIT to a value greater than or equal to the PAGEWAIT setting. No hardware is provided to detect a PAGEWAIT value that is greater then RANDWAIT. 2) When enabling Flash pipeline mode, you must set PAGEWAIT and RANDWAIT to a value greater than zero.

Figure 1–10. OTP Waitstate (FOTPWAIT) Register



Legend: R = Read access, -0 = value after reset

Note: EALLOW-protected register

Table 1–8. OTP Waitstate (FOTPWAIT) Register

Bits	Field	Description
15–5	Reserved	
4–0	OTPWAIT	OTP Read Wait States. These register bits specify the number of wait states for a read operation in CPU clock cycles (0..31 SYSCLKOUT cycles) to the OTP. See CPU Read Or Fetch Access From Flash/OTP section for more details. There is no PAGE mode in the OTP. OTPWAIT must be set greater than 0. That is, a minimum of 1 wait-state must be used. See the device-specific data sheet for the minimum time required for an OTP or ROM access. On ROM devices, these bits affect the wait states of the ROM block that replaced Flash.

Chapter 2

Code Security Module (CSM)

The code security module (CSM) is a security feature incorporated in 28x devices. It prevents access/visibility to on-chip memory to unauthorized persons—i.e., it prevents duplication/reverse engineering of proprietary code.

The word secure means access to on-chip memory is protected. The word unsecure means access to on-chip secure memory is not protected — i.e., the contents of the memory could be read by any means (through a debugging tool such as Code Composer Studio™, for example).

Topic **Page**

2.1 Functional Description

The security module restricts the CPU access to certain on-chip memory. This, in effect, blocks read and write access to various memories through the JTAG port or external peripherals. Security is defined with respect to the access of on-chip memory and prevents unauthorized copying of proprietary code or data.

The device is secure when CPU access to the on-chip secure memory locations is restricted. When secure, two levels of protection are possible, depending on where the program counter is currently pointing. If code is currently running from inside secure memory, only access through JTAG is blocked (i.e., the emulator). This allows secure code to access secure data. Conversely, if code is running from nonsecure memory, all accesses to secure memories are blocked. User code can dynamically jump in and out of secure memory, thereby allowing secure function calls from nonsecure memory. Similarly, interrupt service routines can be placed in secure memory, even if the main program loop is run from nonsecure memory.

Security is protected by a password of 128-bit data (eight 16-bit words) that is used to secure or unsecure the device.

The device is unsecured by executing the password match flow (PMF), described later in this chapter. Table 2–1 shows the levels of security.

Table 2–1. Security Levels

PMF Executed With Correct Password?	Operating Mode	Program Fetch Location	Security Description
No	Secure	Outside secure memory	Only fetches are allowed to secure memory
No	Secure	Inside secure memory	CPU has full access. JTAG port cannot read the secured memory contents.
Yes	Not Secure	Anywhere	Full access for CPU and JTAG port to secure memory

Passwords are stored in code security password locations (PWL) in flash/ROM memory (0x003F 7FF8–0x003F 7FFF). These locations store the password predetermined by the system designer.

In flash devices, the password can be changed anytime if the old password is known. In ROM devices, the password cannot be changed after the device is manufactured by Texas Instruments (TI™).

If the PWL have all 128 bits as ones, the device is labeled unsecure. Since new flash devices have erased flash (all ones), only a read of the PWL is required to bring the device into unsecure mode. If the PWL have all 128 bits as zeros, the device is secure, regardless of the contents of the KEY registers. Do not use all zeros as a password or reset the device after performing a clear routine on the flash. If a device is reset when the PWL is all zeros, the device cannot be debugged or reprogrammed. To summarize, a device with an erased flash array is unsecure. A device with a cleared flash array is secure.

User accessible registers (eight 16-bit words) that are used to secure or unsecure the device are referred to as key registers. These registers are mapped in the memory space at addresses 0x0000 0AE0 – 0x0000 0AE7 and are EALLOW protected.

Note: Security of addresses between 0x3F7F80 and 0x3F7FF5

For code security operation, all addresses between 0x3F7F80 and 0x3F7FF5 cannot be used as program code or data, but must be programmed to 0x0000 when the Code Security Passwords are programmed. If security is not a concern, then these addresses can be used for code or data.

Code Security Module Disclaimer

The Code Security Module (“CSM”) included on this device was designed to password protect the data stored in the associated memory (either ROM or Flash) and is warranted by Texas Instruments (TI), in accordance with its standard terms and conditions, to conform to TI’s published specifications for the warranty period applicable for this device.

TI DOES NOT, HOWEVER, WARRANT OR REPRESENT THAT THE CSM CANNOT BE COMPROMISED OR BREACHED OR THAT THE DATA STORED IN THE ASSOCIATED MEMORY CANNOT BE ACCESSED THROUGH OTHER MEANS. MOREOVER, EXCEPT AS SET FORTH ABOVE, TI MAKES NO WARRANTIES OR REPRESENTATIONS CONCERNING THE CSM OR OPERATION OF THIS DEVICE, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL TI BE LIABLE FOR ANY CONSEQUENTIAL, SPECIAL, INDIRECT, INCIDENTAL, OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING IN ANY WAY OUT OF YOUR USE OF THE CSM OR THIS DEVICE, WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO LOSS OF DATA, LOSS OF GOODWILL, LOSS OF USE OR INTERRUPTION OF BUSINESS OR OTHER ECONOMIC LOSS.

2.2 CSM Impact on Other On-Chip Resources

The CSM has no impact whatsoever on the following on-chip resources:

- Single-access RAM (SARAM) blocks not designated as secure – These memory blocks can be freely accessed and code run from them, whether the device is in secure or unsecure mode.
- Boot ROM contents – Visibility to the boot ROM contents is not impacted by the CSM.
- On-chip peripheral registers – The peripheral registers can be initialized by code running from on-chip or off-chip memory, whether the device is in secure or unsecure mode.
- PIE vector table – Vector tables can be read and written regardless of whether the device is in secure or unsecure mode. Table 2–2 and Table 2–3 show which on-chip resources are affected (or are not affected) by the CSM on the 281x devices. For other devices, see the device-specific data sheet.
- Flash configuration registers – Registers used to configure the waitstates of the Flash and OTP are protected by the CSM. On C281x devices, these registers configure the waitstates of the ROM. To change these registers, the CSM must be unlocked or the code accessing them must be executing from secure memory.

Table 2–2. F281x/C281x Resources Affected by the CSM

Address	Block
0x0000 8000–0x0000 8FFF	L0 SARAM (4K X 16)(1)
0x0000 9000–0x0000 9FFF	L1 SARAM (4K X 16)(1)
0x003D 7800– 0x003D 7BFF	One-time programmable (OTP) or ROM (1K X 16)
0x003D 8000–0x003F 7FFF	Flash or ROM (128K X 16 or 64K X 16)

1) On R281x RAM devices, L0 and L1 SARAM are affected by the CSM. The CSM password, however, is set to all 1s so the CSM can easily be unlocked.

Table 2–3. 281x Resources Not Affected by the CSM

Address	Block
0x0000 0000–0x0000 03FF	M0 SARAM (1K X 16)
0x0000 0400–0x0000 07FF	M1 SARAM (1K X16)
0x0000 0800–0x0000 0CFF	Peripheral Frame 0 (2K X 16)
0x0000 0D00–0x0000 0FFF	PIE Vector RAM (256 X 16)
0x0000 2000–0x0000 3FFF	XINTF Zone 0
0x0000 4000–0x0000 5FFF	XINTF Zone 1
0x0000 6000–0x0000 6FFF	Peripheral Frame 1 (4K X 16)
0x0000 7000–0x0000 7FFF	Peripheral Frame 2 (4K X 16)
0x0008 0000–0x000F FFFF	XINTF Zone 2
0x0010 0000–0x0017 FFFF	XINTF Zone 6
0x003F 8000–0x003F 9FFF	H0 SARAM (8K X 16)
0x003F 0000–0x003F FFFF	XINTF Zone 7
0x003F F000–0x003F FFFF	Boot ROM (4K X 16)

To summarize, it is possible to load code onto the unprotected on-chip program RAM shown in Table 2–3 via the JTAG connector without any impact from the CSM. The code can be debugged and the peripheral registers initialized, independent of whether the device is in secure or unsecure mode.

2.3 Incorporating Code Security in User Applications

Code security is typically not required in the development phase of a project; however, security is needed once a robust code is developed. Before such a code is programmed in the flash memory (or committed to ROM), a password should be chosen to secure the device. Once a password is in place, the device is secured (i.e., programming a password at the appropriate locations and either performing a device reset or setting the FORCESEC bit (CSMSCR.15) is the action that secures the device). From that time on, access to debug the contents of secure memory by any means (via JTAG, code running off external/on-chip memory etc.) requires the supply of a valid password. A password is not needed to run the code out of secure memory (such as in a typical end-customer usage); however, access to secure memory contents for debug purpose requires a password.

Table 2–4. Code Security Module (CSM) Registers

Memory Address	Register Name	Reset Values	Register Description
KEY Registers – Accessible by the user			
0x0000–0AE0	KEY0†	0xFFFF	Low word of the 128-bit KEY register
0x0000–0AE1	KEY1†	0xFFFF	Second word of the 128-bit KEY register
0x0000–0AE2	KEY2†	0xFFFF	Third word of the 128-bit KEY register
0x0000–0AE3	KEY3†	0xFFFF	Fourth word of the 128-bit key
0x0000–0AE4	KEY4†	0xFFFF	Fifth word of the 128-bit key
0x0000–0AE5	KEY5†	0xFFFF	Sixth word of the 128-bit key
0x0000–0AE6	KEY6†	0xFFFF	Seventh word of the 128-bit key
0x0000–0AE7	KEY7†	0xFFFF	High word of the 128-bit KEY register
0x0000–0AEF	CSMSCR†		CSM status and control register
PWL in Memory – Reserved for password only (on R281x devices, the password is set to all Fs)			
0x003F–7FF8	PWL0	User defined	Low word of the 128-bit password
0x003F–7FF9	PWL1	User defined	Second word of the 128-bit password
0x003F–7FFA	PWL2	User defined	Third word of the 128-bit password
0x003F–7FFB	PWL3	User defined	Fourth word of the 128-bit password
0x003F–7FFC	PWL4	User defined	Fifth word of the 128-bit password
0x003F–7FFD	PWL5	User defined	Sixth word of the 128-bit password
0x003F–7FFE	PWL6	User defined	Seventh word of the 128-bit password
0x003F–7FFF	PWL7	User defined	High word of the 128-bit password

† EALLOW protected

Figure 2–1. CSM Status and Control (CSMSCR) Register

15	14 7	6 1	0
FORCE-SEC	Reserved	Reserved	SECURE
W-1	R-0	R-10111	R-1

Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 2–5. CSM Status and Control (CSMSCR) Register Field Descriptions

Bits	Field	Description
15	FORCESEC	Writing a 1 clears the KEY registers and secures the device. A read always returns a zero.
0	SECURE	Read-only bit that reflects the security state of the device. 1 Device is secure (CSM locked) 0 Device is unsecure (CSM unlocked)

2.3.1 Environments That Require Security Unlocking

Following are the typical situations under which unsecuring can be required:

- Code development using debuggers (such as Code Composer Studio™)

  This is the most common environment during the design phase of a product.

- Flash programming using TI's flash utilities

  Flash programming is common during code development and testing. Once the user supplies the necessary password, the flash utilities disable the security logic before attempting to program the flash. The flash utilities can disable the code security logic in new devices without any authorization, since new devices come with an erased flash. However, reprogramming devices (that already contain custom passwords) require passwords to be supplied to the flash utilities in order to enable programming.

- Custom environment defined by the application

  In addition to the above, access to secure memory contents can be required in situations such as:

  - Using the on-chip bootloader to program the flash
  - Executing code from external memory or on-chip unsecure memory and requiring access to secure memory for lookup table. This is not a

suggested operating condition as supplying the password from external code could compromise code security.

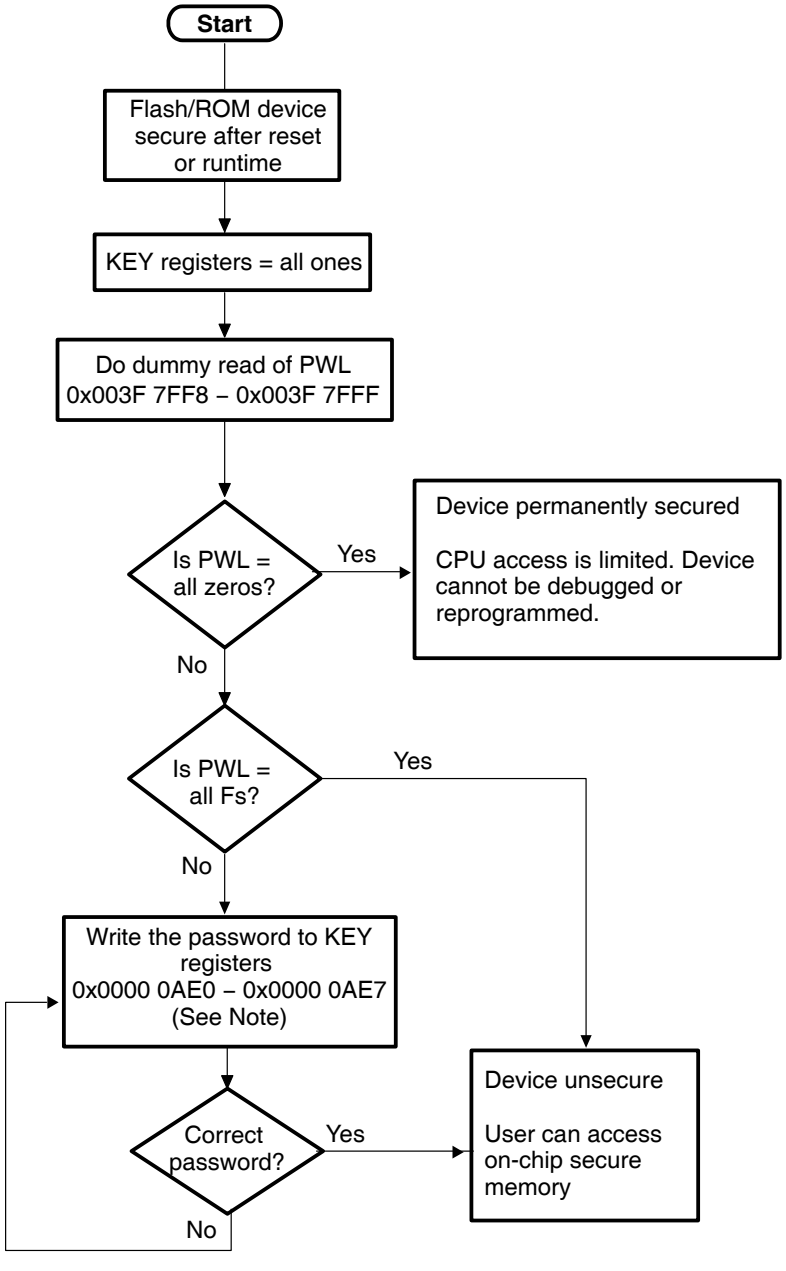
The unsecuring sequence is identical in all the above situations. This sequence is referred to as the *password match flow (PMF)* for simplicity. Figure 2–2 explains the sequence of operation that is required every time the user attempts to unsecure a device. A code example is listed for clarity.

2.3.2 Password Match Flow

Password match flow (PMF) is essentially a sequence of eight dummy reads from password locations (PWL) followed by eight writes to KEY registers.

Figure 2–2 shows how the PMF helps to initialize the security logic registers and disable security logic.

Figure 2–2. Password Match Flow (PMF)



Note: The KEY registers are EALLOW protected.

2.3.3 Unsecuring Considerations for Devices With/Without Code Security

Case 1 and Case 2 provide unsecuring considerations for devices with and without code security.

Case 1: Device With Code Security

A device with code security should have a predetermined password stored in the PWL (locations 0x003F 7FF8–0x003F 7FFF in memory). In addition, locations 0x3F7F80 – 0x3F7FF5 should be programmed with all 0x0000 and not used for program and/or data storage. The following are steps to unsecure this device:

1) Perform a dummy read of the PWL.

2) Write the password into the KEY registers (locations 0x0000 0AE0–0x0000 0AE7 in memory).

3) If the password is correct, the device becomes unsecure; otherwise, it stays secure.

Case 2: Device Without Code Security

A device without code security should have 0x FFFF FFFF FFFF FFFF FFFF FFFF FFFF FFFF (128 bits of all ones) stored in the PWL. The following are steps to use this device:

4) Perform a dummy read of the PWL.

5) Secure memory is fully accessible immediately after this operation is completed.

Note: A dummy read operation must be performed prior to reading, writing, or programming secure memory, even though the device is not protected with a password.

2.3.4 C Code Example to Unsecure

```
volatile int *CSM = (volatile int *)0x000AE0;    //CSM register file
volatile int *PWL = (volatile int *)0x3F7FF8;    //Password location
volatile int tmp;
int i ;
// Read the 128-bits of the password location (PWL)
// in Flash/ROM at address 0x3F7FF8-0x3F7FFF.
// If the device is secure, then the values read will
// not actually be loaded into the temp variable, so
// this is called a dummy read.
for (i = 0; i<8; i++) tmp = *PWL++;

// If the password location (PWL) is all = ones (0xFFFF),
// then the device will now be unsecure.  If the password
// is not all ones (0xFFFF), then the code below is required
// to unsecure the CSM.
// Write the 128-bit password to the KEY registers
// If this password matches that stored in the
// PWL then the CSM will become unsecure.  If it does not
// match, then the device will remain secure.
// An example password of:
// 0x0123456789ABCDEFCDEF89AB45670123 is used.
asm("  EALLOW");    // Key registers are EALLOW protected
*CSM++ = 0x0123;    // Register KEY0 at 0xAE0
*CSM++ = 0x4567;    // Register KEY1 at 0xAE1
*CSM++ = 0x89AB;    // Register KEY2 at 0xAE2
*CSM++ = 0xCDEF;    // Register KEY3 at 0xAE3
*CSM++ = 0xCDEF;    // Register KEY4 at 0xAE4
*CSM++ = 0x89AB;    // Register KEY5 at 0xAE5
*CSM++ = 0x4567;    // Register KEY6 at 0xAE6
*CSM++ = 0x0123;    // Register KEY7 at 0xAE7
asm("  EDIS");
```

2.3.5 C Code Example to Resecure

```
volatile int *CSM = 0x000AE0;       //CSM register file
//Set FORCESEC bit
asm("  EALLOW");                     //CSMSCR register is EALLOW protected.
*CSM = 0x8000;
asm ("EDIS");
```

2.4 DOs and DON'Ts to Protect Security Logic

2.4.1 DOs

- To keep the debug and code development phase simple, use the device in the unsecure mode; i.e., use 128 bits of all ones as PWL words (or use a password that is easy to remember). Use passwords after the development phase when the code is frozen.
- Recheck the passwords in PWL before programming the COFF file using flash utilities.
- The flow of code execution can freely toggle back and forth between secure memory and unsecure memory without compromising security. To access data variables located in secure memory when the device is secured, code execution must currently be running from secure memory.
- Program locations 0x3F7F80 – 0x3F7FF5 with 0x0000 when using the CSM.

2.4.2 DON'Ts

- If code security is desired, do not embed the password in your application anywhere other than in the PWL or security can be compromised.
- Do not use 128 bits of all zeros as the password. This automatically secures the device, regardless of the contents of the KEY register. The device is not debuggable nor reprogrammable.
- Do not pull a reset after clearing the flash array but before erasing the array. This leaves zeros in the PWL that automatically secures the device, regardless of the contents of the KEY register. The device is not debuggable nor reprogrammable.
- Do not use locations 0x3F7F80 – 0x3F7FF5 to store program and/or data. These locations should be programmed to 0x0000 when using the CSM.

2.5 CSM Features – Summary

1) The flash is secured after a reset until the password match flow (PMF) is executed.

2) The standard way of running code out of the flash or ROM is to program the flash with the code (for ROM devices the program is hardcoded at device fabrication) and powering up the DSP in microcomputer mode. Since instruction fetches are always allowed from secure memory, regardless of the state of the CSM, the code functions correctly even without executing the PMF.

3) Secure memory cannot be modified while the device is secured.

4) Secure memory cannot be read from any code running from unsecure memory while the device is secured.

5) Secure memory cannot be read or written to by the debugger (i.e., Code Composer Studio™) at any time that the device is secured.

6) Complete access to secure memory from both the CPU code and the debugger is granted while the device is unsecured.

Chapter 3

Clocking

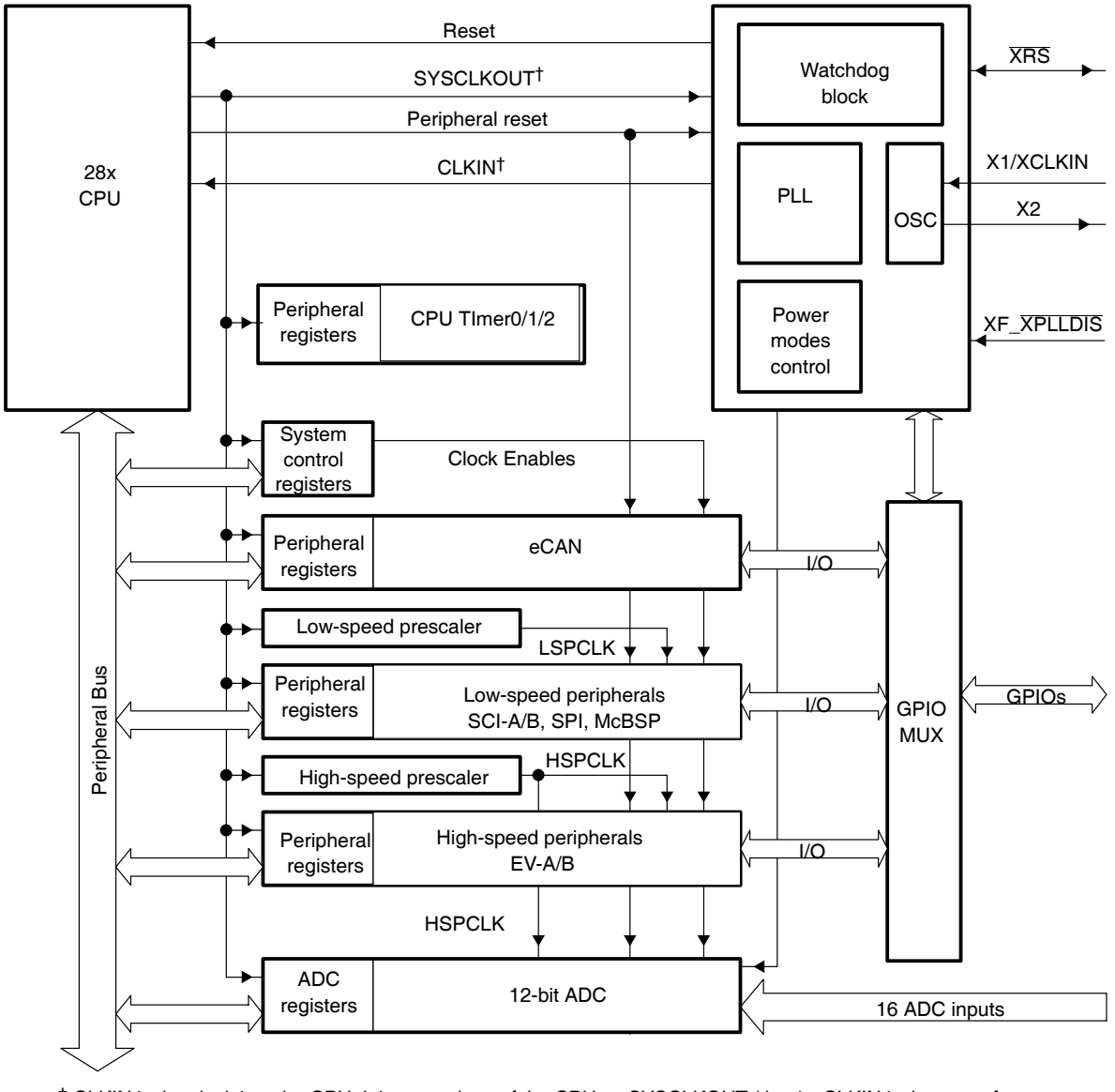
This section describes the 281x oscillator, PLL and clocking mechanisms, the watchdog function, and the low-power modes.

Topic **Page**

3.1 Clocking and System Control

Figure 3–1 shows the various clock and reset domains in the 281x devices.

Figure 3–1. Clock and Reset Domains



† CLKIN is the clock into the CPU. It is passed out of the CPU as SYSCLKOUT (that is, CLKIN is the same frequency as SYSCLKOUT).

The PLL, clocking, watchdog and low-power modes, are controlled by the registers listed in Table 3–1.

Table 3–1. PLL, Clocking, Watchdog, and Low-Power Mode Registers†

Name	Address	Size (x16)	Description
Reserved	0x0000 7010 0x0000 7019	10	
HISPCP	0x0000 701A	1	High-Speed Peripheral Clock Prescaler Register for HSPCLK Clock
LOSPCP	0x0000 701B	1	Low-Speed Peripheral Clock Prescaler Register for HSPCLK clock
PCLKCR	0x0000 701C	1	Peripheral Clock Control Register
Reserved	0x0000 701D	1	
LPMCR0	0x0000 701E	1	Low Power Mode Control Register 0
LPMCR1	0x0000 701F	1	Low Power Mode Control Register 1
Reserved	0x0000 7020	1	
PLLCR	0x0000 7021	1	PLL Control Register‡
SCSR	0x0000 7022	1	System Control & Status Register
WDCNTR	0x0000 7023	1	Watchdog Counter Register
Reserved	0x0000 7024	1	
WDKEY	0x0000 7025	1	Watchdog Reset Key Register
Reserved	0x0000 7026 0x0000 7028	3	
WDCR	0x0000 7029	1	Watchdog Control Register
Reserved	0x0000 702A 0x0000 702F	6	

† All of the registers in this table can be accessed only by executing the EALLOW instruction.
‡ The PLL control register (PLLCR) is reset to a known state by the $\overline{\text{XRS}}$ signal only.

The PCLKCR register enables/disables clocks to the various peripheral modules in the 281x devices. Figure 3–2 lists the bit descriptions of the PCLKCR register.

Figure 3–2. Peripheral Clock Control (PCLKCR) Register

15	14	13	12	11	10	9	8
Reserved	ECANENCLK	Reserved	MCBSPENCLK	SCIBENCLK	SCIAENCLK	Reserved	SPIENCLK
R-0	R/W-0	R-0	R/W-0	R/W-0	R/W-0	R-0	R/W-0

7	4	3	2	1	0
Reserved		ADCENCLK	Reserved	EVBENCLK	EVAENCLK
R-0		R/W-0	R-0	R/W-0	R/W-0

Legend: R = Read access, -0 = value after reset

Notes: 1) EALLOW-protected register

2) If a peripheral block is not used, then the clock to that peripheral can be turned off to minimize power consumption.

Table 3–2. Peripheral Clock Control (PCLKCR) Register Field Descriptions

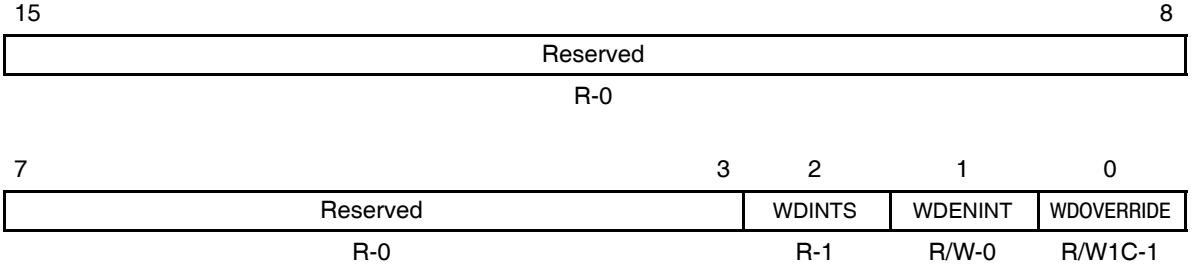
Bits	Field	Description
15	Reserved	Reserved
14	ECANENCLK	If ECANENCLK is set, it enables the system clock within the CAN peripheral. For low power operation, this bit is set to zero by the user or by reset.
13	Reserved	Reserved
12	MCBSPENCLK	If MCBSPENCLK is set, it enables the low-speed clock (LSPCLK) within the McBSP peripheral. For low power operation, this bit is set to zero by the user or by reset.
11	SCIBENCLK	If SCIBENCLK is set, it enables the low-speed clock (LSPCLK) within the SCI-B peripheral. For low power operation, this bit is set to zero by the user or by reset.
10	SCIAENCLK	If SCIAENCLK is set, it enables the low-speed clock (LSPCLK) within the SCI-A peripheral. For low power operation, this bit is set to zero by the user or by reset.
9	Reserved	Reserved
8	SPIAENCLK	If SPIAENCLK is set, it enables the low-speed clock (LSPCLK) within the SPI peripheral. For low power operation, this bit is set to zero by the user or by reset.
7–4	Reserved	
3	ADCENCLK	If ADCENCLK is set, it enables the high-speed clock (HSPCLK) within the ADC peripheral. For low power operation, this bit is set to zero by the user or by reset.
2	Reserved	Reserved

Table 3–2. Peripheral Clock Control (PCLKCR) Register Field Descriptions (Continued)

Bits	Field	Description
1	EVBENCLK	If EVBENCLK is set, it enables the high-speed clock (HSPCLK) within the EV-B peripheral. For low power operation, this bit is set to zero by the user or by reset.
0	EVAENCLK	If EVAENCLK is set, it enables the high-speed clock (HSPCLK) within the EV-A peripheral. For low power operation, this bit is set to zero by the user or by reset.

The system control and status register contains the watchdog override bit and the watchdog interrupt enable/disable bit. Figure 3–3 describes the bit functions of the SCSR register.

Figure 3–3. System Control and Status (SCSR) Register



Legend: R = Read access, -0 = value after reset, W1C = Write 1 to clear

Note: EALLOW-protected register

Table 3–3. System Control and Status (SCSR) Register Field Descriptions

Bits	Field	Description
15–3	Reserved	
2	WDINTS	Watchdog interrupt status bit. WDINTS reflects the current state of the $\overline{\text{WDINT}}$ signal from the watchdog block. If the watchdog interrupt is used to wake the device from IDLE or STANDBY, then you should check this bit to make sure the watchdog interrupt signal is no longer active (WDINTS = 1) before attempting to go back into IDLE or STANDBY mode.

Table 3–3. System Control and Status (SCSR) Register Field Descriptions (Continued)

Bits	Field	Description
1	WDENINT	Watchdog enable interrupt. 0 The $\overline{\text{WDRST}}$ output signal is enabled and the $\overline{\text{WDINT}}$ output signal is disabled. This is the default state on reset ($\overline{\text{XRS}}$). 1 The watchdog reset ($\overline{\text{WDRST}}$) output signal is disabled and the watchdog interrupt ($\overline{\text{WDINT}}$) output signal is enabled.
0	WDOVERRIDE	If WDOVERRIDE is set to 1, the user is allowed to change the state of the Watchdog disable (WDDIS) bit in the Watchdog Control (WDCR) register. If the WDOVERRIDE bit is cleared by writing a 1, the WDDIS bit cannot be modified by the user. This also enables the watchdog if it is currently disabled. Writing a 0 has no effect. If this bit is cleared, then it remains in this state until a reset occurs. The current state of this bit is readable by the user.

The HISPCP and LOSPCP registers are used to configure the high- and low-speed peripheral clocks, respectively. See Figure 3–4 for the HISPCP bit definitions and Figure 3–5 for the LOSPCP bit definitions.

Figure 3–4. High-Speed Peripheral Clock Prescaler (HISPCP) Register

15	3	2	0
Reserved		HSPCLK	
R-0		R/W-001	

Legend: R = Read access, -0 = value after reset

Note: EALLOW-protected register

Table 3–4. High-Speed Peripheral Clock Prescaler (HISPCP) Register Field Descriptions

Bits	Field	Value	Description
15–3	Reserved		
2–0	HSPCLK		These bits configure the high-speed peripheral clock (HSPCLK) rate relative to SYSCLKOUT: If HISPCP ≠ 0, HSPCLK = SYSCLKOUT/(HISPCP† X 2) If HISPCP = 0, HSPCLK = SYSCLKOUT
		000	High speed clock = SYSCLKOUT/1
		001	High speed clock = SYSCLKOUT/2 (reset default)
		010	High speed clock = SYSCLKOUT/4
		011	High speed clock = SYSCLKOUT/6
		100	High speed clock = SYSCLKOUT/8
		101	High speed clock = SYSCLKOUT/10
		110	High speed clock = SYSCLKOUT/12
		111	High speed clock = SYSCLKOUT/14

† HISPCP in this equation denotes the value of bits 2:0 in the HISPCP register.

Figure 3–5. Low-Speed Peripheral Clock Prescaler (LOSPCP) Register

15	3	2	0
Reserved		LSPCLK	
R-0		R/W-010	

Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 3–5. Low-Speed Peripheral Clock Prescaler (LOSPCP) Register Field Descriptions

Bits	Field	Description
15–3	Reserved	
2–0	LSPCLK	These bits configure the low-speed peripheral clock (LSPCLK) rate relative to SYSCLKOUT: If LOSPCP ≠ 0, LSPCLK = SYSCLKOUT/(LOSPCP† X 2) If LOSPCP = 0, LSPCLK = SYSCLKOUT
		000 Low speed clock = SYSCLKOUT/1
		001 Low speed clock= SYSCLKOUT/2
		010 Low speed clock= SYSCLKOUT/4 (reset default)

Table 3–5. Low-Speed Peripheral Clock Prescaler (LOSPCP) Register Field Descriptions (Continued)

Bits	Field	Description	
		011	Low speed clock= SYSCLKOUT/6
		100	Low speed clock= SYSCLKOUT/8
		101	Low speed clock= SYSCLKOUT/10
		110	Low speed clock= SYSCLKOUT/12
		111	Low speed clock= SYSCLKOUT/14

† LOSPCP in this equation denotes the value of bits 2:0 in the LOSPCP register.

3.2 OSC and PLL Block

The on-chip oscillator and phase-locked loop (PLL) block provides the clocking signals for the device, as well as control for low-power mode entry.

3.2.1 PLL-Based Clock Module

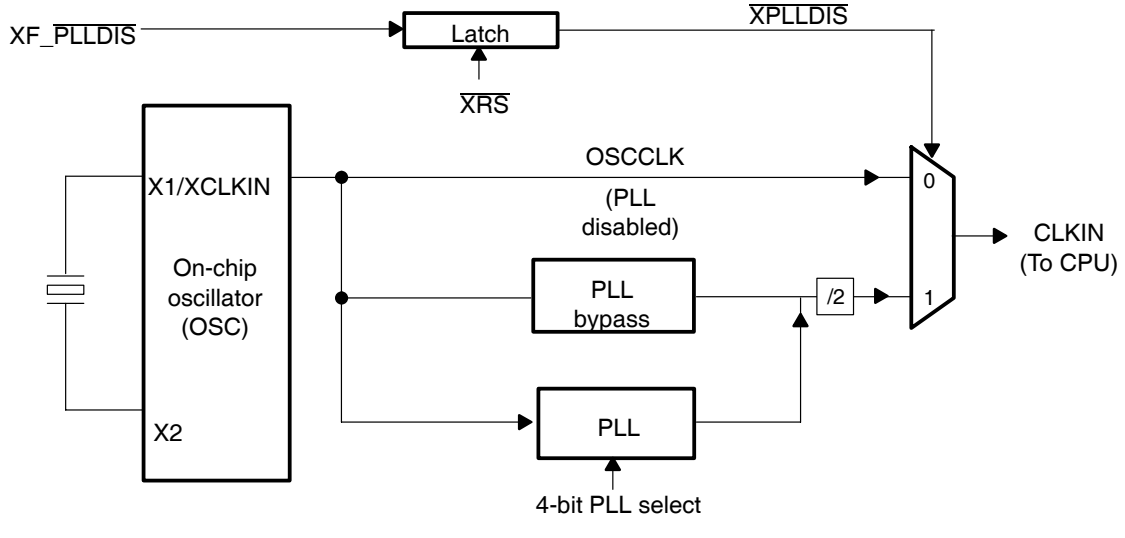
The F281x devices have an on-chip, PLL-based clock module. The PLL has a 4-bit ratio control to select different CPU clock rates.

The PLL-based clock module provides two modes of operation:

- Crystal-operation
  This mode allows the use of an external crystal/resonator to provide the time base to the device.
- External clock source operation
  This mode allows the internal oscillator to be bypassed. The device clocks are generated from an external clock source input on the XTAL1/CLKIN pin. In this case, an external oscillator clock is connected to the XTAL1/CLKIN pin.

Figure 3–6 shows the OSC and PLL block on the 281x.

Figure 3–6. OSC and PLL Block



The OSC circuit enables a crystal to be attached to the 281x devices using the X1/XCLKIN and X2 pins. If a crystal is not used, then an external oscillator can be directly connected to the X1/XCLKIN pin and the X2 pin is left unconnected.

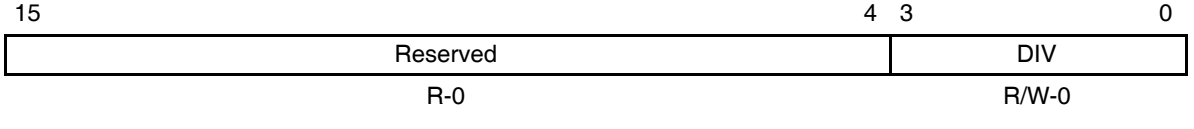
See the *TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, TMS320C2812 Digital Signal Processors Data Manual* (literature number SPRS174) and the *TMS320R2811 and TMS320R2812 Digital Signal Processors* (literature number SPRS257).

Table 3–6. Possible PLL Configuration Modes

PLL Mode	Remarks	SYSCLKOUT
PLL Disabled	Invoked by tying $\overline{\text{XPLLDIS}}$ pin low upon reset. PLL block is completely disabled. Clock input to the CPU (CLKIN) is directly derived from the clock signal present at the X1/XCLKIN pin.	XCLKIN
PLL Bypassed	Default PLL configuration upon power-up, if PLL is not disabled. The PLL itself is bypassed. However, the /2 module in the PLL block divides the clock input at the X1/XCLKIN pin by two before feeding it to the CPU.	XCLKIN/2
PLL Enabled	Achieved by writing a non-zero value “n” into PLLCR register. The /2 module in the PLL block now divides the output of the PLL by two before feeding it to the CPU.	(XCLKIN * n) / 2

The PLLCR register DIV field (bits 3–0) is used to change the PLL multiplier of the device. When the CPU writes to the DIV bits, the PLL logic switches the CPU clock (CLKIN) to OSCCLK/2. Once the PLL is stable and has locked at the new specified frequency, the PLL switches CLKIN to the new frequency as shown in Figure 3–7. The time it takes for the PLL to switch from OSCCLK/2 to the new frequency is 131072 OSCCLK cycles. For time-critical software, you should insert a software delay of the required locking period after writing to the PLLCR register in order for the PLL to complete the locking sequence.

Figure 3–7. PLLCR Register



Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 3–7. PLLCR Register Field Descriptions

Bits	Name	Description
15–4	Reserved	
3–0	DIV	The DIV field controls whether the PLL is bypassed or not, and sets the PLL clocking ratio when not bypassed. 0000 CLKIN = OSCCLK/2 (PLL bypass) 0001 CLKIN = (OSCCLK * 1.0)/2 0010 CLKIN = (OSCCLK * 2.0)/2

Table 3–7. PLLCR Register Field Descriptions (Continued)

Bits	Name	Description	
		0011	CLKIN = (OSCCLK * 3.0)/2
		0100	CLKIN = (OSCCLK * 4.0)/2
		0101	CLKIN = (OSCCLK * 5.0)/2
		0110	CLKIN = (OSCCLK * 6.0)/2
		0111	CLKIN = (OSCCLK * 7.0)/2
		1000	CLKIN = (OSCCLK * 8.0)/2
		1001	CLKIN = (OSCCLK * 9.0)/2
		1010	CLKIN = (OSCCLK * 10.0)/2
		1011–1111	Reserved

† The PLLCR register is reset to a known state by the $\overline{\text{XRS}}$ reset line. If a reset is issued by the debugger, the PLL clocking ratio is not changed.

3.2.2 External Reference Oscillator Clock Option

TI recommends that customers have the resonator/crystal vendor characterize the operation of their device with the DSP chip. The resonator/crystal vendor has the equipment and expertise to tune the tank circuit. The vendor can also advise the customer regarding the tank component values for proper startup and stability over the entire operating range.

3.3 Low-Power Modes Block

The low-power modes on the 281x devices are similar to the 240x devices. Table 3–8 summarizes the various modes.

Table 3–8. 281x Low-Power Modes

Mode	LPMCR0[1:0]	OSCCLK	CLKIN	SYSCLKOUT	Exit†
IDLE	00	On	On	On‡	$\overline{\text{XRS}}$, WAKEINT, Any Enabled Interrupt, XNMI_XINT13
STANDBY	01	On (watchdog still running)	Off	Off	$\overline{\text{XRS}}$, WAKEINT, XINT1, XNMI_XINT13, $\overline{\text{T1/2/3/4CTRIP}}$, $\overline{\text{C1/2/3/4/5/6TRIP}}$, SCIRXDA, SCIRXDB, CANRX, Debugger§
HALT	1X	Off (oscillator and PLL turned off, watchdog not functional)	Off	Off	$\overline{\text{XRS}}$, XNMI_XINT13, Debugger§

† The Exit column lists which signals or under what conditions the low power mode is exited. This signal must be kept low long enough for an interrupt to be recognized by the device. Otherwise the IDLE mode is not exited and the device goes back into the indicated low power mode.

‡ The IDLE mode on the 28x behaves differently than on the 24x/240x. On the 28x, the clock output from the CPU (SYSCLKOUT) is still functional while on the 24x/240x the clock is turned off.

§ On the 28x, the JTAG port can still function even if the clock to the CPU (CLKIN) is turned off.

The various low-power modes operate as follows:

IDLE Mode: This mode is exited by any enabled interrupt or an NMI that is recognized by the processor. The LPM block performs no tasks during this mode as long as the LPMCR0[1:0] bits are set to 00.

HALT Mode: Only the $\overline{\text{XRS}}$ and XNMI_XINT13 external signals can wake the device from HALT mode. The XNMI input to the CPU has an enable/disable bit in the XMNICR register.

STANDBY Mode: All other signals (including XNMI) wake the device from STANDBY mode if selected by the LPMCR1 register. You must select which signal(s) wake the device. The selected signal(s) are also qualified by the OSCCLK before waking the device. The number of OSCCLK cycles is specified in the LPMCR0 register.

The low-power modes are controlled by the LPMCR0 register (see Figure 3–8) and the LPMCR1 register (see Figure 3–9).

Figure 3–8. Low Power Mode Control 0 (LPMCR0) Register

15 – 8	7 – 2	1 – 0
Reserved	QUALSTDBY	LPM
R-0	R/W-1	R/W-0

Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 3–9. Low Power Mode Control 0 (LPMCR0) Register Field Descriptions

Bits	Field	Description†
15–8	Reserved	Reserved
7–2	QUALSTDBY	Select number of OSCCLK clock cycles to qualify the selected inputs when waking the device from STANDBY mode: 000000 = 2 OSCCLKs 000001 = 3 OSCCLKs . 111111 = 65 OSCCLKs
1–0	LPM‡	These bits set the low power mode for the device. 00 Set the low power mode to IDLE 01 Set the low power mode to STANDBY 1x Set the low power mode to HALT

† These bits are cleared by a reset ($\overline{XRS}$).

‡ The low power mode bits (LPM) are only valid when the IDLE instruction is executed. Therefore, the user must set the LPM bits to the appropriate mode before executing the IDLE instruction.

Figure 3–9. Low Power Mode Control 1 (LPMCR1) Register

15	14	13	12	11	10	9	8
CANRX	SCIRXB	SCIRXA	C6TRIP	C5TRIP	C4TRIP	C3TRIP	C2TRIP
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
C1TRIP	T4CTRIP	T3CTRIP	T2CTRIP	T1CTRIP	WDINT	XNMI	XINT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 3–10. Low Power Mode Control 1 (LPMCR1) Register Field Descriptions

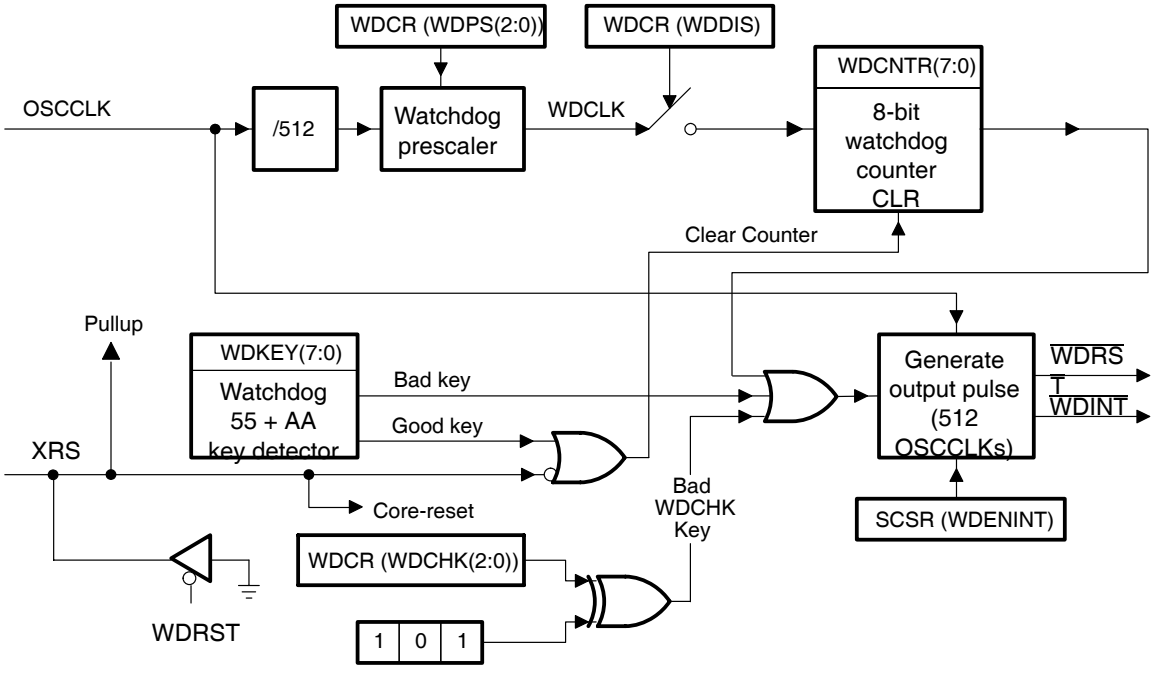
Bits	Field	Description
15	CANRX	If the respective bit is set to 1, it enables the selected signal to wake the device from STANDBY mode. If the bit is cleared, the signal has no effect.
14	SCIRXB	
13	SCIRXA	
12	C6TRIP	
11	C5TRIP	
10	C4TRIP	
9	C3TRIP	
8	C2TRIP	
7	C1TRIP	
6	T4CTRIP	
5	T3CTRIP	
4	T2CTRIP	
3	T1CTRIP	
2	$\overline{\text{WDINT}}$	
1	XNMI_XINT13	
0	XINT1	

† These bits are cleared by a reset ($\overline{\text{XRS}}$).

3.4 Watchdog Block

The watchdog block on the 281x is similar to the one used on the 240x devices. The watchdog module generates an output pulse, 512 oscillator clocks wide (OSCCLK), whenever the 8-bit watchdog up counter has reached its maximum value. To prevent this, the user disables the counter or the software must periodically write a 0x55 + 0xAA sequence into the watchdog key register which resets the watchdog counter. Figure 3–10 shows the various functional blocks within the watchdog module.

Figure 3–10. Watchdog Module



NOTE A: The $\overline{\text{WDRST}}$ signal is driven low for 512 OSCCLK cycles.

The watchdog interrupt ($\overline{\text{WDINT}}$) signal enables the watchdog to be used as a wakeup from IDLE/STANDBY mode timer. If the watchdog interrupt is used to wake up from an IDLE or STANDBY low power mode condition, then you should make sure that the $\overline{\text{WDINT}}$ signal goes back high again before attempting to go back into the IDLE or STANDBY mode. You can determine the state of this signal by reading the WDENINT bit in the SCSR register.

In STANDBY mode, all peripherals are turned off on the device. The only peripheral that remains functional is the watchdog. The WATCHDOG module

runs off the PLL clock or the oscillator clock. The $\overline{\text{WDINT}}$ signal is fed to the LPM block so that it can wake the device from STANDBY (if enabled). See Low-Power Modes Block section of this data sheet for more details.

In IDLE mode, the $\overline{\text{WDINT}}$ signal can generate an interrupt to the CPU, (the WAKEINT interrupt in the PIE), to take the CPU out of IDLE mode.

In HALT mode, this feature cannot be used because the oscillator (and PLL) are turned off and, therefore, so is the watchdog.

Figure 3–11. Watchdog Counter (WDCNTR) Register

15	8	7	0
Reserved		WDCNTR	
R-0		R-0	

Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 3–11. Watchdog Counter (WDCNTR) Register Field Descriptions

Bits	Field	Description
15–8	Reserved	
7–0	WDCNTR	These bits contain the current value of the WD counter. The 8-bit counter continually increments at the watchdog clock (WDCLK), rate. If the counter overflows, then the watchdog initiates a reset. If the WDKEY register is written with a valid combination, then the counter is reset to zero. The watchdog clock rate is configured in the WDCR register.

Figure 3–12. Watchdog Reset Key (WDKEY) Register

15	8	7	0
Reserved		WDKEY	
R-0		R/W-0	

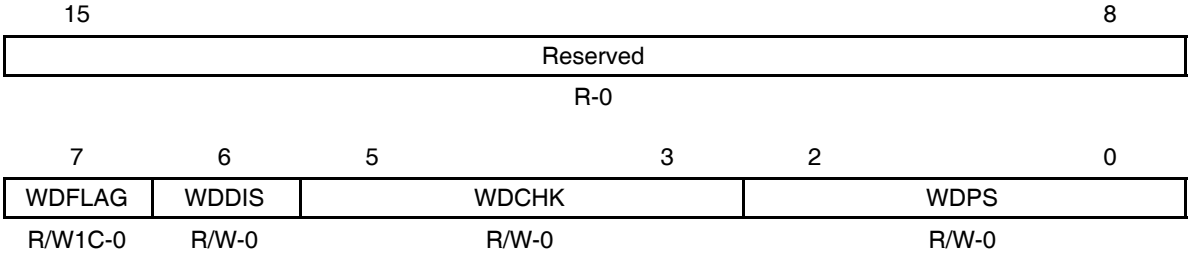
Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 3–12. Watchdog Reset Key (WDKEY) Register Field Descriptions

Bits	Field	Description
15–8	Reserved	Reserved
7–0	WDKEY	Writing 0x55 followed by 0xAA causes the WDCNTR bits to be cleared. Writing any other value causes an immediate watchdog reset to be generated. Reads return the value of the WDCR register.

Figure 3–13. Watchdog Control (WDCR) Register



Legend: R = Read access, W = write access, W1C = write 1 to clear, -0 = value after reset

Note: EALLOW-protected register

Table 3–13. Watchdog Control (WDCR) Register Field Descriptions

Bits	Field	Description
15–8	Reserved	
7	WDFLAG	Watchdog reset status flag bit. WDFLAG, if set, indicates a watchdog reset ($\overline{\text{WDRST}}$) generated the reset condition. If 0, then it was an external device or power-up reset condition. This bit remains latched until the user writes a 1 to clear the condition. Writes of 0 are ignored.
6	WDDIS	Watchdog disable. Writing a 1 to WDDIS disables the watchdog module. Writing a 0 enables the module. WDDISt can only be modified if the WDOVERRIDE bit in the SCSR2 register is set to 1. On reset, the watchdog module is enabled.
5–3	WDCHK(2–0)	Watchdog check. You must ALWAYS write 1,0,1 to these bits whenever a write to this register is performed. Writing any other value causes an immediate reset to the core (if WD enabled). These bits always read back as 0, 0, 0.
2–0	WDPS(2–0)	These bits configure the watchdog counter clock (WDCLK) rate relative to OSCCLK/512: 000 WDCLK = OSCCLK/512/1 001 WDCLK = OSCCLK/512/1 010 WDCLK = OSCCLK/512/2 011 WDCLK = OSCCLK/512/4 100 WDCLK = OSCCLK/512/8 101 WDCLK = OSCCLK/512/16 110 WDCLK = OSCCLK/512/32 111 WDCLK = OSCCLK/512/64

When the $\overline{\text{XRS}}$ line is low, the WDFLAG bit is forced low. The WDFLAG bit is only set if a rising edge on $\overline{\text{WDRST}}$ signal is detected (after synch and a 4 cycle

delay) and the $\overline{\text{XRS}}$ signal is high. If the $\overline{\text{XRS}}$ signal is low when $\overline{\text{WDRST}}$ goes high, then the WDFLAG bit remains at 0. In a typical application, the $\overline{\text{WDRST}}$ signal connects to the $\overline{\text{XRS}}$ input. Hence, to distinguish between a watchdog reset and an external device reset, an external reset must be longer in duration then the watchdog pulse.

3.4.1 Emulation Considerations

The watchdog module behaves as follows under various debug conditions:

CPU Suspended: When the CPU is suspended, the watchdog clock (WDCLK) is suspended.

Run-Free Mode: When the CPU is placed in run-free mode, then the watchdog module resumes operation as normal.

Real-Time Single-Step Mode: When the CPU is in real-time single-step mode, the watchdog clock (WDCLK) is suspended. The watchdog remains suspended even within real-time interrupts.

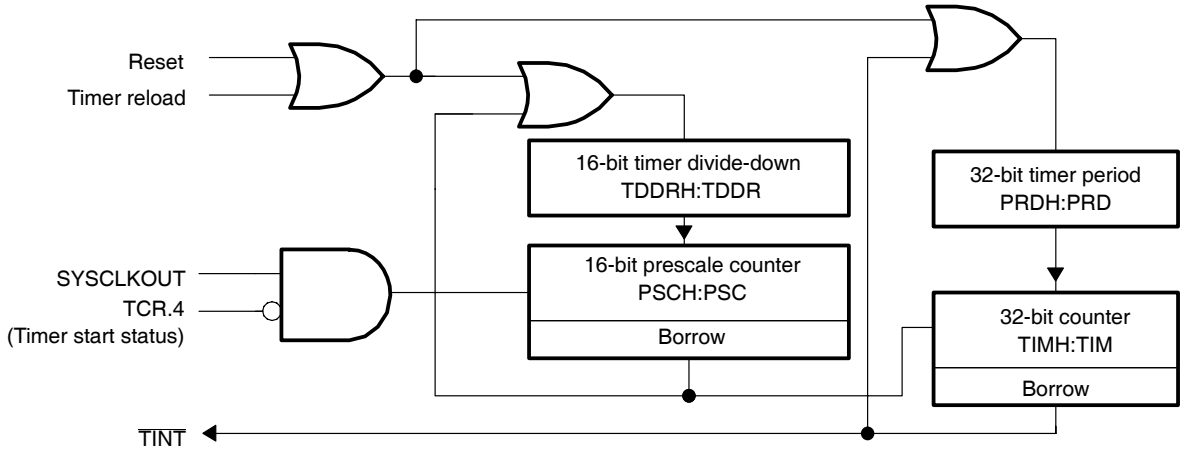
Real-Time Run-Free Mode: When the CPU is in real-time run-free mode, the watchdog operates as normal.

3.5 32-Bit CPU Timers 0/1/2

This section describes the three 32-bit CPU-timers on the 281x devices (TIMER0/1/2).

CPU-Timers 1 and 2 are reserved for the real-time OS (such as DSP-BIOS).† CPU-Timer 0 can be used in user applications.

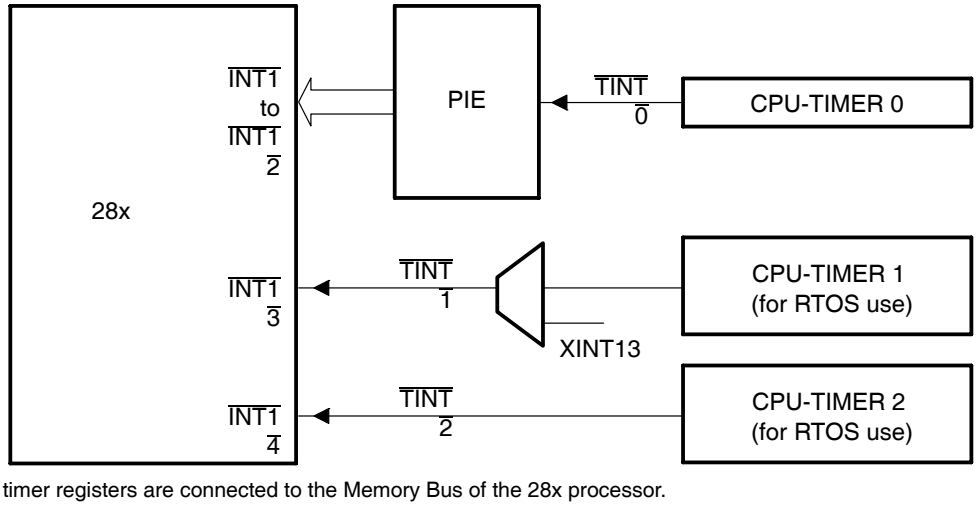
Figure 3–14. CPU-Timers



NOTE A: The CPU-Timers are different from the general-purpose (GP) timers that are present in the event manager modules (EVA, EVB).

In the 281x devices, the CPU-timer interrupt signals ($\overline{\text{TINT0}}$, $\overline{\text{TINT1}}$, $\overline{\text{TINT2}}$) are connected as shown in Figure 3–15.

Figure 3–15. CPU-Timer Interrupts Signals and Output Signal



NOTES: A. The timer registers are connected to the Memory Bus of the 28x processor.
B. The timing of the timers is synchronized to SYSCLKOUT of the processor clock.

The general operation of the CPU-timer is as follows: The 32-bit counter register TIMH:TIM is loaded with the value in the period register PRDH:PRD. The counter register decrements at the SYSCLKOUT rate of the 28x. When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse. The registers listed in Table 3−14 are used to configure the timers.

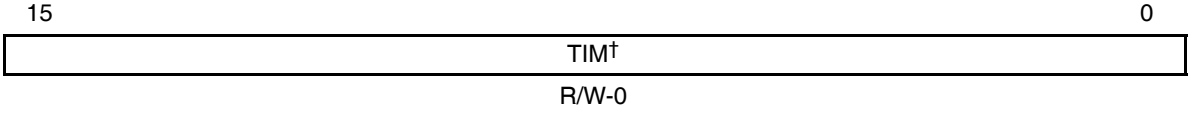
Table 3−14. CPU-Timers 0, 1, 2 Configuration and Control Registers

Name	Address	Size (x16)	Description
TIMER0TIM	0x0000 0C00	1	CPU-Timer 0, Counter Register
TIMER0TIMH	0x0000 0C01	1	CPU-Timer 0, Counter Register High
TIMER0PRD	0x0000 0C02	1	CPU-Timer 0, Period Register
TIMER0PRDH	0x0000 0C03	1	CPU-Timer 0, Period Register High
TIMER0TCR	0x0000 0C04	1	CPU-Timer 0, Control Register
Reserved	0x0000 0C05	1	
TIMER0TPR	0x0000 0C06	1	CPU-Timer 0, Prescale Register
TIMER0TPRH	0x0000 0C07	1	CPU-Timer 0, Prescale Register High
TIMER1TIM	0x0000 0C08	1	CPU-Timer 1, Counter Register
TIMER1TIMH	0x0000 0C09	1	CPU-Timer 1, Counter Register High
TIMER1PRD	0x0000 0C0A	1	CPU-Timer 1, Period Register
TIMER1PRDH	0x0000 0C0B	1	CPU-Timer 1, Period Register High
TIMER1TCR	0x0000 0C0C	1	CPU-Timer 1, Control Register
Reserved	0x0000 0C0D	1	
TIMER1TPR	0x0000 0C0E	1	CPU-Timer 1, Prescale Register
TIMER1TPRH	0x0000 0C0F	1	CPU-Timer 1, Prescale Register High
TIMER2TIM	0x0000 0C10	1	CPU-Timer 2, Counter Register
TIMER2TIMH	0x0000 0C11	1	CPU-Timer 2, Counter Register High
TIMER2PRD	0x0000 0C12	1	CPU-Timer 2, Period Register
TIMER2PRDH	0x0000 0C13	1	CPU-Timer 2, Period Register High
TIMER2TCR	0x0000 0C14	1	CPU-Timer 2, Control Register
Reserved	0x0000 0C15	1	

Table 3–14. CPU-Timers 0, 1, 2 Configuration and Control Registers (Continued)

Name	Address	Size (x16)	Description
TIMER2TPR	0x0000 0C16	1	CPU-Timer 2, Prescale Register
TIMER2TPRH	0x0000 0C17	1	CPU-Timer 2, Prescale Register High
Reserved	0x0000 0C18 0x0000 0C3F	40	

Figure 3–16. TIMERxTIM Register



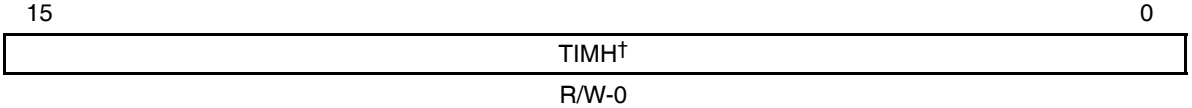
Legend: R = Read access, W = write access, -0 = value after reset

Table 3–15. TIMERxTIM Register Field Descriptions

Bits	Field	Description
15–0	TIM	CPU-Timer Counter Registers (TIMH:TIM): The TIM register holds the low 16 bits of the current 32-bit count of the timer. The TIMH register holds the high 16 bits of the current 32-bit count of the timer. The TIMH:TIM decrements by one every (TDDRH:TDDR+1) clock cycles, where TDDRH:TDDR is the timer prescale divide-down value. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers. The timer interrupt ($\overline{\text{TINT}}$) signal is generated.

† x = 0, 1, or 2

Figure 3–17. TIMERxTIMH Register



Legend: R = Read access, W = write access, -0 = value after reset

Table 3–16. TIMERxTIMH Register Field Descriptions

Bits	Field	Description
15–0	TIMH	See description for TIMERxTIM.

† x = 0, 1, or 2

Figure 3–18. TIMERxPRD Register†

15	0
PRD	
R/W-0	

Legend: R = Read access, W = write access, -0 = value after reset

Table 3–17. TIMERxPRD Register Field Descriptions

Bits	Field	Description
15–0	PRD	CPU-Timer Period Registers (PRDH:PRD): The PRD register holds the low 16 bits of the 32-bit period. The PRDH register holds the high 16 bits of the 32-bit period. When the TIMH:TIM decrements to zero, the TIMH:TIM register is reloaded with the period value contained in the PRDH:PRD registers, at the start of the next timer input clock cycle (the output of the prescaler). The PRDH:PRD contents are also loaded into the TIMH:TIM when you set the timer reload bit (TRB) in the Timer Control Register (TCR).

† x = 0, 1, or 2

Figure 3–19. TIMERxPRDH Register †

15	0
PRDH	
R/W-0	

Legend: R = Read access, W = write access, -0 = value after reset

Table 3–18. TIMERxPRDH Register Field Description

Bits	Field	Description
15–0	PRDH	See description for TIMERxPRD

† x = 0, 1, or 2

Figure 3–20. TIMERxTCR Register†

15	14	13 12	11	10	9 8
TIF	TIE	Reserved	FREE	SOFT	Reserved
R/W-0	R/W-0	R-0	R/W-0	R/W-0	R-0

7 6	5	4	3 0
Reserved	TRB	TSS	Reserved
R-0	R/W-0	R/W-0	R-0

Legend: R = Read access, W = write access, -0 = value after reset

Table 3–19. TIMERxTCR Register Field Descriptions

Bits	Field	Description
15	TIF	CPU-Timer Interrupt Flag. This flag gets set when the timer decrements to zero. TIF can be cleared by software writing a 1, but it can only be set by the timer reaching zero. 0 Writing a zero has no effect. 1 Writing a 1 to this bit clears it
14	TIE	CPU-Timer Interrupt Enable. If the timer decrements to zero, and TIE is set, the timer asserts its interrupt request.
13–12	Reserved	Reserved
11 10	FREE SOFT	CPU-Timer Emulation Modes: These bits are special emulation bits that determine the state of the timer when a breakpoint is encountered in the high-level language debugger. If the FREE bit is set to 1, then, upon a software breakpoint, the timer continues to run (that is, free runs). In this case, SOFT is a *don't care*. But if FREE is 0, then SOFT takes effect. In this case, if SOFT = 0, the timer halts the next time the TIMH:TIM decrements. If the SOFT bit is 1, then the timer halts when the TIMH:TIM has decremented to zero.

FREE	SOFT	CPU-Timer Emulation Mode
0	0	Stop after the next decrement of the TIMH:TIM (hard stop)
0	1	Stop after the TIMH:TIM decrements to 0 (soft stop)
1	0	Free run
1	1	Free run

Note: That in the SOFT STOP mode, the timer generates an interrupt before shutting down (since reaching 0 is the interrupt causing condition).

9–6	Reserved	Reserved

Table 3–19. TIMERxTCR Register Field Descriptions (Continued)

Bits	Field	Description
5	TRB	CPU-Timer Reload bit. When you write a 1 to TRB, the TIMH:TIM is loaded with the value in the PRDH:PRD, and the prescaler counter (PSCH:PSC) is loaded with the value in the timer divide-down register (TDDRH:TDDR). The TRB bit is always read as zero.
4	TSS	CPU-Timer stop status bit. TSS is a 1-bit flag that stops or starts the timer. 0 To start or restart the timer, set TSS to 0. At reset, TSS is cleared to 0 and the timer immediately starts. 1 To stop the timer, set TSS to 1.
3–0	Reserved	Reserved

† x = 0, 1, or 2

Figure 3–21. TIMERxTPR Register†

15 8	7 0
PSC	TDDR
R-0	R/W-0

Legend: R = Read access, W = write access, -0 = value after reset

Table 3–20. TIMERxTPR Register Field Descriptions

Bits	Field	Description
15–8	PSC	CPU-Timer Prescale Counter. These bits hold the current prescale count for the timer. For every timer clock source cycle that the PSCH:PSC value is greater than 0, the PSCH:PSC decrements by one. One timer clock (output of the timer prescaler) cycle after the PSCH:PSC reaches 0, the PSCH:PSC is loaded with the contents of the TDDRH:TDDR, and the timer counter register (TIMH:TIM) decrements by one. The PSCH:PSC is also reloaded whenever the timer reload bit (TRB) is set by software. The PSCH:PSC can be checked by reading the register, but it cannot be set directly. It must get its value from the timer divide-down register (TDDRH:TDDR). At reset, the PSCH:PSC is set to 0.
7–0	TDDR	CPU-Timer Divide-Down. Every (TDDRH:TDDR + 1) timer clock source cycles, the timer counter register (TIMH:TIM) decrements by one. At reset, the TDDRH:TDDR bits are cleared to 0. To increase the overall timer count by an integer factor, write this factor minus one to the TDDRH:TDDR bits. When the prescaler counter (PSCH:PSC) value is 0, one timer clock source cycle later, the contents of the TDDRH:TDDR reload the PSCH:PSC, and the TIMH:TIM decrements by one. TDDRH:TDDR also reloads the PSCH:PSC whenever the timer reload bit (TRB) is set by software.

† x = 0, 1, or 2

Table 3–21. TIMERxTPRH Register†

15	8	7	0
PSCH		TDDRH	
R-0		R/W-0	

Legend: R = Read access, W = write access, -0 = value after reset

Table 3–22. TIMERxTPRH Register Field Descriptions

Bits	Field	Description
15–8	PSCH	See description of TIMERxTPR.
7–0	TDDRH	See description of TIMERxTPR.

† x = 0, 1, or 2

Chapter 4

General-Purpose Input/Output (GPIO)

The GPIO MUX registers are used to select the operation of shared pins on the 281x devices. The pins can be individually selected to operate as Digital I/O or connected to Peripheral I/O signals (via the GPxMUX registers). If selected for Digital I/O mode, registers are provided to configure the pin direction (via the GPxDIR registers) and to qualify the input signal to remove unwanted noise (via the GPxQUAL) registers).

Topic **Page**

4.1 GPIO MUX

Table 4–1 lists the GPIO MUX Registers.

Table 4–1. GPIO MUX Registers†‡

Name	Address	Size (x16)	Register Description
GPAMUX	0x0000 70C0	1	GPIO A MUX Control Register
GPADIR	0x0000 70C1	1	GPIO A Direction Control Register
GPAQUAL	0x0000 70C2	1	GPIO A Input Qualification Control Register
Reserved	0x0000 70C3	1	
GPBMUX	0x0000 70C4	1	GPIO B MUX Control Register
GPBDIR	0x0000 70C5	1	GPIO B Direction Control Register
GPBQUAL	0x0000 70C6	1	GPIO B Input Qualification Control Register
Reserved	0x0000 70C7 – 0x0000 70CB	5	
GPDMUX	0x0000 70CC	1	GPIO D MUX Control Register
GPDDIR	0x0000 70CD	1	GPIO D Direction Control Register
GPDQUAL	0x0000 70CE	1	GPIO D Input Qualification Control Register
Reserved	0x0000 70CF	1	
GPEMUX	0x0000 70D0	1	GPIO E MUX Control Register
GPEDIR	0x0000 70D1	1	GPIO E Direction Control Register
GPEQUAL	0x0000 70D2	1	GPIO E Input Qualification Control Register
Reserved	0x0000 70D3	1	
GPFMUX	0x0000 70D4	1	GPIO F MUX Control Register
GPFDIR	0x0000 70D5	1	GPIO F Direction Control Register
Reserved	0x0000 70D6 – 0x0000 70D7	2	
GPGMUX	0x0000 70D8	1	GPIO G MUX Control Register

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are EALLOW protected. This prevents spurious writes from overwriting the contents and corrupting the system.

Table 4–1. GPIO MUX Registers†‡ (Continued)

Name	Address	Size (x16)	Register Description
GPGDIR	0x0000 70D9	1	GPIO G Direction Control Register
Reserved	0x0000 70DA – 0x0000 70DF	6	

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are EALLOW protected. This prevents spurious writes from overwriting the contents and corrupting the system.

If configured for Digital I/O mode, additional registers are provided for setting individual I/O signals (via the GPxSET registers), for clearing individual I/O signals (via the GPxCLEAR registers), for toggling individual I/O signals (via the GPxTOGGLE registers), or for reading/writing to the individual I/O signals (via the GPxDAT registers). Table 4–2 lists the GPIO Data Registers.

Table 4–2. GPIO Data Registers†‡

Name	Address	Size (x16)	Register Description
GPADAT	0x0000 70E0	1	GPIO A Data Register
GPASET	0x0000 70E1	1	GPIO A Set Register
GPACLEAR	0x0000 70E2	1	GPIO A Clear Register
GPATOGGLE	0x0000 70E3	1	GPIO A Toggle Register
GPBDAT	0x0000 70E4	1	GPIO B Data Register
GPBSET	0x0000 70E5	1	GPIO B Set Register
GPBCLEAR	0x0000 70E6	1	GPIO B Clear Register
GPBTOGGLE	0x0000 70E7	1	GPIO B Toggle Register
Reserved	0x0000 70E8 – 0x0000 70EB	4	
GPDDAT	0x0000 70EC	1	GPIO D Data Register
GPDSET	0x0000 70ED	1	GPIO D Set Register
GPDCLEAR	0x0000 70EE	1	GPIO D Clear Register
GPDTOGGLE	0x0000 70EF	1	GPIO D Toggle Register

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are NOT EALLOW protected and is typically accessed regularly by the user.

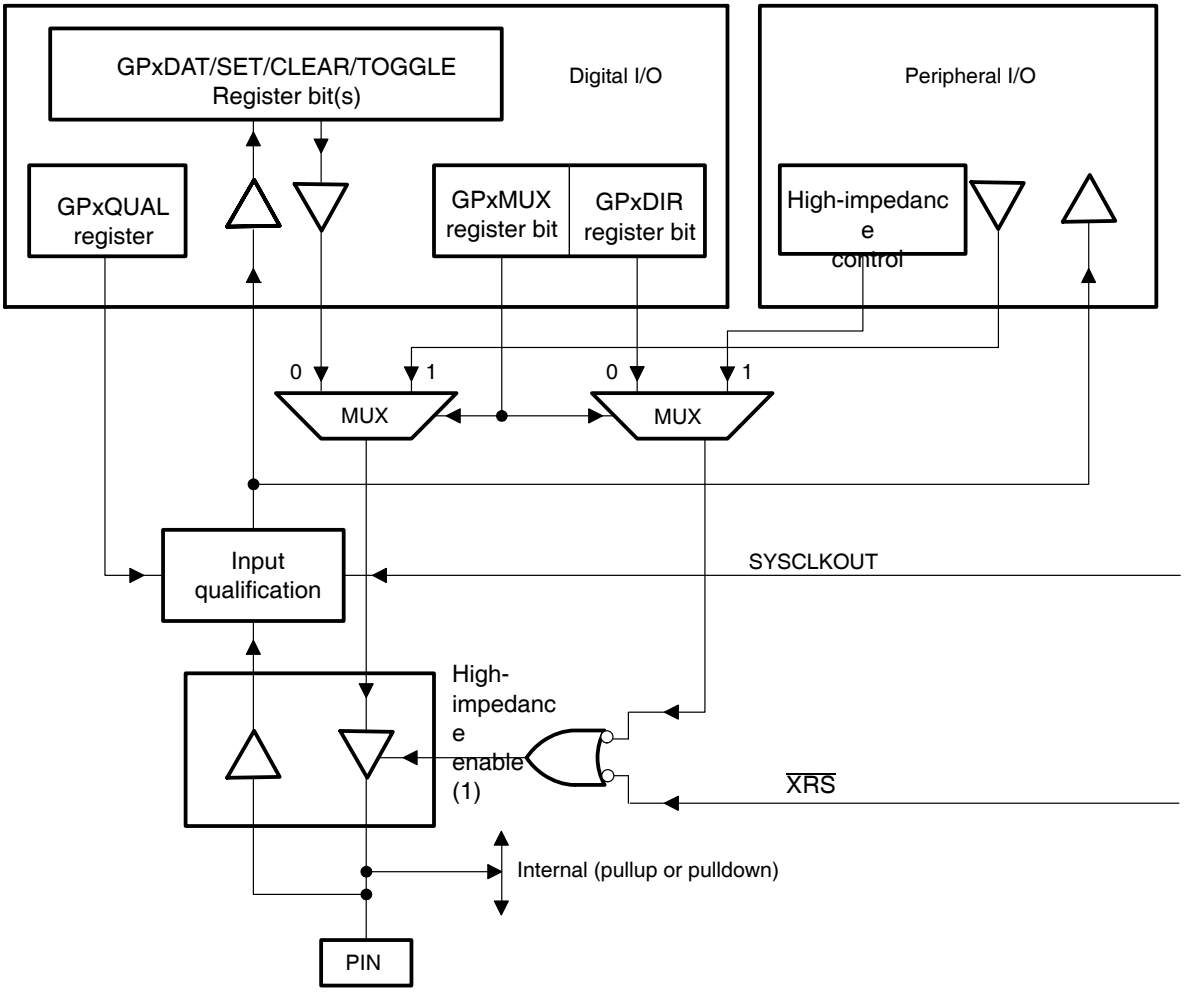
Table 4–2. GPIO Data Registers†‡ (Continued)

Name	Address	Size (x16)	Register Description
GPEDAT	0x0000 70F0	1	GPIO E Data Register
GPESET	0x0000 70F1	1	GPIO E Set Register
GPECLEAR	0x0000 70F2	1	GPIO E Clear Register
GPETOGGLE	0x0000 70F3	1	GPIO E Toggle Register
GPFDAT	0x0000 70F4	1	GPIO F Data Register
GPFSET	0x0000 70F5	1	GPIO F Set Register
GPFCLEAR	0x0000 70F6	1	GPIO F Clear Register
GPFTOGGLE	0x0000 70F7	1	GPIO F Toggle Register
GPGDAT	0x0000 70F8	1	GPIO G Data Register
GPGSET	0x0000 70F9	1	GPIO G Set Register
GPGCLEAR	0x0000 70FA	1	GPIO G Clear Register
GPGTOGGLE	0x0000 70FB	1	GPIO G Toggle Register
Reserved	0x0000 70FC – 0x0000 70FF	4	

† For reserved locations, read values are undefined and writes have no effect.
‡ These registers are NOT EALLOW protected and is typically accessed regularly by the user.

Figure 4–1 shows how the various register bits select the various modes of operation.

Figure 4–1. GPIO/Peripheral Pin MUXing



Notes: 1) Via the GPxDAT register, the state of any PIN can be read, regardless of the operating mode.

2) The GPxQUAL register specifies the qualification sampling period. The sampling window is six samples wide and the output is only changed when all samples are the same (all 0s or all 1s) as shown in Figure 4–3. This feature removes unwanted spikes from the input signal.

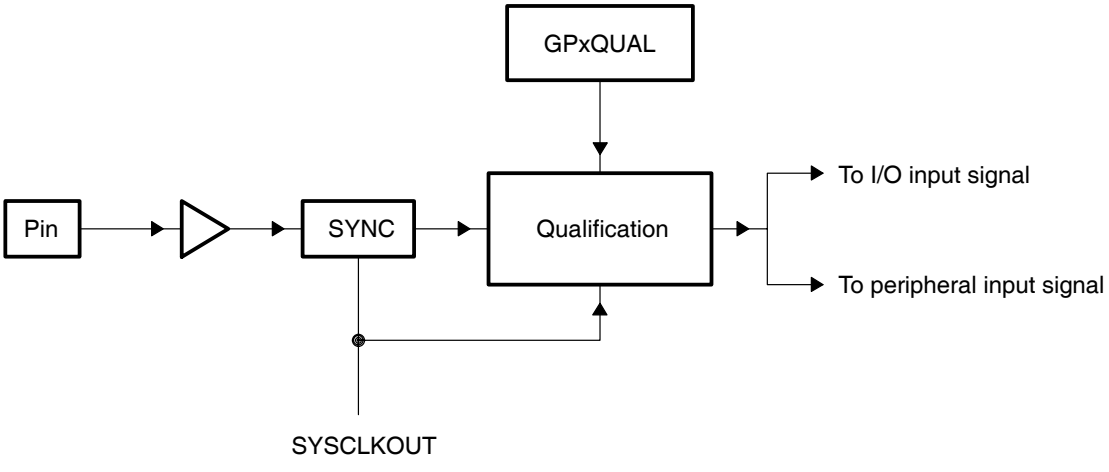
The input function of the GPIO pin and the input path to the peripheral are always enabled. It is the output function of the GPIO pin that is multiplexed with the output path of the primary (peripheral) function. Since the output buffer of a pin connects back to the input buffer, any GPIO signal present at the pin is propagated to the peripheral module as well. Therefore, when a pin is configured for GPIO operation, the corresponding peripheral functionality (and interrupt–generating capability) must be disabled. Otherwise, interrupts may be inadvertently triggered.

4.2 Input Qualification

Two types of input qualification are performed on inputs.

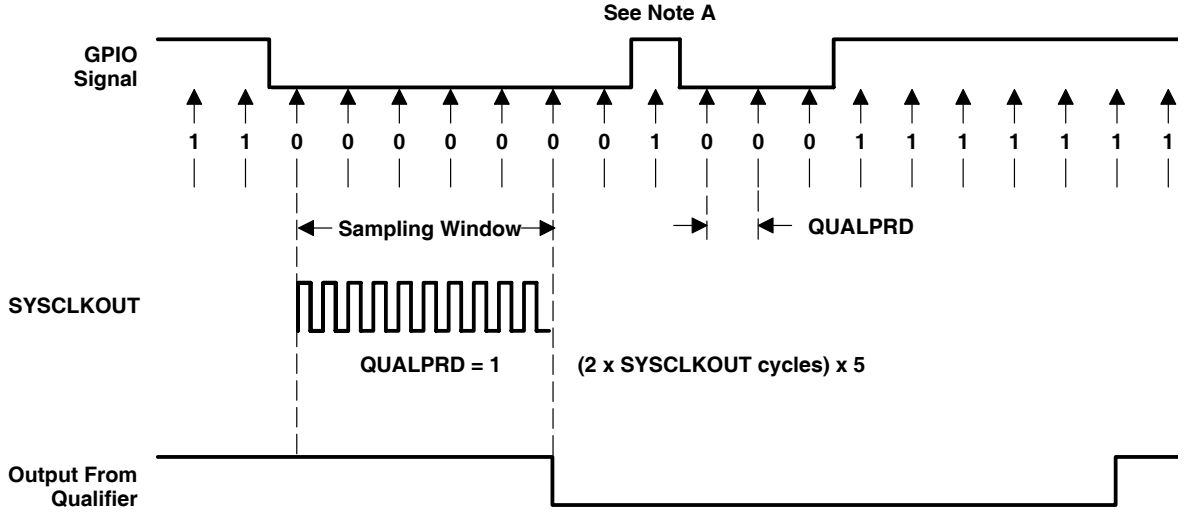
In type 1 qualification, the input signal is first synchronized to SYSCLKOUT and the signal qualified by the specified number of signals before the input is allowed to change. Figure 4–2 shows how type 1 input qualification is performed to eliminate unwanted noise.

Figure 4–2. Type 1 Input Qualification



To qualify the signal, the input is synchronized to SYSCLKOUT and then sampled at a regular period. The sampling period is specified by the GPxQUAL register. The sampling window is six samples wide and the input is changed only when all six samples are the same as shown in Figure 4–3. Because the incoming signal is asynchronous, there can be up to one SYSCLKOUT delay for synchronization before the qualification sampling window begins.

Figure 4–3. Input Qualifier Clock Cycles

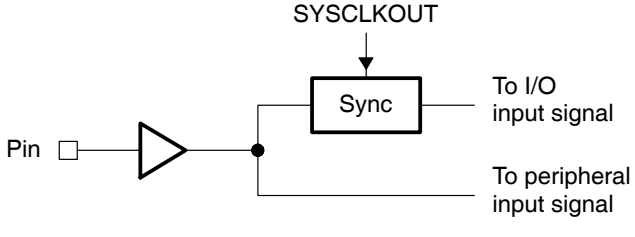


NOTES: A. This glitch will be ignored by the input qualifier. The QUALPRD bit field specifies the qualification sampling perio from 00 to 0xFF. Input qualification is not applicable when QUALPRD = 00. For any other value “n”, the qualifica period in 2n SYSCLKOUT cycles (i.e., at every 2n SYSCLKOUT cycle, the GPIO pin will be sampled). Six consec must be of the same value for a given input to be recognized.

B. For the qualifier to detect the change, the input should be stable for 10 SYSCLKOUT cycles or greater. In other wo should be stable for (5 x QUALPRD x 2) SYSCLKOUT cycles. This would ensure six sampling windows for detec Since external signals are driven asynchronously, an 11-SYSCLKOUT-wide pulse ensures reliable recognition.

Figure 4–4 shows type 2 qualification, where only the digital I/O signal is synchronized to the core clock (SYSCLKOUT). The peripheral signal connects directly to the pin without qualification or synchronization because some peripherals perform their own synchronization.

Figure 4–4. Type 2 Input Qualification



The type of input qualification used for each signal is indicated in the GPxMUX Register definitions in Section 4.3.

4.3 Register Functional Overview

Each GP I/O port is controlled by the MUX, direction, data, set, clear, and toggle registers. In the following section, the bit mapping for the GPxMUX and GPxDIR registers is shown. The corresponding data, set, clear, and toggle registers have identical bit to I/O mapping.

GPxMUX Registers

Each I/O port has one MUX register. The MUX registers are used to select between the corresponding peripheral operation and the I/O operation of each of the I/O pins. At reset all GP I/O pins are configured as I/O pins.

If GPxMUX.bit = 0, then the pin is configured as an I/O

If GPxMUX.bit = 1, then the pin is configured for the peripheral functionality

As shown in Figure 4–1, the input function of the I/O and the input path to the peripheral are always enabled. It is the output path of the peripheral that is MUXed with the output of the I/O. Therefore, when a pin is configured as an I/O pin, the corresponding peripheral function must be disabled. Otherwise, interrupts may be inadvertently triggered.

GPxDIR Registers

Each I/O port has one direction control register. The direction register controls whether the corresponding I/O pin is configured as an input or an output. At reset all GP I/O pins are configured as inputs.

If GPxDIR.bit = 0, then the pin is configured as an input

If GPxDIR.bit = 1, then the pin is configured as an output

On reset, the default value for all GPxMUX and GPxDIR register bits is 0. That is, at reset all I/O ports are configured as input pins. Before changing the direction of the port from input to output using the GPxDIR register bit, the current level of the pin is reflected in the GPxDAT register. The GPxDAT register is described later in this section. When the direction of the port is changed from input to output, the value already in the GPxDAT register is used to determine the state of the pin.

For example, if a pin has an internal pullup, then after reset the pullup would force the GPxDAT Register bit to a1 to reflect the current state of the pin. When the direction of the port changes from an input to an output the 1 already in the GPxDAT Register forces the pin to the same level. In this manner, the pin can be switched from input to output without a glitch.

GPxDAT Registers

Each I/O port has one data register. The data register is a R/W register that reflects the current state of the input I/O signal after qualification. Writing to the register sets the corresponding state of any I/O signal that is configured as an output.

If GPxDAT.bit = 0, AND the pin is an output, then pull the pin low

If GPxDAT.bit = 1, AND the pin is an output, then pull the pin high

When using the GPxDAT register to change the level of an output pin, you should be cautious not to accidentally change the level of another pin. For example, if you mean to change the level of GPIOA0 by writing to the GPADAT register bit 0, using a read-modify-write instruction. The problem can occur if another I/O port A signal changes the level between the read and the write stage of the instruction. If this happens, the signal level that changed is overwritten by the original value read during the read stage of the instruction. You can avoid this scenario by using the GPxSET, GPxCLEAR, and GPxTOGGLE registers instead.

GPxSET Registers

Each I/O port has one set register. The set registers are write-only registers that read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the set register will pull the corresponding pin high. Writing a 0 to a bit has no effect.

If GPxSET.bit = 0, ignored

If GPxSET.bit = 1, AND the pin is an output, then pull the pin high

GPxCLEAR Registers

Each I/O port has one clear register. The clear registers are write-only registers that read back 0. If the corresponding pin is configured as an output, then writing a 1 to that bit in the clear register will pull the corresponding pin low. Writing a 0 to a bit has no effect.

If GPxCLEAR.bit = 0, ignored

If GPxCLEAR.bit = 1, AND the pin is an output, then pull the pin low

GPxTOGGLE Registers

Each I/O port has one toggle register. The toggle registers are write-only registers that read back 0. If the corresponding pin is configured as an output,

then writing a 1 to that bit in the toggle register will pull the corresponding pin in the opposite direction. That is, if the output pin is low, writing a 1 to the corresponding bit in the toggle register will pull the pin high. Likewise, if the output pin is high, writing a 1 to the corresponding bit in the toggle register will pull the pin low. Writing a 0 to a bit has no effect.

If GPxTOGGLE.bit = 0, ignored

If GPxTOGGLE.bit = 1, AND the pin is an output, then pull the pin in the opposite direction

4.4 Register Bit to I/O Mapping

The following tables describe the register bit to I/O mapping on the 281x devices. For each port, the bit mapping is the same for each of the controlling registers: MUX, direction, set, clear, and toggle. The functionality of these registers is described in more detail in Section 4.3.

Some of the input signals are qualified using either type 1 or type 2 qualification. The type of qualification for each signal is indicated in the Input Qual column of the register bit to I/O mapping tables. More detailed information for the input qualification types is in Section 4.2.

Table 4–3 shows the MUX, direction, set, clear, and toggle register bit to I/O mapping for GPIO port A. Unused I/Os are reserved for future use.

Table 4–3. GPIO A Register Bit to I/O Mapping

Register Bit	Peripheral Name GPAMUX Bit = 1	GPIO Name GPAMUX Bit = 0	GPAMUX/DIR Type	Input Qual
EV-A Peripheral				
0	PWM1 (O)	GPIOA0	R/W-0	Type 1
1	PWM2 (O)	GPIOA1	R/W-0	Type 1
2	PWM3 (O)	GPIOA2	R/W-0	Type 1
3	PWM4 (O)	GPIOA3	R/W-0	Type 1
4	PWM5 (O)	GPIOA4	R/W-0	Type 1
5	PWM6 (O)	GPIOA5	R/W-0	Type 1
6	T1PWM_T1CMP (0)	GPIOA6	R/W-0	Type 1
7	T2PWM_T2CMP (0)	GPIOA7	R/W-0	Type 1
8	CAP1_QEP1 (I)	GPIOA8	R/W-0	Type 1
9	CAP2_QEP2 (I)	GPIOA9	R/W-0	Type 1
10	CAP3_QEPI1 (I)	GPIOA10	R/W-0	Type 1
11	TDIRA (I)	GPIOA11	R/W-0	Type 1
12	TCLKINA (I)	GPIOA12	R/W-0	Type 1
13	$\overline{\text{C1TRIP}}$ (I)	GPIOA13	R/W-0	Type 1

Note: GPAMUX and GPADIR are EALLOW–protected registers.

Table 4–3. GPIO A Register Bit to I/O Mapping (Continued)

Register Bit	Peripheral Name GPAMUX Bit = 1	GPIO Name GPAMUX Bit = 0	GPAMUX/DIR Type	Input Qual
14	$\overline{\text{C2TRIP}}$ (I)	GPIOA14	R/W-0	Type 1
15	$\overline{\text{C3TRIP}}$ (I)	GPIOA15	R/W-0	Type 1

Note: GPAMUX and GPADIR are EALLOW–protected registers.

Figure 4–5. GPIO A Input Qualification Control (GPAQUAL) Register

15	8	7	0
Reserved		QUALPRD	
R-0		R/W-0	

Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 4–4. GPIO A Input Qualification Control (GPAQUAL) Register Field Descriptions

Bits	Field	Description
15–8	Reserved	
7–0	QUALPRD	Specifies the qualification sampling period: 0x00 No qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = 2 SYSCLKOUT cycles 0x02 QUALPRD = 4 SYSCLKOUT cycles … 0xFF QUALPRD = 510 SYSCLKOUT cycles

Table 4–5 shows the MUX, direction, set, clear, and toggle register bit to I/O mapping for GPIO port B. Unused I/Os are reserved for future use.

Table 4–5. GPIO B Register Bit to I/O Mapping

Register Bit	Peripheral Name GPBMUX Bit = 1	GPIO Name GPBMUX Bit = 0	GPBMUX/DIR Type	Input Qual
EV-B Peripheral				
0	PWM7 (O)	GPIOB0	R/W–0	Type 1
1	PWM8 (O)	GPIOB1	R/W-0	Type 1
2	PWM9 (O)	GPIOB2	R/W-0	Type 1
3	PWM10 (O)	GPIOB3	R/W-0	Type 1

Note: GPBMUX and GPBDIR are EALLOW–protected registers.

Table 4–5. GPIO B Register Bit to I/O Mapping(Continued)

Register Bit	Peripheral Name GPBMUX Bit = 1	GPIO Name GPBMUX Bit = 0	Type	Input Qual
4	PWM11 (O)	GPIOB4	R/W-0	Type 1
5	PWM12 (O)	GPIOB5	R/W-0	Type 1
6	T3PWM_T3CMP (0)	GPIOB6	R/W-0	Type 1
7	T4PWM_T4CMP (0)	GPIOB7	R/W-0	Type 1
8	CAP4_QEP3 (I)	GPIOB8	R/W-0	Type 1
9	CAP5_QEP4 (I)	GPIOB9	R/W-0	Type 1
10	CAP6_QEPI2 (I)	GPIOB10	R/W-0	Type 1
11	TDIRB (I)	GPIOB11	R/W-0	Type 1
12	TCLKINB (I)	GPIOB12	R/W-0	Type 1
13	$\overline{\text{C4TRIP}}$ (I)	GPIOB13	R/W-0	Type 1
14	$\overline{\text{C5TRIP}}$ (I)	GPIOB14	R/W-0	Type 1
15	$\overline{\text{C6TRIP}}$ (I)	GPIOB15	R/W-0	Type 1

Note: GPBMUX and GPBDIR are EALLOW–protected registers.

Figure 4–6. GPIO B Input Qualification Control (GPBQUAL) Register

15	8	7	0
Reserved		QUALPRD	
R-0		R/W-0	

Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 4–6. GPIO B Input Qualification Control (GPBQUAL) Register Field Descriptions

Bits	Field	Description
15–8	Reserved	
7–0	QUALPRD	Specifies the qualification sampling period: 0x00 No qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = 2 SYSCLKOUT cycles 0x02 QUALPRD = 4 SYSCLKOUT cycles … 0xFF QUALPRD = 510 SYSCLKOUT cycles

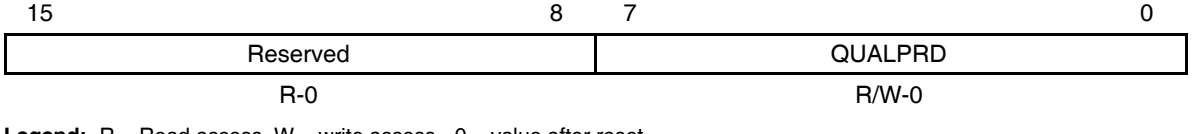
Table 4–7 shows the MUX, direction, set, clear, and toggle register bit to I/O mapping for GPIO port D. Unused I/Os are reserved for future use.

Table 4–7. GPIO D Register Bit to I/O Mapping

Register Bit	Peripheral Name GPDMUX Bit = 1	GPIO Name GPDMUX Bit = 0	GPDMUX/DIR Type	Input Qual
EV-A Peripheral				
0	$\overline{\text{T1CTRIP\_PDPINTA}}$ (I)	GPIOD0	R/W-0	Type 1
1	$\overline{\text{T2CTRIP}}$ (I)	GPIOD1	R/W-0	Type 1
2	Reserved	Reserved	R-0	–
3	Reserved	Reserved	R-0	–
4	Reserved	Reserved	R-0	–
EV-B Peripheral				
5	$\overline{\text{T3CTRIP\_PDPINTB}}$ (I)	GPIOD5	R/W-0	Type 1
6	$\overline{\text{T4CTRIP}}$ (I)	GPIOD6	R/W-0	Type 1
7	Reserved	Reserved	R-0	–
8	Reserved	Reserved	R-0	–
9	Reserved	Reserved	R-0	–
10	Reserved	Reserved	R-0	–
11	Reserved	Reserved	R-0	–
12	Reserved	Reserved	R-0	–
13	Reserved	Reserved	R-0	–
14	Reserved	Reserved	R-0	–
15	Reserved	Reserved	R-0	–

Note: GPDMUX and GPDDIR are EALLOW–protected registers.

Figure 4–7. GPIO D Input Qualification Control (GPDQUAL) Register



Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 4–8. GPIO D Input Qualification Control (GPDQUAL) Register Field Descriptions

Bits	Field	Description
15–8	Reserved	
7–0	QUALPRD	Specifies the qualification sampling period: 0x00 No qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = 2 SYSCLKOUT cycles 0x02 QUALPRD = 4 SYSCLKOUT cycles . 0xFF QUALPRD = 510 SYSCLKOUT cycles

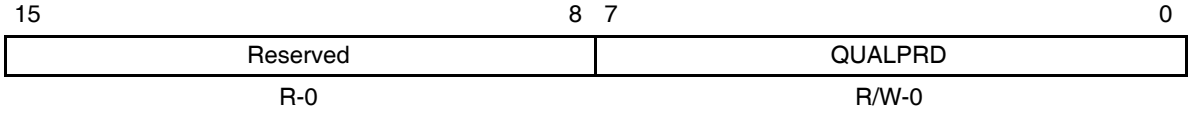
Table 4–9 shows the MUX, direction, set, clear, and toggle register bit to I/O mapping for GPIO port E. Unused I/Os are reserved for future use.

Table 4–9. GPIO E MUX Register Bit to I/O Mapping

Register Bit	Peripheral Name GPEMUX Bit = 0	GPIO Name GPEMUX Bit = 0	GPEMUXDIR Type	Input Qual
Interrupts				
0	XINT1_$\overline{\text{XBIO}}$ (I)	GPIOE0	R/W-0	Type 1
1	XINT2_ADCSOC (I)	GPIOE1	R/W-0	Type 1
2	XNMI_XINT13 (I)	GPIOE2	R/W-0	Type 1
3	Reserved	Reserved	R-0	–
4	Reserved	Reserved	R-0	–
5	Reserved	Reserved	R-0	–
6	Reserved	Reserved	R-0	–
7	Reserved	Reserved	R-0	–
8	Reserved	Reserved	R-0	–
9	Reserved	Reserved	R-0	–
10	Reserved	Reserved	R-0	–
11	Reserved	Reserved	R-0	–
12	Reserved	Reserved	R-0	–
13	Reserved	Reserved	R-0	–
14	Reserved	Reserved	R-0	–
15	Reserved	Reserved	R-0	–

Note: GPEMUX and GPEDIR are EALLOW-protected registers.

Figure 4–8. GPIO E Input Qualification Control (GPEQUAL) Register



Legend: R = Read access, W = write access, -0 = value after reset

Note: EALLOW-protected register

Table 4–10. GPIO E Input Qualification Control (GPEQUAL) Register Field Descriptions

Bits	Field	Description
15–8	Reserved	
7–0	QUALPRD	Specifies the qualification sampling period: 0x00 No qualification (just SYNC to SYSCLKOUT) 0x01 QUALPRD = SYSCLKOUT/2 0x02 QUALPRD = SYSCLKOUT/4 . 0xFF QUALPRD = SYSCLKOUT/510

Table 4–11 shows the MUX, direction, set, clear, and toggle register bit to I/O mapping for GPIO port E. Unused I/Os are reserved for future use.

Table 4–11. GPIO F Register to Bit I/O Mapping

Register Bit	Peripheral Name GPFMUX Bit = 1	GPIO Name GPFMUX Bit = 0	GPFMUX/DIR Type	Input Qual
SPI Peripheral				
0	SPISIMO (O)	GPIOF0	R/W-0	Type 2
1	SPISOMI (I)	GPIOF1	R/W-0	Type 2
2	SPICLK (I/O)	GPIOF2	R/W-0	Type 2
3	SPISTE (I/O)	GPIOF3	R/W-0	Type 2
SCIA Peripheral				
4	SCITXDA (O)	GPIOF4	R/W-0	Type 2
5	SCIRXDA (I)	GPIOF5	R/W-0	Type 2
CAN Peripheral				
6	CANTX (O)	GPIOF6	R/W-0	Type 2
7	CANRX (I)	GPIOF7	R/W-0	Type 2

Note: GPFMUX and GPFDIR are EALLOW-protected registers.

Table 4–11. GPIO F Register to Bit I/O Mapping (Continued)(Continued)

Register Bit	GPIO Name GPFMUX Bit = 1	GPFMUX Bit = 0	Type	Input Qual
McBSP Peripheral				
8	MCLKX (I/O)	GPIOF8	R/W-0	Type 2
9	MCLKR (I/O)	GPIOF9	R/W-0	Type 2
10	MFSX (I/O)	GPIOF10	R/W-0	Type 2
11	MFSR (I/O)	GPIOF11	R/W-0	Type 2
12	MDX (O)	GPIOF12	R/W-0	Type 2
13	MDR (I)	GPIOF13	R/W-0	Type 2
XF CPU Output Signal				
14	XF (0)	GPIOF14	R/W-0	Type 2
15	Reserved	Reserved	R-0	–

Note: GPFMUX and GPFDIR are EALLOW-protected registers.

Table 4–12 shows the MUX, direction, set, clear, and toggle register bit to I/O mapping for GPIO port G. Unused I/Os are reserved for future use.

Table 4–12. GPIO G Register to Bit I/O Mapping

Register Bit	Peripheral Name GPGMUX Bit = 1	GPIO Name GPGMUX Bit = 0	GPGMUX/DIR Type	Input Qual
0	Reserved	Reserved	R-0	–
1	Reserved	Reserved	R-0	–
2	Reserved	Reserved	R-0	–
3	Reserved	Reserved	R-0	–
SCI-B Peripheral				
4	SCITXDB (O)	GPIOG4	R/W-0	Type 2
5	SCIRXDB (I)	GPIOG5	R/W-0	Type 2
6	Reserved	Reserved	R-0	–
7	Reserved	Reserved	R-0	–
8	Reserved	Reserved	R-0	–

Note: GPGMUX and GPGDIR are EALLOW-protected registers.

Table 4–12. GPIO G Register to Bit I/O Mapping (Continued)

Register Bit	Peripheral Name GPGMUX Bit = 1	GPIO Name GPGMUX Bit = 0	GPGMUX/DIR Type	Input Qual
9	Reserved	Reserved	R-0	–
10	Reserved	Reserved	R-0	–
11	Reserved	Reserved	R-0	–
12	Reserved	Reserved	R-0	–
13	Reserved	Reserved	R-0	–
14	Reserved	Reserved	R-0	–
15	Reserved	Reserved	R-0	–

Note: GPGMUX and GPGDIR are EALLOW-protected registers.

Chapter 5

Peripheral Frames

This chapter describes the peripheral frames. It also describes the device emulation registers.

Topic **Page**

5.1 Peripheral Frame Registers

The 281x devices contain three peripheral register spaces. The spaces are categorized as follows:

- ❑ Peripheral Frame 0: These are peripherals that are mapped directly to the CPU memory bus. See Table 5–1.
- ❑ Peripheral Frame 1: These are peripherals that are mapped to the 32-bit peripheral bus. See Table 5–2.
- ❑ Peripheral Frame 2: These are peripherals that are mapped to the 16-bit peripheral bus. See Table 5–3.

Table 5–1. Peripheral Frame 0 Registers

Name	Address Range	Size (x16)	Access Type†
Device Emulation Registers	0x0000 0880 0x0000 09FF	384	EALLOW protected
Reserved	0x0000 0A00 0x0000 0B00	128	
FLASH Registers‡	0x0000 0A80 0x0000 0ADF	96	EALLOW protected CSM Protected
Code Security Module Registers	0x0000 0AE0 0x0000 0AEF	16	EALLOW protected
Reserved	0x0000 0AF0 0x0000 0B1F	48	
XINTF Registers	0x0000 0B20 0x0000 0B3F	32	Not EALLOW protected
Reserved	0x0000 0B40 0x0000 0BFF	192	
CPU-TIMER0/1/2 Registers	0x0000 0C00 0x0000 0C3F	64	Not EALLOW protected
Reserved	0x0000 0C40 0x0000 0CDF	160	
PIE Registers	0x0000 0CE0 0x0000 0CFF	32	Not EALLOW protected

† If registers are EALLOW protected, you cannot perform writes until you execute the EALLOW instruction. The EDIS instruction disables writes to prevent stray code or pointers from corrupting register contents.

‡ The flash registers are also protected by the Code Security Module (CSM).

Table 5–1. Peripheral Frame 0 Registers (Continued)

Name	Address Range	Size (x16)	Access Type†
PIE Vector Table	0x0000 0D00 0x0000 0DFF	256	EALLOW protected
Reserved	0x0000 0E00 0x0000 0FFF	512	

† If registers are EALLOW protected, you cannot perform writes until you execute the EALLOW instruction. The EDIS instruction disables writes to prevent stray code or pointers from corrupting register contents.
‡ The flash registers are also protected by the Code Security Module (CSM).

Table 5–2. Peripheral Frame 1 Registers

Name†	Address Range	Size (x16)	Access Type
eCAN Registers	0x00 6000 0x00 60FF	256 (128 x 32)	Some eCAN control registers (and selected bits in other eCAN control registers) are EALLOW-protected. See Section 5.2 on page 5-5.
eCAN Mailbox RAM	0x00 6100 0x00 61FF	256 (128 x 32)	Not EALLOW-protected
Reserved	0x00 6200 0x00 6FFF	3584	

† Peripheral Frame 1 allows 16-bit and 32-bit accesses. All 32-bit accesses are aligned to even address boundaries.

Table 5–3. Peripheral Frame 2 Registers†

Name	Address Range	Size (x16)	Access Type
Reserved	0x0000 7000 0x0000 700F	16	
System Control Registers	0x0000 7010 0x0000 702F	32	EALLOW Protected
Reserved	0x0000 7030 0x0000 703F	16	
SPI Registers	0x0000 7040 0x0000 704F	16	Not EALLOW Protected
SCI-A Registers	0x0000 7050 0x0000 705F	16	Not EALLOW Protected
Reserved	0x0000 7060 0x0000 706F	16	

† Peripheral Frame 2 only allows 16-bit accesses. All 32-bit accesses are ignored (invalid data can be returned or written).

Table 5–3. Peripheral Frame 2 Registers† (Continued)

Name	Address Range	Size (x16)	Access Type
External Interrupt Registers	0x0000 7070 0x0000 707F	16	Not EALLOW Protected
Reserved	0x0000 7080 0x0000 70BF	64	
GPIO MUX Registers	0x0000 70C0 0x0000 70DF	32	EALLOW Protected
GPIO Data Registers	0x0000 70E0 0x0000 70FF	32	Not EALLOW Protected
ADC Registers	0x0000 7100 0x0000 711F	32	Not EALLOW Protected
Reserved	0x0000 7120 0x0000 73FF	736	
EV-A Registers	0x0000 7400 0x0000 743F	64	Not EALLOW Protected
Reserved	0x0000 7440 0x0000 74FF	192	
EV-B Registers	0x0000 7500 0x0000 753F	64	Not EALLOW Protected
Reserved	0x0000 7540 0x0000 774F	528	
SCI-B Registers	0x0000 7750 0x0000 775F	16	Not EALLOW Protected
Reserved	0x0000 7760 0x0000 77FF	160	
McBSP Registers	0x0000 7800 0x0000 783F	64	Not EALLOW Protected
Reserved	0x0000 7840 0x0000 7FFF	1984	

† Peripheral Frame 2 only allows 16-bit accesses. All 32-bit accesses are ignored (invalid data can be returned or written).

5.2 EALLOW Protected Registers

Several control registers on the 281x devices are protected from spurious CPU writes by the EALLOW protection mechanism. The EALLOW bit in status register 1 (ST1) indicates if the state of protection as shown in Table 5–4.

Table 5–4. Access to EALLOW Protected Registers

EALLOW Bit	CPU Writes	CPU Reads	JTAG Writes	JTAG Reads
0	Ignored	Allowed	Allowed†	Allowed
1	Allowed	Allowed	Allowed	Allowed

† The EALLOW bit is overridden via the JTAG port, allowing full access of protected registers during debug from the Code Composer Studio interface.

At reset the EALLOW bit is cleared enabling EALLOW protection. While protected, all writes to protected registers by the CPU are ignored and only CPU reads, JTAG reads, and JTAG writes are allowed. If this bit is set, by executing the EALLOW instruction, then the CPU is allowed to write freely to protected registers. After modifying registers, they can once again be protected by executing the EDI instruction to clear the EALLOW bit.

The following registers are EALLOW protected:

- ❑ Device Emulation Registers
- ❑ Flash Registers
- ❑ CSM Registers
- ❑ PIE Vector Table
- ❑ System Control Registers
- ❑ GPIO MUX Registers
- ❑ Certain eCAN Registers

Table 5–5. EALLOW Protected Device Emulation Registers

Name	Address Range	Size (x16)	Description
DEVICECNF	0x0000 0880 0x0000 0881	2	Device Configuration Register
PROTSTART	0x0000 0884	1	Block Protection Start Address Register
PROTRANGE	0x0000 0885	1	Block Protection Range Address Register

Table 5–6. EALLOW Protected Flash/OTP Configuration Registers

Name	Address	Size (x16)	Description
		Configuration Registers	
FOPT	0x0000–0A80	1	Flash Option Register
FPWR	0x0000–0A82	1	Flash Power Modes Register
FSTATUS	0x0000–0A83	1	Status Register
FSTDBYWAIT	0x0000–0A84	1	Flash Sleep To Standby Wait State Register
FACTIVEWAIT	0x0000–0A85	1	Flash Standby To Active Wait State Register
FBANKWAIT	0x0000–0A86	1	Flash Read Access Wait State Register
FOTPWAIT	0x0000–0A87	1	OTP Read Access Wait State Register

Table 5–7. EALLOW Protected Code Security Module (CSM) Registers

Register Name	Memory Address	Reset Values	Register Description
KEY Registers – Accessible by the User			
KEY0	0x0000 – 0AE0	0xFFFF	Low word of the 128-bit KEY register
KEY1	0x0000 – 0AE1	0xFFFF	Second word of the 128-bit KEY register
KEY2	0x0000 – 0AE2	0xFFFF	Third word of the 128-bit KEY register
KEY3	0x0000 – 0AE3	0xFFFF	Fourth word of the 128-bit KEY register
KEY4	0x0000 – 0AE4	0xFFFF	Fifth word of the 128-bit KEY register
KEY5	0x0000 – 0AE5	0xFFFF	Sixth word of the 128-bit KEY register
KEY6	0x0000 – 0AE6	0xFFFF	Seventh word of the 128-bit KEY register
KEY7	0x0000 – 0AE7	0xFFFF	High word of the 128-bit KEY register
CSMSCR	0x0000 – 0AEF		CSM status and control register

Table 5–8. EALLOW Protected PIE Vector Table

Name	Address	Size (x16)	Description	CPU Priority	PIE Group Priority
Not used	0x0000 0D00	2	Reserved	–	–
	0x0000 0D02				
	0x0000 0D04				
	0x0000 0D06				
	0x0000 0D08				
	0x0000 0D0A				
	0x0000 0D0C				
	0x0000 0D0E				
	0x0000 0D10				
	0x0000 0D12				
	0x0000 0D14				
	0x0000 0D16				
	0x0000 0D18				
INT13	0x0000 0D1A	2	External Interrupt 13 (XINT13) or CPU-Timer 1 (for RTOS use)	17	–
INT14	0x0000 0D1C	2	CPU-Timer 2 (for RTOS use)	18	–
DATALOG	0x0000 0D1E	2	CPU Data Logging Interrupt	19 (lowest)	–
RTOSINT	0x0000 0D20	2	CPU Real-Time OS Interrupt	4	–
EMUINT	0x0000 0D22	2	CPU Emulation Interrupt	2	–
NMI	0x0000 0D24	2	External Non-Maskable Interrupt	3	–
ILLEGAL	0x0000 0D26	2	Illegal Operation	–	–
USER0	0x0000 0D28	2	User-Defined Trap	–	–
.	.	.	.	.	.
USER11	0x0000 0D3E	2	User-Defined Trap	–	–
INT1.1	0x0000 0D40	2	Group 1 Interrupt Vectors	5	1 (highest)
.	.	.			.
INT1.8	0x0000 0D4E	2			8 (lowest)
.	.	.	Group 2 Interrupt Vectors	6	
.	.	.	to Group 11 Interrupt Vectors	to	
.	.	.		15	
INT12.1	0x0000 0DF0	2	Group 12 Interrupt Vectors	16	1 (highest)
.	.	.			.
INT12.8	0x0000 0DFE	2			8 (lowest)

Table 5–9. EALLOW Protected PLL, Clocking, Watchdog, and Low-Power Mode Registers

Name	Address	Size (x16)	Description
HISPCP	0x0000 701A	1	High-Speed Peripheral Clock Prescaler Register for HSPCLK Clock
LOSPCP	0x0000 701B	1	Low-Speed Peripheral Clock Prescaler Register for HSPCLK Clock
PCLKCR	0x0000 701C	1	Peripheral Clock Control Register
LPMCR0	0x0000 701E	1	Low Power Mode Control Register 0
LPMCR1	0x0000 701F	1	Low Power Mode Control Register 1
PLLCR	0x0000 7021	1	PLL Control Register‡
SCSR	0x0000 7022	1	System Control and Status Register
WDCNTR	0x0000 7023	1	Watchdog Counter Register
WDKEY	0x0000 7025	1	Watchdog Reset Key Register
WDCR	0x0000 7029	1	Watchdog Control Register

Table 5–10. EALLOW Protected GPIO MUX Registers

Name	Address	Size (x16)	Register Description
GPAMUX	0x0000 70C0	1	GPIO A MUX Control Register
GPADIR	0x0000 70C1	1	GPIO A Direction Control Register
GPAQUAL	0x0000 70C2	1	GPIO A Input Qualification Control Register
GPBMUX	0x0000 70C4	1	GPIO B MUX Control Register
GPBDIR	0x0000 70C5	1	GPIO B Direction Control Register
GPBQUAL	0x0000 70C6	1	GPIO B Input Qualification Control Register
GPDMUX	0x0000 70CC	1	GPIO D MUX Control Register
GPDDIR	0x0000 70CD	1	GPIO D Direction Control Register
GPDQUAL	0x0000 70CE	1	GPIO D Input Qualification Control Register
GPEMUX	0x0000 70D0	1	GPIO E MUX Control Register
GPEDIR	0x0000 70D1	1	GPIO E Direction Control Register
GPEQUAL	0x0000 70D2	1	GPIO E Input Qualification Control Register

Table 5–10. EALLOW Protected GPIO MUX Registers(Continued)

Name	Address	Size (x16)	Register Description
GPFMUX	0x0000 70D4	1	GPIO F MUX Control Register
GPFDIR	0x0000 70D5	1	GPIO F Direction Control Register
GPGMUX	0x0000 70D8	1	GPIO G MUX Control Register
GPGDIR	0x0000 70D9	1	GPIO G Direction Control Register

Table 5–11. EALLOW Protected eCAN Registers

Name	Address	Size (x16)	Description
CANMC	0x0000–6014	2	Master Control Register†
CANBTC	0x0000–6016	2	Bit Timing Configuration Register‡
CANGIM	0x0000–6020	2	Global Interrupt Mask Register§
CANMIM	0x0000–6024	2	Mailbox Interrupt Mask Register
CANTSC	0x0000–602E	2	Time Stamp Counter
CANTIOC	0x0000–602A	1	I/O Control Register for CANTXA Pin¶
CANRIOC	0x0000–602C	1	I/O Control Register for CANRXA Pin#

† Only bits CANMC[15:9] and [7:6] are protected
‡ Only bits BCR[23:16] and [10:0] are protected
§ Only bits CANGIM[17:16] , [14:8], and [2:0] are protected
¶ Only IOCONT1[3] is protected
# Only IOCONT2[3] is protected

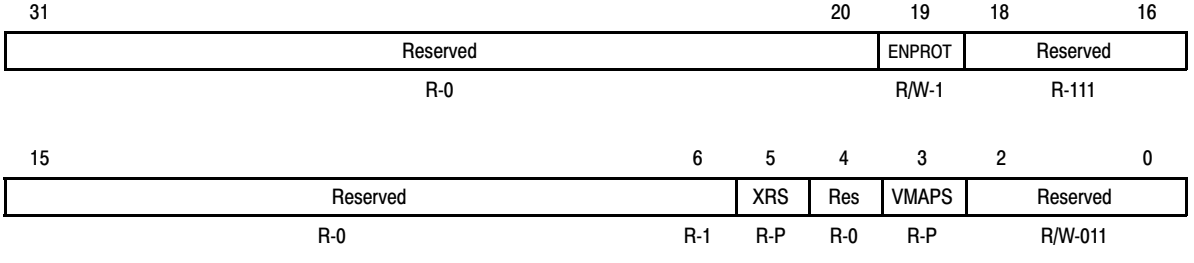
5.3 Device Emulation Registers

These registers are used to control the protection mode of the C28x CPU and to monitor some critical device signals. The registers are defined in Table 5–12.

Table 5–12. Device Emulation Registers

Name	Address Range	Size (x16)	Description
DEVICECNF	0x0000 0880 0x0000 0881	2	Device Configuration Register
PARTID	0x0000 0882	1	Part ID Register
REVID	0x0000 0883	1	Revision ID Register
PROTSTART	0x0000 0884	1	Block Protection Start Address Register
PROTRANGE	0x0000 0885	1	Block Protection Range Address Register
Reserved	0x0000 0886 0x0000 09FF	378	

Figure 5–1. Device Configuration (DEVICECNF) Register



Legend: R = Read, W = Write, P = pin value after reset, -n = reset value

Note: EALLOW-protected register

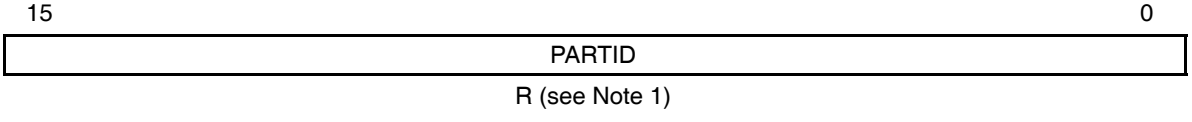
Table 5–13. Device Configuration (DEVICECNF) Register Field Descriptions

Bits	Field	Description
31–20	Reserved	
19	ENPROT	Enable Write-Read Protection Mode Bit
		0 Disables write-read protection mode
		1 Enables write-read protection as specified by the PROTSTART and PROTRANGE registers

Table 5–13. Device Configuration (DEVICECNF) Register Field Descriptions (Continued)

Bits	Field	Description
18–6	Reserved	
5	XRS	Reset Input Signal Status. This is connected directly to the $\overline{\text{XRS}}$ input pin.
4	Reserved	
3	VMAPS	VMAP Configure Status. This indicates the status of VMAP.
2–0	Reserved	

Figure 5–2. Part ID Register



1) The reset value depends on the device as indicated in the register description.

Table 5–14. Part ID Register Field Descriptions

Bits	Field	Description
15–0	PARTID	These 16 bits specify the part number for the of the device as follows: 0x0001 F281x (see Note 2) 0x0002 F281x (see Note 2) 0x0003 C281x 0x002C F2801 0x0034 F2806 0x003C F2808 All other values are reserved.

2) On Rev C and higher of the F281x devices, the ability to distinguish between F281x devices was removed. Hence, a value of 0x0001 or 0x0002 may represent any F281x device.

Figure 5–3. DEVICEID Register

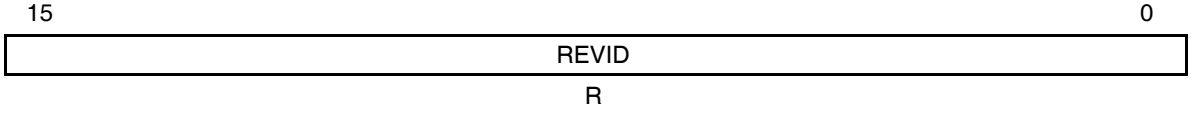


Table 5–15. DEVICEID Register Field Descriptions

Bits	Field	Description	
15–0	REVID	These 16 bits specify the silicon revision number for the particular part. This number always starts with 0x0000 on the first revision of the silicon and is incremented on any subsequent revisions.	
		0x0000	Revision 0 (for first silicon)
		0x0001	Revision A
		0x0002	Revision B and so forth

5.4 Write-Followed-by-Read Protection

The PROTSTART and PROTRANGE registers set the memory address range for which CPU “write” followed by “read” operations are protected (operations occur in sequence rather then in their natural pipeline order). This is necessary protection for certain peripheral operations.

Example: The following lines of code perform a write to register 1 (REG1) location and then the next instruction performs a read from Register 2 (REG2) location. On the processor memory bus, with block protection disabled, the read operation is issued before the write as shown:

```
MOV    @REG1,AL      -----  +
TBIT   @REG2,#BIT_X  -----  |----> Read
                     +----> Write
```

If block protection is enabled, then the read is stalled until the write occurs as shown:

```
MOV    @REG1,AL      -----  +
TBIT   @REG2,#BIT_X  ---    +      |
                     |      +----> Write
                     +------> Read
```

NOTE: The C28x CPU automatically protects writes followed by reads to the same memory address. The protection mechanism described above is for cases where the address is not the same, but within a given region in memory (as defined by the PROTSTART and PROTRANGE registers).

Table 5–16. PROTSTART and PROTRANGE Registers

Name	Address	Size	Type	Reset	Description
PROTSTART	0x0000 0884	16	R/W	0x0100†	The PROTSTART register sets the starting address relative to the 16 most significant bits of the processors lower 22-bit address reach. Hence, the smallest resolution is 64 words.
PROTRANGE	0x0000 0885	16	R/W	0x00FF†	The PROTRANGE register sets the block size (from the starting address), starting with 64 words and incrementing by binary multiples (64, 128, 256, 512, 1K, 2K, 4K, 8K, 16K, ..., 2M).

† The default values of these registers on reset are selected to cover the Peripheral Frame 1, Peripheral Frame 2, and XINTF Zone 1 areas of the memory map (address range 0x0000 4000 to 0x0000 8000).

Table 5–17. PROTSTART Valid Values†

Start Address	Register Value	Register Bits															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
0x0000 0000	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0x0000 0040	0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
0x0000 0080	0x0002	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
0x0000 00C0	0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
0x003F FF00	0xFFFC	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0
0x003F FF40	0xFFFD	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
0x003F FF80	0xFFFE	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
0x003F FFC0	0xFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

† The quickest way to calculate register value is to divide the desired block starting address by 64.

Table 5–18. PROTRANGE Valid Values‡

Block Size	Register Value	Register Bits															
		15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
64	0x0000	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
128	0x0001	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
256	0x0003	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1
512	0x0007	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1
1K	0x000F	0	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1
2K	0x001F	0	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1
4K	0x003F	0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1
8K	0x007F	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1
16K	0x00FF	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
32K	0x01FF	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1
64K	0x03FF	0	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1
128K	0x07FF	0	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1
256K	0x0FFF	0	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1
512K	0x1FFF	0	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1
1M	0x3FFF	0	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1
2M	0x7FFF	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
4M	0xFFFF	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1

‡ Not all register values are valid. The PROTSTART address value must be a multiple of the range value. For example: if the block size is set to 4K, then the start address can only be at any 4K boundary.

Chapter 6

Peripheral Interrupt Expansion (PIE)

The peripheral interrupt expansion (PIE) block multiplexes numerous interrupt sources into a smaller set of interrupt inputs. The PIE block can support 96 individual interrupts that are grouped into blocks of eight. Each group is fed into one of 12 core interrupt lines ($\overline{\text{INT1}}$ to $\overline{\text{INT12}}$). Each of the 96 interrupts is supported by its own vector stored in a dedicated RAM block that you can modify. The CPU, upon servicing the interrupt, automatically fetches the appropriate interrupt vector. It takes nine CPU clock cycles to fetch the vector and save critical CPU registers. Therefore, the CPU can respond quickly to interrupt events. Prioritization of interrupts is controlled in hardware and software. Each individual interrupt can be enabled/disabled within the PIE block.

Topic **Page**

6.1 Overview of the PIE Controller

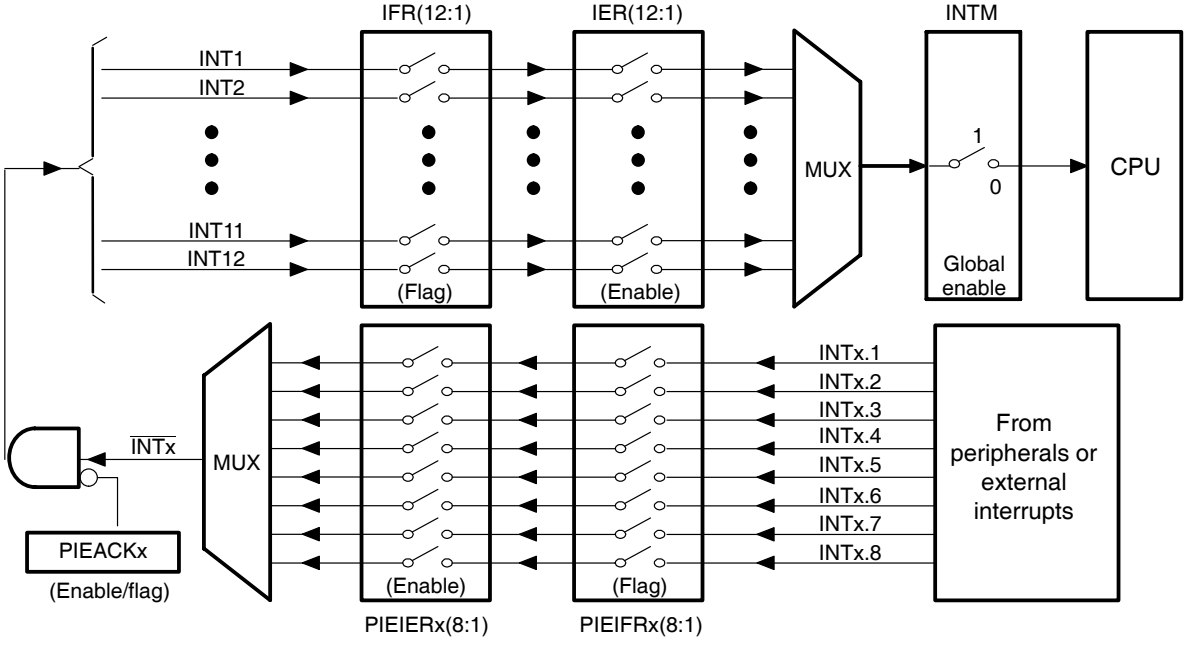
The 28x CPU supports one nonmaskable interrupt (NMI) and 16 maskable prioritized interrupt requests (INT1–INT14, RTOSINT, and DLOGINT) at the CPU level. The 28x devices have many peripherals and each peripheral is capable of generating one or more interrupts in response to many events at the peripheral level. Because the CPU does not have sufficient capacity to handle all peripheral interrupt requests at the CPU level, a centralized peripheral interrupt expansion (PIE) controller is required to arbitrate the interrupt requests from various sources such as peripherals and other external pins.

The PIE vector table is used to store the address (vector) of each interrupt service routine (ISR) within the system. There is one vector per interrupt source including all MUXed and nonMUXed interrupts. You populate the vector table during device initialization and you can update it during operation.

Interrupt Operation Sequence

Figure 6–1 shows an overview of the interrupt operation sequence for all multiplexed PIE interrupts. Interrupt sources that are not multiplexed are fed directly to the CPU.

Figure 6–1. Overview: Multiplexing of Interrupts Using the PIE Block



❑ **Peripheral Level**

An interrupt-generating event occurs in a peripheral. The interrupt flag (IF) bit corresponding to that event is set in a register for that particular peripheral.

If the corresponding interrupt enable (IE) bit is set, the peripheral generates an interrupt request to the PIE controller. If the interrupt is not enabled at the peripheral level, then the IF remains set until cleared by software. If the interrupt is enabled at a later time, and the interrupt flag is still set, the interrupt request is asserted to the PIE.

Interrupt flags within the peripheral registers must be manually cleared. See the peripheral reference guide for a specific peripheral for more information.

❑ **PIE Level**

The PIE block multiplexes eight peripheral and external pin interrupts into one CPU interrupt. These interrupts are divided into 12 groups: PIE group 1 – PIE group 12. The interrupts within a group are multiplexed into one CPU interrupt. For example, PIE group 1 is multiplexed into CPU interrupt 1 (INT1) while PIE group 12 is multiplexed into CPU interrupt 12 (INT12). Interrupt sources connected to the remaining CPU interrupts are not multiplexed. For the nonmultiplexed interrupts, the PIE passes the request directly to the CPU.

For multiplexed interrupt sources, each interrupt group in the PIE block has an associated flag bit (PIEIFRx.y) and enable bit (PIEIERx.y). In addition, there is one acknowledge bit (PIEACK) for every PIE interrupt group (INT1 to INT12) referred to as PIEACKx. Figure 6–2 illustrates the behavior of the PIE hardware under various PIEIFR and PIEIER register conditions.

Once the request is made to the PIE controller, the corresponding PIE interrupt flag (PIEIFRx.y) bit is set. If the PIE interrupt enable (PIEIERx.y) bit is also set for the given interrupt then the PIE checks the corresponding PIEACKx bit to determine if the CPU is ready for an interrupt from that group. If the PIEACKx bit is clear for that group, then the PIE sends the interrupt request to the CPU. If PIEACKx is set, then the PIE waits until it is cleared to send the request for INTx. See Section 6.3 for details.

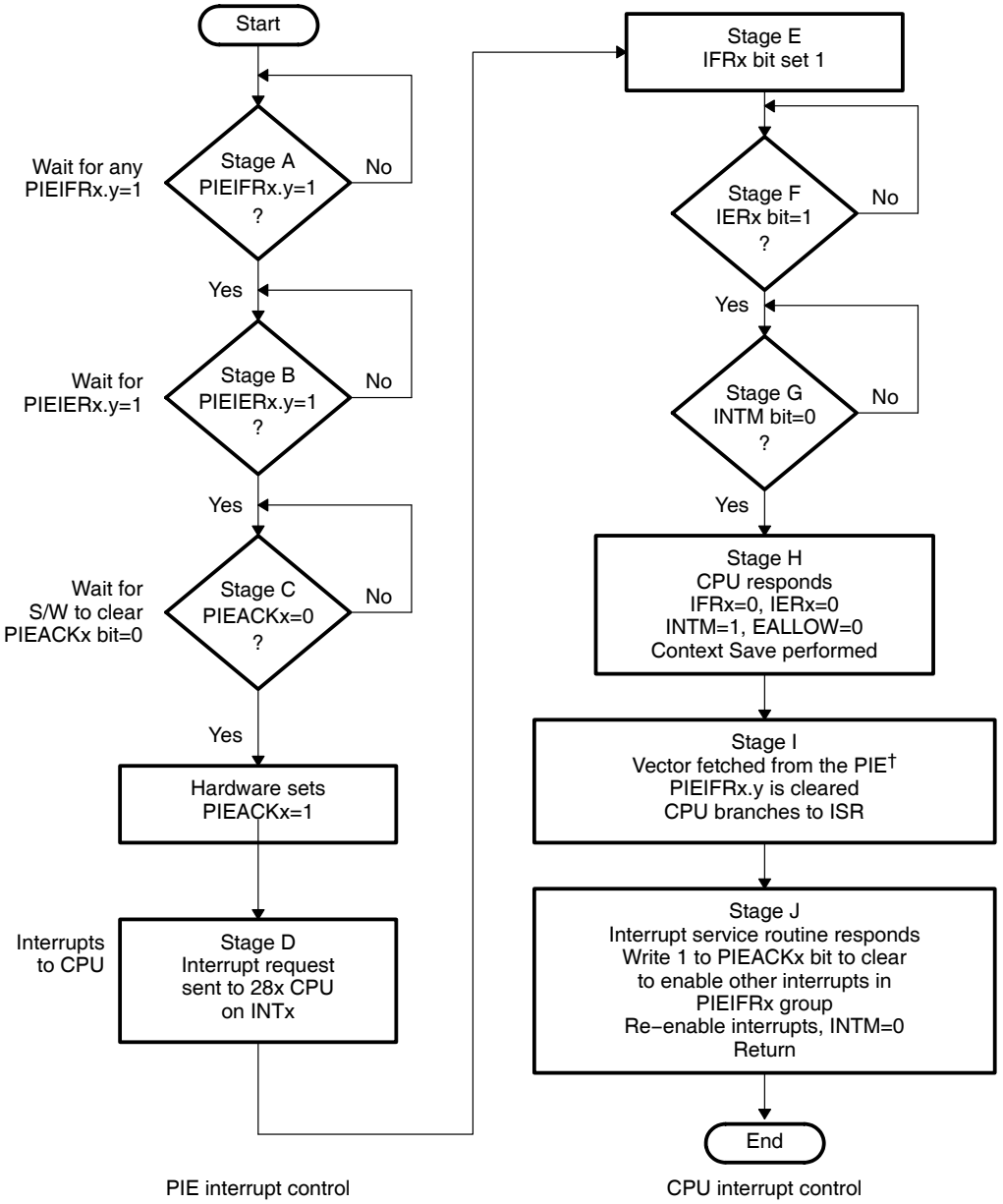
❑ **CPU Level**

Once the request is sent to the CPU, the CPU level interrupt flag (IFR) bit corresponding to INTx is set. After a flag has been latched in the IFR, the corresponding interrupt is not serviced until it is appropriately enabled in the CPU interrupt enable (IER) register or the debug interrupt enable register (DBGIER) and the global interrupt mask (INTM) bit.

As shown in Table 6–1, the requirements for enabling the maskable interrupt at the CPU level depends on the interrupt handling process being used. In the

standard process, which happens most of the time, the DBGIER register is not used. When the 28x is in real-time emulation mode and the CPU is halted, a different process is used. In this special case, the DBGIER is used and the INTM bit is ignored. If the DSP is in real-time mode and the CPU is running, the standard interrupt-handling process applies.

Figure 6–2. Typical PIE/CPU Interrupt Response – INTx.y



† For multiplexed interrupts, the PIE will respond with the highest priority interrupt that is both flagged and enabled. If there is no interrupt that is both flagged and enabled. If there is no interrupt both flagged and enabled, then the highest priority interrupt within the group (INTx.1) is used. See Section 6.3.3 for more details.

Table 6–1. Enabling Interrupt

Interrupt Handling Process	Interrupt Enabled If…
Standard	INTM = 0 and bit in IER is 1
DSP in real-time mode and halted	Bit in IER is 1 and DBGIER is 1

The CPU then prepares to service the interrupt. This preparation process is described in detail in *TMS320C28x DSP CPU and Instruction Set Reference Guide* (literature number SPRU430). In preparation, the corresponding CPU IFR and IER bits are cleared, EALLOW and LOOP are cleared, INTM and DBGM are set, the pipeline is flushed and the return address is stored, and the automatic context save is performed. The vector of the ISR is then fetched from the PIE module. If the interrupt request comes from a multiplexed interrupt, the PIE module uses the group PIEIERx and PIEIFRx registers to decode which interrupt needs to be serviced. This decode process is described in detail in Section 6.3.3.

The address for the interrupt service routine that is executed is fetched directly from the PIE interrupt vector table. There is one 32-bit vector for each of the possible 96 interrupts within the PIE. Interrupt flags within the PIE module (PIEIFRx.y) are automatically cleared when the interrupt vector is fetched. The PIE acknowledge bit for a given interrupt group, however, must be cleared manually when ready to receive more interrupts from the PIE group.

6.2 Vector Table Mapping

On 28xx devices, the interrupt vector table can be mapped to five distinct locations in memory. In practice only the PIE vector table mapping is used for F28xx devices.

This vector mapping is controlled by the following mode bits/signals:

VMAP: VMAP is found in Status Register 1 ST1 (bit 3). A device reset sets this bit to 1. The state of this bit can be modified by writing to ST1 or by SETC/CLRC VMAP instructions. For normal F2810/12 operation leave this bit set.

M0M1MAP: M0M1MAP is found in Status Register 1 ST1 (bit 11). A device reset sets this bit to 1. The state of this bit can be modified by writing to ST1 or by SETC/CLRC M0M1MAP instructions. For normal 28xx device operation, this bit should remain set. M0M1MAP = 0 is reserved for TI testing only.

MP/MC: This bit is found in XINTCNF2 Register (bit 8). On the devices with an external interface (XINTF) the default value of this bit, on reset, is set by the XMP/MC input device signal. On devices without an XINTF, the XMP/MC is tied low internally. The state of this bit can be modified after reset by writing to the XINTCNF2 register (address 0x0000 0B34).

ENPIE: ENPIE is found in PIECTRL Register (bit 0). The default value of this bit, on reset, is set to 0 (PIE disabled). The state of this bit can be modified after reset by writing to the PIECTRL register (address 0x0000 0CE0).

Using these bits and signals the possible vector table mappings are shown in Table 6–2.

Table 6–2. Interrupt Vector Table Mapping†

Vector MAPS	Vectors Fetched From	Address Range	VMAP	M0M1MAP	MP/$\overline{MC}$	ENPIE
M1 Vector‡	M1 SARAM Block	0x000000–0x00003F	0	0	X	X
M0 Vector‡	M0 SARAM Block	0x000000–0x00003F	0	1	X	X
BROM Vector	ROM Block	0x3FFFC0–0x3FFFFF	1	X	0	0
XINTF Vector§	XINTF Zone 7 Block	0x3FFFC0–0x3FFFFF	1	X	1	0
PIE Vector	PIE Block	0x000D00–0x000DFF	1	X	X	1

† On the 281x devices, the VMAP and M0M1MAP modes are set to 1 on reset. The ENPIE mode is forced to 0 on reset.
‡ Vector map MO and M1 Vector is a reserved mode only. On the28x devices these are used as RAM.
§ Valid on F2812, C2812, and R2812 devices only

The M1 and M0 vector table mapping is reserved for TI testing only. When using other vector mappings, the M0 and M1 memory blocks are treated as RAM blocks and can be used freely without any restrictions.

After a device reset operation, the vector table is mapped as shown in Table 6–3.

Table 6–3. Vector Table Mapping After Reset Operation†

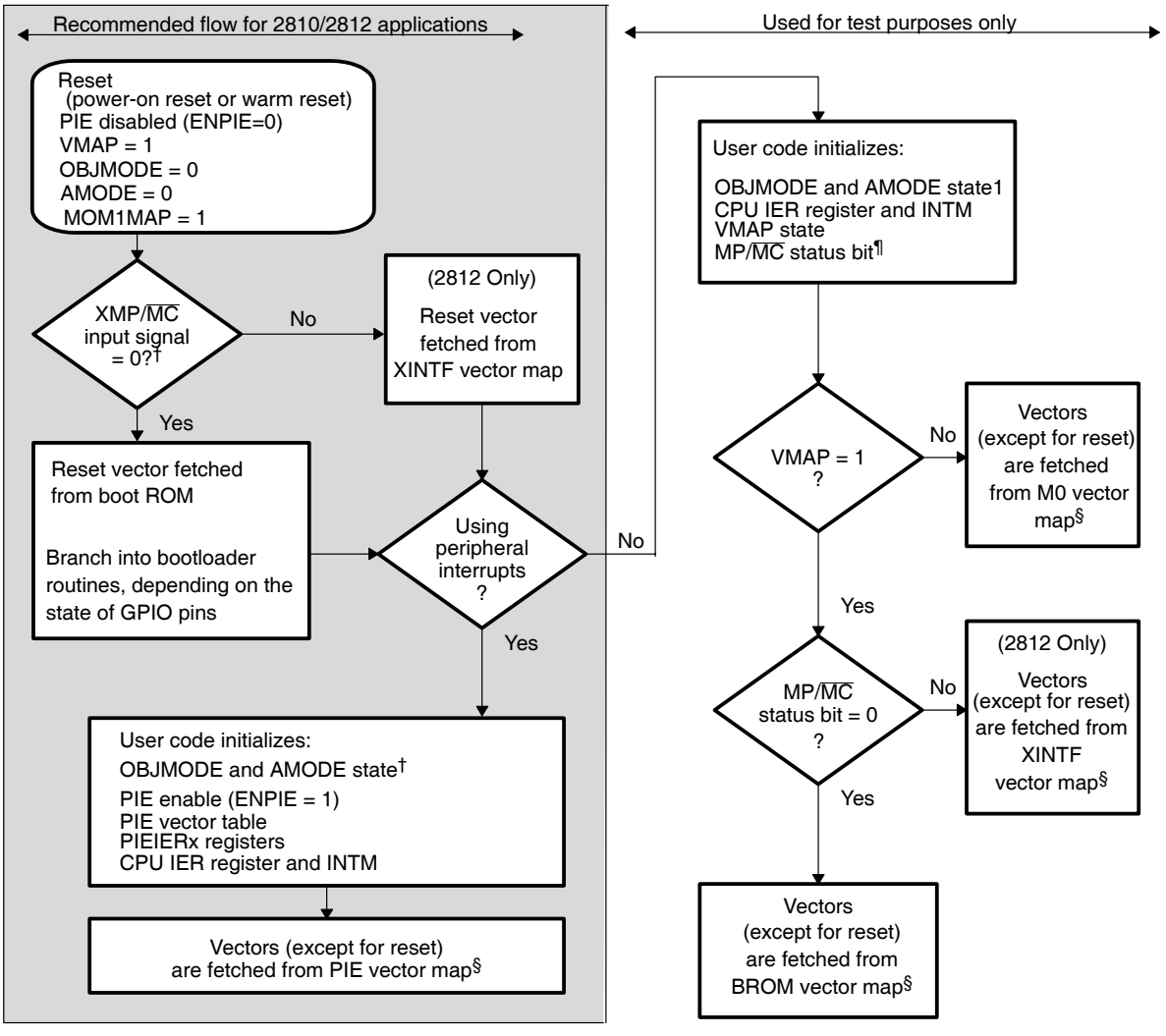
VECTOR MAPS	RESET FETCHED FROM	ADDRESS RANGE	VMAP	M0M1MAP	MP/$\overline{\text{MC}}$	ENPIE
BROM Vector	ROM Block	0x3FFFC0–0x3FFFFF	1	1	0	0
XINTF Vector§	XINTF Zone 7 Block	0x3FFFC0–0x3FFFFF	1	1	1	0

† On the 28x devices, the VMAP and M0M1MAP modes are set to 1 on reset. The ENPIE mode is forced to 0 on reset.
§ Valid on F2812, C2812, and R2812 devices only

After the reset and boot is complete, the PIE vector table should be initialized by the user's code. Then the application enables the PIE vector table. From that point on the interrupt vectors are fetched from the PIE vector table. Note: when a reset occurs, the reset vector is always fetched from the vector table as shown in Table 6–3. After a reset the PIE vector table is always disabled.

Figure 6–3 illustrates the process by which the vector table mapping is selected.

Figure 6–3. Reset Flow Diagram



† The XMP/$\overline{MC}$ input signal is tied low internally on the F2810.

‡ The compatibility operating mode of the F2810 and F2812 is determined by a combination of the OBJMODE and AMODE bits in Status Register 1 (ST1):

Operating Mode	OBJMODE	AMODE	
C28x Mode	1	0	
C2xLP Source-Compatible	1	1	
C27x Object-Compatible	0	0	(Default at reset)

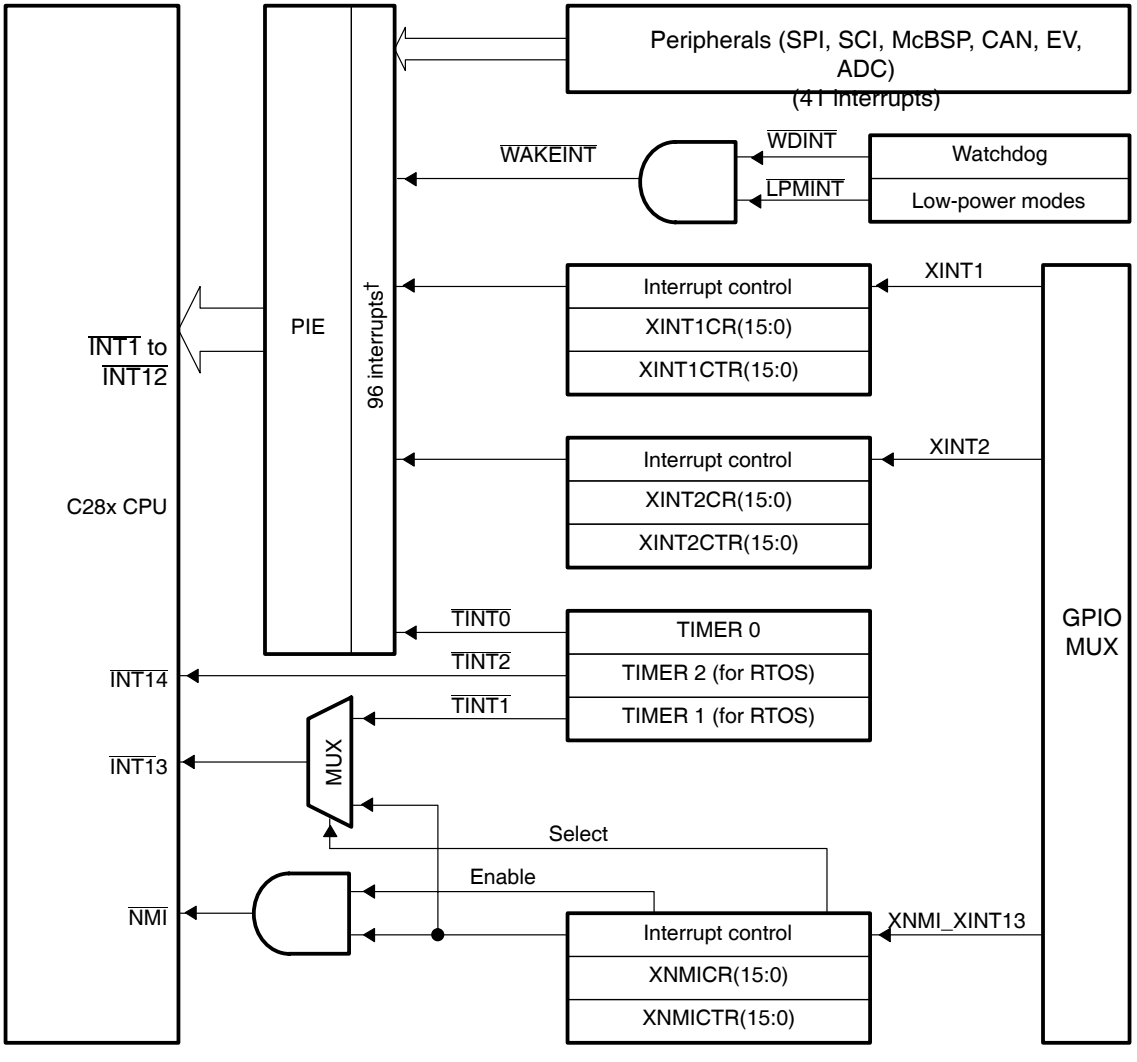
§ The reset vector is always fetched from either the BROM or XINTF vector map depending on the XMP/$\overline{MC}$ input signal.

¶ The state of the XMP/$\overline{MC}$ signal is latched into the MP/$\overline{MC}$ bit at reset, it can then be modified by software.

6.3 Interrupt Sources

Figure 6–4 shows how the various interrupt sources are multiplexed within the F2810 and F2812 devices. This MUXing scheme may not be exactly the same on all F28xx devices. See the data sheet of your particular device for details.

Figure 6–4. Interrupt Sources



† Out of a possible 96 interrupts, 45 are currently used by peripherals.

Notes: 1) In the GPIO MUX, the XINT1, XINT2 and XNMI signals are synchronized and optionally qualified by a user programmable number of clock cycles. This filters out glitches from the input source. See the GPIO MUX section for more details.

2) The WAKEINT input must be synchronized (using SYSCLKOUT of the processor) before being fed to the PIE block.

6.3.1 Procedure for Handling Multiplexed Interrupts

The PIE module multiplexes eight peripheral and external pin interrupts into one CPU interrupt. These interrupts are divided into 12 groups: PIE group 1 – PIE group 12. Each group has an associated enable PIEIER and flag PIEIFR register. These registers are used to control the flow of interrupts to the CPU. The PIE module also uses the PIEIER and PIEIFR registers to decode to which interrupt service routine the CPU should branch.

There are three main rules that should be followed when clearing bits within the PIEIFR and the PIEIER registers:

1) **Never clear a PIEIFR bit**. An incoming interrupt may be lost while the read−modify−write operation takes place. To clear a PIEIFR bit, the pending interrupt must be serviced. If you want to clear the PIEIFR bit without executing the normal service routine, then use the following procedure:

Step 1: Set the EALLOW bit to allow modification to the PIE vector table.

Step 2: Modify the PIE vector table so that the vector for the peripheral's service routine points to a temporary ISR. This temporary ISR will only perform a return from interrupt (IRET) operation.

Step 3: Enable the interrupt so that the interrupt will be serviced by the temporary ISR.

Step 4: After the temporary interrupt routine is serviced, the PIEIFR bit will be clear

Step 5: Modify the PIE vector table to re−map the peripheral's service routine to the proper service routine.

Step 6: Clear the EALLOW bit.

The CPU IFR register is integrated within the CPU. Because of this clearing a bit within the CPU IFR register can be done without concern for losing an incoming interrupt.

2) **Software-prioritizing interrupts.** Use the method found in *C281x C/C++ Header Files and Peripheral Examples in C* (literature number SPRC097).

   Use the CPU IER register as a global priority and the individual PIEIER registers for group priorities. In this case the PIEIER register is only modified within an interrupt. In addition, only the PIEIER for the same group as the interrupt being serviced is modified. This modification is done while the PIEACK bit holds additional interrupts back from the CPU.

Never disable a PIEIER bit for a group when servicing an interrupt from an unrelated group.

3) **Disabling interrupts using PIEIER**. If the PIEIER registers are used to enable and then later disable an interrupt then the procedure described in Section 6.3.2 must be followed.

6.3.2 Procedures for Enabling And Disabling Multiplexed Peripheral Interrupts

The proper procedure for enabling or disabling an interrupt is by using the peripheral interrupt enable/disable flags. The primary purpose of the PIEIER and CPU IER registers is for software prioritization of interrupts within the same interrupt group. The software *package C28x Peripheral Examples in C* (literature number SPRC097) includes an example showing this method of software prioritizing interrupts. Should bits within the PIEIER registers need to be cleared outside of this context, one of the following two procedures should be followed:

1) **Use the PIEIERx register to disable the interrupt and preserve the associated PIEIFRx flags.**

To clear bits within a PIEIERx register while preserving the associated flags in the PIEIFRx register the following procedure should be followed:

Step 1: Disable global interrupts (INTM = 1)

Step 2: Clear the PIEIERx.y bit to disable the interrupt for a given peripheral. This can be done for one or more peripherals within the same group.

Step 3: Wait 5 cycles. This delay is required to insure that any interrupt that was incoming to the CPU has been flagged within the CPU IFR register.

Step 4: Clear the CPU IFRx bit for the peripheral group. This is a safe operation on the CPU IFR register.

Step 5: Clear the PIEACKx bit for the peripheral group

Step 6: Enable global interrupts (INTM = 0)

2) **Use the PIEIERx register to disable the interrupt and clear the associated PIEIFRx flags.**

To perform a software reset of a peripheral interrupt and clear the associated flag in the PIEIFRx register and CPU IFR register, then the following procedure should be followed:

Step 1: Disable global interrupts (INTM = 1)

Step 2: Set the EALLOW bit.

Step 3: Modify the PIE vector table to temporarily map the vector of the specific peripheral interrupt to a empty interrupt service routine (ISR). This empty ISR will only perform a return from interrupt (IRET) instruction. This is the safe way to clear a single PIEIFRx.y bit without losing any interrupts from other peripherals within the group.

Step 4: Disable the peripheral interrupt at the peripheral register.

Step 5: Enable global interrupts (INTM = 0).

Step 6: Wait for any pending interrupt from the peripheral to be serviced by the empty ISR routine.

Step 7: Disable global interrupts (INTM = 1).

Step 8: Modify the PIE vector table to map the peripheral vector back to its original ISR.

Step 9: Clear the EALLOW bit.

Step 10: Disable the PIEIER bit for given peripheral.

Step 11: Clear the IFR bit for given peripheral group (this is safe operation on CPU IFR register).

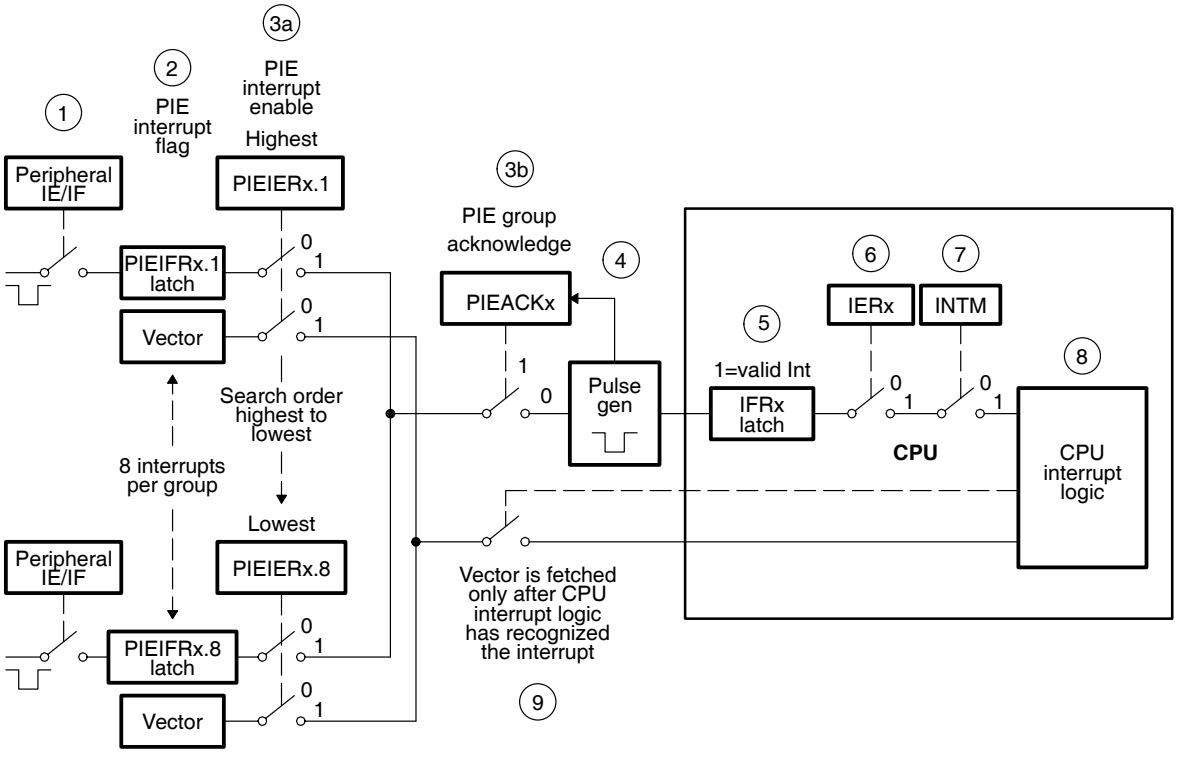
Step 12: Clear the PIEACK bit for the PIE group.

Step 13: Enable global interrupts.

6.3.3 Flow of a Multiplexed Interrupt Request From a Peripheral to the CPU

Figure 6–5 shows the flow with the steps shown in circled numbers. Following the diagram, the steps are described.

Figure 6–5. Multiplexed Interrupt Request Flow Diagram



Step 1: Any peripheral or external interrupt within the PIE group generates an interrupt. If interrupts are enabled within the peripheral module then the interrupt request is sent to the PIE module.

Step 2: The PIE module recognizes that interrupt y within PIE group x (INTx.y) has asserted an interrupt and the appropriate PIE interrupt flag bit is latched: PIEIFRx.y = 1.

Step 3: For the interrupt request to be sent from the PIE to the CPU, both of the following conditions must be true:

1) The proper enable bit must be set (PIEIERx.y = 1) and
2) The PIEACKx bit for the group must be clear.

Step 4: If both conditions in Step 3 are true, then an interrupt request is sent to the CPU and the acknowledge bit is again set (PIEACKx = 1). The

PIEACKx bit will remain set until you clear it to indicate that additional interrupts from the group can be sent from the PIE to the CPU.

Step 5: The CPU interrupt flag bit is set (CPU IFRx = 1) to indicate a pending interrupt x at the CPU level.

Step 6: If the CPU interrupt is enabled (CPU IER bit x = 1, or DBGIER bit x = 1) AND (7) the global interrupt mask is clear (INTM = 0) then the CPU will service the INTx.

Step 7: The CPU recognizes the interrupt and performs the automatic context save, clears the IER bit, sets INTM, and clears EALLOW. All of the steps that the CPU takes in order to prepare to service the interrupt are documented in the *TMS320C28x DSP CPU and Instruction Set Reference Guide* (literature number SPRU430).

Step 8: The CPU will then request the appropriate vector from the PIE.

Step 9: For multiplexed interrupts, the PIE module uses the current value in the PIEIERx and PIEIFRx registers to decode which vector address should be used.

There are two possible cases:

- The vector for the highest priority interrupt within the group that is both a) enabled in the PIEIERx register, and b) flagged as pending in the PIEIFRx is fetched and used as the branch address. In this manner if an even higher priority enabled interrupt was flagged after Step 4**,** it will be serviced first.
- If no flagged interrupts within the group are enabled, then the PIE will respond with the vector for the highest priority interrupt within that group. That is the branch address used for INTx.1. This behavior corresponds to the 28x TRAP or INT instructions.

Note:

Because the PIEIERx register is used to determine which vector will be used for the branch, you must take care when clearing bits within the PIEIERx register. The proper procedure for clearing bits within a PIEIERx register is described in Section 6.3.2. Failure to follow these steps can result in changes occurring to the PIEIERx register after an interrupt has been passed to the CPU at point (5) in Figure 6–5. In this case the PIE will respond as if a TRAP or INT instruction was executed unless there are other interrupts both pending and enabled.

At this point, the PIEIFRx.y bit is cleared and the CPU branches to the vector of the interrupt fetched from the PIE.

6.3.4 The PIE Vector Table

The PIE vector table (see Table 6–4) consists of a 256 x 16 SARAM block that can also be used as RAM (in data space only) if the PIE block is not in use. The PIE vector table contents are undefined on reset. The CPU fixes interrupt priority for INT1 to INT12. The PIE controls priority for each group of eight interrupts. For example, if INT1.1 should occur simultaneously with INT8.1, both interrupts are presented to the CPU simultaneously by the PIE block, and the CPU services INT1.1 first. If INT1.1 should occur simultaneously with INT1.8, then INT1.1 is sent to the CPU first and then INT1.8 follows. Interrupt prioritization is performed during the vector fetch portion of the interrupt processing.

A TRAP 1 to TRAP 12 instruction or an INTR INT1 to INTR INT12 instruction fetches the vector from the first location of each group (INTR1.1 to INT12.1). Similarly an OR IFR,#16-bit operation causes the vector to be fetched from INTR1.1 to INTR12.1 locations if the respective interrupt flag is set. All other TRAP, INTR, OR IFR,#16-bit operations fetch the vector from the respective table location. You should avoid using such operations for INTR1 to INTR12. The TRAP #0 operation returns a vector value of 0x000000 The vector table is EALLOW protected.

Table 6–4. 281x PIE Vector Table

Name	Vector ID	Address	Size (x16)	Description	CPU Priority	Pie Group Priority
Reset	0	0x0000 0D00	2	Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7	1 (highest)	–
INT1	1	0x0000 0D02	2	Not used. See PIE Group 1	5	–
INT2	2	0x0000 0D04	2	Not used. See PIE Group 2	6	–
INT3	3	0x0000 0D06	2	Not used. See PIE Group 3	7	–
INT4	4	0x0000 0D08	2	Not used. See PIE Group 4	8	–
INT5	5	0x0000 0D0A	2	Not used. See PIE Group 5	9	–
INT6	6	0x0000 0D0C	2	Not used. See PIE Group 6	10	–
INT7	7	0x0000 0D0E	2	Not used. See PIE Group 7	11	–
INT8	8	0x0000 0D10	2	Not used. See PIE Group 8	12	–

Notes: 1) All the locations within the PIE vector table are EALLOW protected.

2) The VECTOR ID is used by DSP/BIOS.

3) Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7.

Table 6–4. 281x PIE Vector Table (Continued)

Name	Vector ID	Address	Size (x16)	Description	CPU Priority	Pie Group Priority
INT9	9	0x0000 0D12	2	Not used. See PIE Group 9	13	–
INT10	10	0x0000 0D14	2	Not used. See PIE Group 10	14	–
INT11	11	0x0000 0D16	2	Not used. See PIE Group 11	15	–
INT12	12	0x0000 0D18	2	Not used. See PIE Group 12	16	–
INT13	13	0x0000 0D1A	2	External Interrupt 13 (XINT13) or CPU–Timer1 (for TI/RTOS use)	17	–
INT14	14	0x0000 0D1C	2	CPU–Timer2 (for TI/RTOS use)	18	–
DATALOG	15	0x0000 0D1E	2	CPU Data Logging Interrupt	19 (lowest)	–
RTOSINT	16	0x0000 0D20	2	CPU Real–Time OS Interrupt	4	–
EMUINT	17	0x0000 0D22	2	CPU Emulation Interrupt	2	–
NMI	18	0x0000 0D24	2	External Nonmaskable Interrupt	3	–
ILLEGAL	19	0x0000 0D26	2	Illegal Operation	–	–
USER1	20	0x0000 0D28	2	User–Defined Trap	–	–
USER2	21	0x0000 0D2A	2	User Defined Trap	–	–
USER3	22	0x0000 0D2C	2	User Defined Trap	–	–
USER4	23	0x0000 0D2E	2	User Defined Trap	–	–
USER5	24	0x0000 0D30	2	User Defined Trap	–	–
USER6	25	0x0000 0D32	2	User Defined Trap	–	–
USER7	26	0x0000 0D34	2	User Defined Trap	–	–
USER8	27	0x0000 0D36	2	User Defined Trap	–	–
USER9	28	0x0000 0D38	2	User Defined Trap	–	–
USER10	29	0x0000 0D3A	2	User Defined Trap	–	–

Notes: 1) All the locations within the PIE vector table are EALLOW protected.

2) The VECTOR ID is used by DSP/BIOS.

3) Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7.

Table 6–4. 281x PIE Vector Table (Continued)

Name	Vector ID	Address	Size (x16)	Description		CPU Priority	Pie Group Priority
USER11	30	0x0000 0D3C	2	User Defined Trap		–	–
USER12	31	0x0000 0D3E	2	User Defined Trap		–	–
PIE Group 1 Vectors - MUXed into CPU INT1							
INT1.1	32	0x0000 0D40	2	PDPINTA	(EV–A)	5	1 (highest)
INT1.2	33	0x0000 0D42	2	PDPINTB	(EV–B)	5	2
INT1.3	34	0x0000 0D44	2	Reserved		5	3
INT1.4	35	0x0000 0D46	2	XINT1		5	4
INT1.5	36	0x0000 0D48	2	XINT2		5	5
INT1.6	37	0x0000 0D4A	2	ADCINT	(ADC)	5	6
INT1.7	38	0x0000 0D4C	2	TINT0	(CPU–Timer0)	5	7
INT1.8	39	0x0000 0D4E	2	WAKEINT	(LPM/WD)	5	8 (lowest)
PIE Group 2 Vectors – MUXed into CPU INT2							
INT2.1	40	0x0000 0D50	2	CMP1INT	(EV–A)	6	1 (highest)
INT2.2	41	0x0000 0D52	2	CMP2INT	(EV–A)	6	2
INT2.3	42	0x0000 0D54	2	CMP3INT	(EV–A)	6	3
INT2.4	43	0x0000 0D56	2	T1PINT	(EV–A)	6	4
INT2.5	44	0x0000 0D58	2	T1CINT	(EV–A)	6	5
INT2.6	45	0x0000 0D5A	2	T1UFINT	(EV–A)	6	6
INT2.7	46	0x0000 0D5C	2	T1OFINT	(EV–A)	6	7
INT2.8	47	0x0000 0D5E	2	Reserved		6	8 (lowest)
PIE Group 3 Vectors – MUXed into CPU INT3							
INT3.1	48	0x0000 0D60	2	T2PINT	(EV–A)	7	1 (highest)
INT3.2	49	0x0000 0D62	2	T2CINT	(EV–A)	7	2

Notes: 1) All the locations within the PIE vector table are EALLOW protected.

2) The VECTOR ID is used by DSP/BIOS.

3) Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7.

Table 6–4. 281x PIE Vector Table (Continued)

Name	Vector ID	Address	Size (x16)	Description		CPU Priority	Pie Group Priority
INT3.3	50	0x0000 0D64	2	T2UFINT	(EV–A)	7	3
INT3.4	51	0x0000 0D66	2	T2OFINT	(EV–A)	7	4
INT3.5	52	0x0000 0D68	2	CAPINT1	(EV–A)	7	5
INT3.6	53	0x0000 0D6A	2	CAPINT2	(EV–A)	7	6
INT3.7	54	0x0000 0D6C	2	CAPINT3	(EV–A)	7	7
INT3.8	55	0x0000 0D6E	2	Reserved		7	8 (lowest)
PIE Group 4 Vectors – MUXed into CPU INT4							
INT4.1	56	0x0000 0D70	2	CMP4INT	(EV–B)	8	1 (highest)
INT4.2	57	0x0000 0D72	2	CMP5INT	(EV–B)	8	2
INT4.3	58	0x0000 0D74	2	CMP6INT	(EV–B)	8	3
INT4.4	59	0x0000 0D76	2	T3PINT	(EV–B)	8	4
INT4.5	60	0x0000 0D78	2	T3CINT	(EV–B)	8	5
INT4.6	61	0x0000 0D7A	2	T3UFINT	(EV–B)	8	6
INT4.7	62	0x0000 0D7C	2	T3OFINT	(EV–B)	8	7
INT4.8	63	0x0000 0D7E	2	Reserved		8	8 (lowest)
PIE Group 5 Vectors – MUXed into CPU INT5							
INT5.1	64	0x0000 0D80	2	T4PINT	(EV–B)	9	1 (highest)
INT5.2	65	0x0000 0D82	2	T4CINT	(EV–B)	9	2
INT5.3	66	0x0000 0D84	2	T4UFINT	(EV–B)	9	3
INT5.4	67	0x0000 0D86	2	T4OFINT	(EV–B)	9	4
INT5.5	68	0x0000 0D88	2	CAPINT4	(EV–B)	9	5
INT5.6	69	0x0000 0D8A	2	CAPINT5	(EV–B)	9	6
INT5.7	70	0x0000 0D8C	2	CAPINT6	(EV–B)	9	7

Notes: 1) All the locations within the PIE vector table are EALLOW protected.

2) The VECTOR ID is used by DSP/BIOS.

3) Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7.

Table 6–4. 281x PIE Vector Table (Continued)

Name	Vector ID	Address	Size (x16)	Description		CPU Priority	Pie Group Priority
INT5.8	71	0x0000 0D8E	2	Reserved		9	8 (lowest)
PIE Group 6 Vectors – MUXed into CPU INT6							
INT6.1	72	0x0000 0D90	2	SPIRXINTA	(SPI)	10	1 (highest)
INT6.2	73	0x0000 0D92	2	SPITXINTA	(SPI)	10	2
INT6.3	74	0x0000 0D94	2	Reserved		10	3
INT6.4	75	0x0000 0D96	2	Reserved		10	4
INT6.5	76	0x0000 0D98	2	MRINT	(McBSP)	10	5
INT6.6	77	0x0000 0D9A	2	MXINT	(McBSP)	10	6
INT6.7	78	0x0000 0D9C	2	Reserved		10	7
INT6.8	79	0x0000 0D9E	2	Reserved		10	8 (lowest)
PIE Group 7 Vectors – MUXed into CPU INT7							
INT7.1	80	0x0000 0DA0	2	Reserved		11	1 (highest)
INT7.2	81	0x0000 0DA2	2	Reserved		11	2
INT7.3	82	0x0000 0DA4	2	Reserved		11	3
INT7.4	83	0x0000 0DA6	2	Reserved		11	4
INT7.5	84	0x0000 0DA8	2	Reserved		11	5
INT7.6	85	0x0000 0DAA	2	Reserved		11	6
INT7.7	86	0x0000 0DAC	2	Reserved		11	7
INT7.8	87	0x0000 0DAE	2	Reserved		11	8 (lowest)
PIE Group 8 Vectors – MUXed into CPU INT8							
INT8.1	88	0x0000 0DB0	2	Reserved		12	1 (highest)
INT8.2	89	0x0000 0DB2	2	Reserved		12	2
INT8.3	90	0x0000 0DB4	2	Reserved		12	3

Notes: 1) All the locations within the PIE vector table are EALLOW protected.
2) The VECTOR ID is used by DSP/BIOS.
3) Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7.

Table 6–4. 281x PIE Vector Table (Continued)

Name	Vector ID	Address	Size (x16)	Description		CPU Priority	Pie Group Priority
INT8.4	91	0x0000 0DB6	2	Reserved		12	4
INT8.5	92	0x0000 0DB8	2	Reserved		12	5
INT8.6	93	0x0000 0DBA	2	Reserved		12	6
INT8.7	94	0x0000 0DBC	2	Reserved		12	7
INT8.8	95	0x0000 0DBE	2	Reserved		12	8 (lowest)
PIE Group 9 Vectors – MUXed into CPU INT9							
INT9.1	96	0x0000 0DC0	2	SCIRXINTA	(SCI–A)	13	1 (highest)
INT9.2	97	0x0000 0DC2	2	SCITXINTA	(SCI–A)	13	2
INT9.3	98	0x0000 0DC4	2	SCIRXINTB	(SCI–B)	13	3
INT9.4	99	0x0000 0DC6	2	SCITXINTB	(SCI–B)	13	4
INT9.5	100	0x0000 0DC8	2	ECAN0INT	(ECAN)	13	5
INT9.6	101	0x0000 0DCA	2	ECAN1INT	(ECAN)	13	6
INT9.7	102	0x0000 0DCC	2	Reserved		13	7
INT9.8	103	0x0000 0DCE	2	Reserved		13	8 (lowest)
PIE Group 10 Vectors – MUXed into CPU INT10							
INT10.1	104	0x0000 0DD0	2	Reserved		14	1 (highest)
INT10.2	105	0x0000 0DD2	2	Reserved		14	2
INT10.3	106	0x0000 0DD4	2	Reserved		14	3
INT10.4	107	0x0000 0DD6	2	Reserved		14	4
INT10.5	108	0x0000 0DD8	2	Reserved		14	5
INT10.6	109	0x0000 0DDA	2	Reserved		14	6
INT10.7	110	0x0000 0DDC	2	Reserved		14	7
INT10.8	111	0x0000 0DDE	2	Reserved		14	8 (lowest)

Notes: 1) All the locations within the PIE vector table are EALLOW protected.

2) The VECTOR ID is used by DSP/BIOS.

3) Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7.

Table 6–4. 281x PIE Vector Table (Continued)

Name	Vector ID	Address	Size (x16)	Description	CPU Priority	Pie Group Priority
PIE Group 11 Vectors – MUXed into CPU INT11						
INT11.1	112	0x0000 0DE0	2	Reserved	15	1 (highest)
INT11.2	113	0x0000 0DE2	2	Reserved	15	2
INT11.3	114	0x0000 0DE4	2	Reserved	15	3
INT11.4	115	0x0000 0DE6	2	Reserved	15	4
INT11.5	116	0x0000 0DE8	2	Reserved	15	5
INT11.6	117	0x0000 0DEA	2	Reserved	15	6
INT11.7	118	0x0000 0DEC	2	Reserved	15	7
INT11.8	119	0x0000 0DEE	2	Reserved	15	8 (lowest)
PIE Group 12 Vectors – MUXed into CPU INT12						
INT12.1	120	0x0000 0DF0	2	Reserved	16	1 (highest)
INT12.2	121	0x0000 0DF2	2	Reserved	16	2
INT12.3	122	0x0000 0DF4	2	Reserved	16	3
INT12.4	123	0x0000 0DF6	2	Reserved	16	4
INT12.5	124	0x0000 0DF8	2	Reserved	16	5
INT12.6	125	0x0000 0DFA	2	Reserved	16	6
INT12.7	126	0x0000 0DFC	2	Reserved	16	7
INT12.8	127	0x0000 0DFE	2	Reserved	16	8 (lowest)

Notes: 1) All the locations within the PIE vector table are EALLOW protected.
2) The VECTOR ID is used by DSP/BIOS.
3) Reset is always fetched from location 0x003F FFC0 in Boot ROM or XINTF Zone 7.

The interrupt grouping for peripherals and external interrupts connected to the PIE module is shown in Table 6–5. Each row in the table shows the 8 interrupts multiplexed into a particular CPU interrupt.

Table 6–5. 281x PIE Peripheral Interrupts†

CPU Interrupts	PIE Interrupts							
	INTx.8	**INTx.7**	**INTx.6**	**INTx.5**	**INTx.4**	**INTx.3**	**INTx.2**	**INTx.1**
INT1.y	$\overline{\text{WAKEINT}}$ (LPM/WD)	TINT0 (TIMER 0)	ADCINT (ADC)	XINT2	XINT1	Reserved	PDPINTB (EV-B)	PDPINTA (EV-A)
INT2.y	Reserved	T1OFINT (EV-A)	T1UFINT (EV-A)	T1CINT (EV-A)	T1PINT (EV-A)	CMP3INT (EV-A)	CMP2INT (EV-A)	CMP1INT (EV-A)
INT3.y	Reserved	CAPINT3 (EV-A)	CAPINT2 (EV-A)	CAPINT1 (EV-A)	T2OFINT (EV-A)	T2UFINT (EV-A)	T2CINT (EV-A)	T2PINT (EV-A)
INT4.y	Reserved	T3OFINT (EV-B)	T3UFINT (EV-B)	T3CINT (EV-B)	T3PINT (EV-B)	CMP6INT (EV-B)	CMP5INT (EV-B)	CMP4INT (EV-B)
INT5.y	Reserved	CAPINT6 (EV-B)	CAPINT5 (EV-B)	CAPINT4 (EV-B)	T4OFINT (EV-B)	T4UFINT (EV-B)	T4CINT (EV-B)	T4PINT (EV-B)
INT6.y	Reserved	Reserved	MXINT (McBSP)	MRINT (McBSP)	Reserved	Reserved	SPITXINTA (SPI)	SPIRXINTA (SPI)
INT7.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
INT8.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
INT9.y	Reserved	Reserved	ECAN1INT (CAN)	ECAN0INT (CAN)	SCITXINTB (SCI-B)	SCIRXINTB (SCI-B)	SCITXINTA (SCI-A)	SCIRXINTA (SCI-A)
INT10.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
INT11.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved
INT12.y	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved	Reserved

† Out of the 96 possible interrupts, 45 interrupts are currently used. the remaining interrupts are reserved for future devices. These interrupts can be used as software interrupts if they are enabled at the PIEIFRx level, provided none of the interrupts within the group is being used by a peripheral. Otherwise, interrupts coming in from peripherals may be lost by accidentally clearing their flag while modifying the PIEIFR.
To summarize, there are two safe cases when the reserved interrupts could be used as software interrupts:
1) No peripheral within the group is asserting interrupts.
2) No peripheral interrupts are assigned to the group (example PIE group 12).

6.4 PIE Configuration Registers

The registers controlling the functionality of the PIE block are shown in Table 6–6.

Table 6–6. PIE Configuration and Control Registers

Name	Address	Size (x16)	Description
PIECTRL	0x0000–0CE0	1	PIE, Control Register
PIEACK	0x0000–0CE1	1	PIE, Acknowledge Register
PIEIER1	0x0000–0CE2	1	PIE, INT1 Group Enable Register
PIEIFR1	0x0000–0CE3	1	PIE, INT1 Group Flag Register
PIEIER2	0x0000–0CE4	1	PIE, INT2 Group Enable Register
PIEIFR2	0x0000–0CE5	1	PIE, INT2 Group Flag Register
PIEIER3	0x0000–0CE6	1	PIE, INT3 Group Enable Register
PIEIFR3	0x0000–0CE7	1	PIE, INT3 Group Flag Register
PIEIER4	0x0000–0CE8	1	PIE, INT4 Group Enable Register
PIEIFR4	0x0000–0CE9	1	PIE, INT4 Group Flag Register
PIEIER5	0x0000–0CEA	1	PIE, INT5 Group Enable Register
PIEIFR5	0x0000–0CEB	1	PIE, INT5 Group Flag Register
PIEIER6	0x0000–0CEC	1	PIE, INT6 Group Enable Register
PIEIFR6	0x0000–0CED	1	PIE, INT6 Group Flag Register
PIEIER7	0x0000–0CEE	1	PIE, INT7 Group Enable Register
PIEIFR7	0x0000–0CEF	1	PIE, INT7 Group Flag Register
PIEIER8	0x0000–0CF0	1	PIE, INT8 Group Enable Register
PIEIFR8	0x0000–0CF1	1	PIE, INT8 Group Flag Register
PIEIER9	0x0000–0CF2	1	PIE, INT9 Group Enable Register
PIEIFR9	0x0000–0CF3	1	PIE, INT9 Group Flag Register
PIEIER10	0x0000–0CF4	1	PIE, INT10 Group Enable Register
PIEIFR10	0x0000–0CF5	1	PIE, INT10 Group Flag Register
PIEIER11	0x0000–0CF6	1	PIE, INT11 Group Enable Register

Note: The PIE configuration and control registers are not protected by EALLOW mode. The PIE vector table is protected.

Table 6–6. PIE Configuration and Control Registers (Continued)

Name	Address	Size (x16)	Description
PIEIFR11	0x0000–0CF7	1	PIE, INT11 Group Flag Register
PIEIER12	0x0000–0CF8	1	PIE, INT12 Group Enable Register
PIEIFR12	0x0000–0CF9	1	PIE, INT12 Group Flag Register
Reserved	0x0000–0CFA 0x0000–0CFF	6	Reserved

Note: The PIE configuration and control registers are not protected by EALLOW mode. The PIE vector table is protected.

6.5 PIE Interrupt Registers

Table 6–7. PIECTRL Register-Address CE0

15	1	0
PIEVECT		ENPIE
R-0		R/W-0

Legend: R = Read access, W = write access, -0 = value after reset

Table 6–8. PIECTRL Register-Field Descriptions

Bits	Field	Description
15–1	PIEVECT	These bits indicate the address within the PIE vector table from which the vector was fetched. The least significant bit of the address is ignored and only bits 1 to 15 of the address is shown. You can read the vector value to determine which interrupt generated the vector fetch. **Example** `If PIECTRL = 0x0d27` then the vector from address 0x0D26 (illegal operation) was fetched.
0	ENPIE	Enable vector fetching from PIE block. When ENPIE is set to 1, all vectors are fetched from the PIE vector table. If this bit is set to 0, the PIE block is disabled and vectors are fetched from the CPU vector table in boot ROM or external interface Zone 7. All PIE block registers (PIEACK, PIEIFR, PIEIER) can be accessed even when the PIE block is disabled. **Note:** The reset vector is never fetched from the PIE, even when it is enabled. This vector is always fetched from boot ROM or XINTF Zone 7 depending on the state of the XMPNMC input signal.

Figure 6–6. PIE Interrupt Acknowledge Register (PIEACK) Register-Address CE1

15	12	11	0
Reserved		PIEACKx	
R-0		R/W1C-0	

Legend: R = Read access, W1C = write1 to clear, -0 = value after reset

Table 6–9. PIE Interrupt Acknowledge Register (PIEACK) Register Field Descriptions

Bits	Field	Description
15–12	Reserved	
11–0	PIEACKx	Writing a 1 to the respective interrupt bit clears the bit and enables the PIE block to drive a pulse into the core interrupt input, if an interrupt is pending on any of the group interrupts. Reading this register indicates if an interrupt is pending in the respective group. Bit 0 refers to $\overline{\text{INT1}}$ up to Bit 11, which refers to $\overline{\text{INT12}}$. Note: Writes of 0 are ignored.

6.5.1 PIE Interrupt Flag Registers

There are twelve PIEIFR registers, one for each CPU interrupt used by the PIE module (INT1–INT12).

Figure 6–7. PIEIFRx Register (x = 1 to 12)

15							8
Reserved							
R-0							
7	**6**	**5**	**4**	**3**	**2**	**1**	**0**
INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Table 6–10. PIEIFRx Register (x = 1 to 12) Field Descriptions

Bits	Field	Description
15–8	Reserved	
7	INTx.8	These register bits indicate if an interrupt is currently active. They behave very much like the CPU interrupt flag register. When an interrupt is active, the respective register bit is set. The bit is cleared when the interrupt is serviced or by writing a 0 to the register bit. This register can also be read to determine which interrupts are active or pending. x = 1 to 12. INTx means CPU INT1 to INT12
6	INTx.7	
5	INTx.6	
4	INTx.5	
3	INTx.4	
2	INTx.3	
1	INTx.2	
0	INTx.1	

Notes: 1) All of the above registers reset values are set by reset.

2) Hardware has priority over CPU accesses to the PIEIFR registers.

3) The PIEIFR register bit is cleared during the interrupt vector fetch portion of the interrupt processing.

Note:

Never clear a PIEIFR bit. An interrupt may be lost during the read-modify-write operation. See Section 6.3.1 for a method to clear flagged interrupts.

6.5.2 PIE Interrupt Enable Registers

There are twelve PIEIER registers, one for each CPU interrupt used by the PIE module (INT1–INT12).

Figure 6–8. PIEIERx Register (x = 1 to 12)

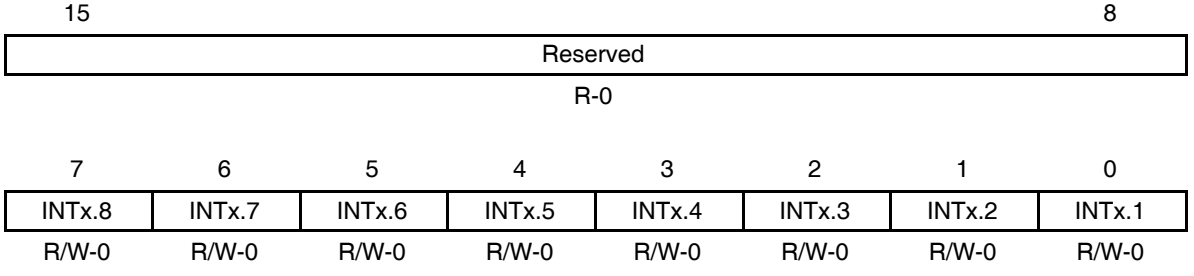


Table 6–11. PIEIERx Register (x = 1 to 12) Field Descriptions

Bits	Field	Description
15–8	Reserved	
7	INTx.8	These register bits individually enable an interrupt within a group and behave very much like the core interrupt enable register. Setting a bit to 1 enables the servicing of the respective interrupt. Setting a bit to 0 disables the servicing of the interrupt.
6	INTx.7	
5	INTx.6	x = 1 to 12. INTx means CPU INT1 to INT12
4	INTx.5	
3	INTx.4	
2	INTx.3	
1	INTx.2	
0	INTx.1	

Note: All of the above registers reset values are set by reset.

Note:

Care must be taken when clearing PIEIER bits during normal operation. See Section 6.3.2 for the proper procedure for handling these bits.

6.5.3 CPU Interrupt Flag Register (IFR)

The CPU interrupt flag register (IFR), is a 16-bit, CPU register and is used to identify and clear pending interrupts. The IFR contains flag bits for all the

maskable interrupts at the CPU level (INT1–INT14, DLOGINT and RTOSINT). When the PIE is enabled, the PIE module multiplexes interrupt sources for INT1–INT12.

When a maskable interrupt is requested, the flag bit in the corresponding peripheral control register is set to 1. If the corresponding mask bit is also 1, the interrupt request is sent to the CPU, setting the corresponding flag in the IFR. This indicates that the interrupt is pending or waiting for acknowledgement.

To identify pending interrupts, use the PUSH IFR instruction and then test the value on the stack. Use the OR IFR instruction to set IFR bits and use the AND IFR instruction to manually clear pending interrupts. All pending interrupts are cleared with the AND IFR #0 instruction or by a hardware reset.

The following events also clear an IFR flag:

- ❑ The CPU acknowledges the interrupt.
- ❑ The 28x device is reset.

Notes:

1) To clear an IFR bit, you must write a zero to it, not a one.

2) When a maskable interrupt is acknowledged, *only* the IFR bit is cleared automatically. The flag bit in the corresponding peripheral control register is *not* cleared. If an application requires that the control register flag be cleared, the bit must be cleared by software.

3) When an interrupt is requested by an INTR instruction and the corresponding IFR bit is set, the CPU does not clear the bit automatically. If an application requires that the IFR bit be cleared, the bit must be cleared by software.

4) IMR and IFR registers pertain to core-level interrupts. All peripherals have their own interrupt mask and flag bits in their respective control/configuration registers. Note that several peripheral interrupts are grouped under one core-level interrupt.

Figure 6–9. Interrupt Flag Register (IFR) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, –0 = value after reset

Table 6–12. Interrupt Flag Register (IFR) Field Descriptions

Bits	Field	Description
15	RTOSINT	Real-time operating system flag. RTOSINT is the flag for RTOS interrupts. 0 No RTOS interrupt is pending 1 At least one RTOS interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
14	DLOGINT	Data logging interrupt fag. DLOGINT is the flag for data logging interrupts. 0 No DLOGINT is pending 1 At least one DLOGINT interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
13	INT14	Interrupt 14 flag. INT14 is the flag for interrupts connected to CPU interrupt level INT14. 0 No INT14 interrupt is pending 1 At least one INT14 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
12	INT13	Interrupt 13 flag. INT13 is the flag for interrupts connected to CPU interrupt level INT13. 0 No INT13 interrupt is pending 1 At least one INT13 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
11	INT12	Interrupt 12 flag. INT12 is the flag for interrupts connected to CPU interrupt level INT12. 0 No INT12 interrupt is pending 1 At least one INT12 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request

Table 6–12. Interrupt Flag Register (IFR) Field Descriptions (Continued)

Bits	Field	Description
10	INT11	Interrupt 11 flag. INT11 is the flag for interrupts connected to CPU interrupt level INT11. 0 No INT11 interrupt is pending 1 At least one INT11 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
9	INT10	Interrupt 10 flag. INT10 is the flag for interrupts connected to CPU interrupt level INT10. 0 No INT10 interrupt is pending 1 At least one INT6 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
8	INT9	Interrupt 9 flag. INT9 is the flag for interrupts connected to CPU interrupt level INT6. 0 No INT9 interrupt is pending 1 At least one INT9 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
7	INT8	Interrupt 8 flag. INT8 is the flag for interrupts connected to CPU interrupt level INT6. 0 No INT8 interrupt is pending 1 At least one INT8 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
6	INT7	Interrupt 7 flag. INT7 is the flag for interrupts connected to CPU interrupt level INT7. 0 No INT7 interrupt is pending 1 At least one INT7 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
5	INT6	Interrupt 6 flag. INT6 is the flag for interrupts connected to CPU interrupt level INT6. 0 No INT6 interrupt is pending 1 At least one INT6 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
4	INT5	Interrupt 5 flag. INT5 is the flag for interrupts connected to CPU interrupt level INT5. 0 No INT5 interrupt is pending 1 At least one INT5 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request

Table 6–12. Interrupt Flag Register (IFR) Field Descriptions (Continued)

Bits	Field	Description
3	INT4	Interrupt 4 flag. INT4 is the flag for interrupts connected to CPU interrupt level INT4. 0 No INT4 interrupt is pending 1 At least one INT4 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
2	INT3	Interrupt 3 flag. INT3 is the flag for interrupts connected to CPU interrupt level INT3. 0 No INT3 interrupt is pending 1 At least one INT3 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
1	INT2	Interrupt 2 flag. INT2 is the flag for interrupts connected to CPU interrupt level INT2. 0 No INT2 interrupt is pending 1 At least one INT2 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request
0	INT1	Interrupt 1 flag. INT1 is the flag for interrupts connected to CPU interrupt level INT1. 0 No INT1 interrupt is pending 1 At least one INT1 interrupt is pending. Write a 0 to this bit to clear it to 0 and clear the interrupt request

6.5.4 Interrupt Enable Register (IER) and Debug Interrupt Enable Register (DBGIER)

The IER is a 16-bit CPU register. The IER contains enable bits for all the maskable CPU interrupt levels (INT1–INT14, RTOSINT and DLOGINT). Neither NMI nor XRS is included in the IER; thus, IER has no effect on these interrupts.

You can read the IER to identify enabled or disabled interrupt levels, and you can write to the IER to enable or disable interrupt levels. To enable an interrupt level, set its corresponding IER bit to one using the OR IER instruction. To disable an interrupt level, set its corresponding IER bit to zero using the AND IER instruction. When an interrupt is disabled, it is not acknowledged, regardless of the value of the INTM bit. When an interrupt is enabled, it is acknowledged if the corresponding IFR bit is one and the INTM bit is zero.

When using the OR IER and AND IER instructions to modify IER bits make sure they do not modify the state of bit 15 (RTOSINT) unless a real-time operating system is present.

When a hardware interrupt is serviced or an INTR instruction is executed, the corresponding IER bit is cleared automatically. When an interrupt is requested by the TRAP instruction the IER bit is not cleared automatically. In the case of the TRAP instruction if the bit needs to be cleared it must be done by the interrupt service routine.

At reset, all the IER bits are cleared to 0, disabling all maskable CPU level interrupts.

The IER register is shown in Figure 6–10, and descriptions of the bits follow the figure.

Figure 6–10. Interrupt Enable Register (IER) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0
7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0	R/W–0

Note: R = Read access, W = Write access, –0 = value after reset

Table 6–13. Interrupt Enable Register (IER) Field Descriptions

Bits	Field	Description
15	RTOSINT	Real-time operating system interrupt enable. RTOSINT enables or disables the CPU RTOS interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
14	DLOGINT	Data logging interrupt enable. DLOGINT enables or disables the CPU data logging interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
13	INT14	Interrupt 14 enable. INT14 enables or disables CPU interrupt level INT14. 0 Level INT14 is disabled 1 Level INT14 is enabled
12	INT13	Interrupt 13 enable. INT13 enables or disables CPU interrupt level INT13. 0 Level INT13 is disabled

Table 6–13. Interrupt Enable Register (IER) Field Descriptions (Continued)

Bits	Field	Description
		1 Level INT13 is enabled
11	INT12	Interrupt 12 enable. INT12 enables or disables CPU interrupt level INT12.
		0 Level INT12 is disabled
		1 Level INT12 is enabled
10	INT11	Interrupt 11 enable. INT11 enables or disables CPU interrupt level INT11.
		0 Level INT11 is disabled
		1 Level INT11 is enabled
9	INT10	Interrupt 10 enable. INT10 enables or disables CPU interrupt level INT10.
		0 Level INT10 is disabled
		1 Level INT10 is enabled
8	INT9	Interrupt 9 enable. INT9 enables or disables CPU interrupt level INT9.
		0 Level INT9 is disabled
		1 Level INT9 is enabled
7	INT8	Interrupt 8 enable. INT8 enables or disables CPU interrupt level INT8.
		0 Level INT8 is disabled
		1 Level INT8 is enabled
6	INT7	Interrupt 7 enable. INT7 enables or disables CPU interrupt level INT7.
		0 Level INT7 is disabled
		1 Level INT7 is enabled
5	INT6	Interrupt 6 enable. INT6 enables or disables CPU interrupt level INT6.
		0 Level INT6 is disabled
		1 Level INT6 is enabled
4	INT5	Interrupt 5 enable. INT5 enables or disables CPU interrupt level INT5.
		0 Level INT5 is disabled
		1 Level INT5 is enabled
3	INT4	Interrupt 4 enable. INT4 enables or disables CPU interrupt level INT4.

Table 6–13. Interrupt Enable Register (IER) Field Descriptions (Continued)

Bits	Field	Description
		0 Level INT4 is disabled
		1 Level INT4 is enabled
2	INT3	Interrupt 3 enable. INT3 enables or disables CPU interrupt level INT3.
		0 Level INT3 is disabled
		1 Level INT3 is enabled
1	INT2	Interrupt 2 enable. INT2 enables or disables CPU interrupt level INT2.
		0 Level INT2 is disabled
		1 Level INT2 is enabled
0	INT1	Interrupt 1 enable. INT1 enables or disables CPU interrupt level INT1.
		0 Level INT1 is disabled
		1 Level INT1 is enabled

The Debug Interrupt Enable Register (DBGIER) is used only when the CPU is halted in real-time emulation mode. An interrupt enabled in the DBGIER is defined as a time-critical interrupt. When the CPU is halted in real-time mode, the only interrupts that are serviced are time-critical interrupts that are also enabled in the IER. If the CPU is running in real-time emulation mode, the standard interrupt-handling process is used and the DEBIER is ignored.

As with the IER, you can read the DBGIER to identify enabled or disabled interrupts and write to the DBGIER to enable or disable interrupts. To enable an interrupt, set its corresponding bit to 1. To disable an interrupt, set its corresponding bit to 0. Use the PUSH DBGIER instruction to read from the DBGIER and POP DBGIER to write to the DEBIER register. At reset, all the DBGIER bits are set to 0.

Figure 6–11. Debug Interrupt Enable Register (DBGIER) — CPU Register

15	14	13	12	11	10	9	8
RTOSINT	DLOGINT	INT14	INT13	INT12	INT11	INT10	INT9
R/W-0							

7	6	5	4	3	2	1	0
INT8	INT7	INT6	INT5	INT4	INT3	INT2	INT1
R/W-0							

Note: R = Read access, W = Write access, –0 = value after reset

Table 6–14. Debug Interrupt Enable Register (DBGIER) Field Descriptions

Bits	Field	Description
15	RTOSINT	Real-time operating system interrupt enable. RTOSINT enables or disables the CPU RTOS interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
14	DLOGINT	Data logging interrupt enable. DLOGINT enables or disables the CPU data logging interrupt. 0 Level INT6 is disabled 1 Level INT6 is enabled
13	INT14	Interrupt 14 enable. INT14 enables or disables CPU interrupt level INT14. 0 Level INT14 is disabled 1 Level INT14 is enabled
12	INT13	Interrupt 13 enable. INT13 enables or disables CPU interrupt level INT13. 0 Level INT13 is disabled 1 Level INT13 is enabled
11	INT12	Interrupt 12 enable. INT12 enables or disables CPU interrupt level INT12. 0 Level INT12 is disabled 1 Level INT12 is enabled
10	INT11	Interrupt 11 enable. INT11 enables or disables CPU interrupt level INT11. 0 Level INT11 is disabled 1 Level INT11 is enabled

Table 6–14. Debug Interrupt Enable Register (DBGIER) Field Descriptions (Continued)

Bits	Field	Description
9	INT10	Interrupt 10 enable. INT10 enables or disables CPU interrupt level INT10. 0 Level INT10 is disabled 1 Level INT10 is enabled
8	INT9	Interrupt 9 enable. INT9 enables or disables CPU interrupt level INT9. 0 Level INT9 is disabled 1 Level INT9 is enabled
7	INT8	Interrupt 8 enable. INT8 enables or disables CPU interrupt level INT8. 0 Level INT8 is disabled 1 Level INT8 is enabled
6	INT7	Interrupt 7 enable. INT7 enables or disables CPU interrupt level INT77. 0 Level INT7 is disabled 1 Level INT7 is enabled
5	INT6	Interrupt 6 enable. INT6 enables or disables CPU interrupt level INT6. 0 Level INT6 is disabled 1 Level INT6 is enabled
4	INT5	Interrupt 5 enable. INT5 enables or disables CPU interrupt level INT5. 0 Level INT5 is disabled 1 Level INT5 is enabled
3	INT4	Interrupt 4 enable. INT4 enables or disables CPU interrupt level INT4. 0 Level INT4 is disabled 1 Level INT4 is enabled
2	INT3	Interrupt 3 enable. INT3 enables or disables CPU interrupt level INT3. 0 Level INT3 is disabled 1 Level INT3 is enabled

Table 6–14. Debug Interrupt Enable Register (DBGIER) Field Descriptions (Continued)

Bits	Field	Description
1	INT2	Interrupt 2 enable. INT2 enables or disables CPU interrupt level INT2. 0 Level INT2 is disabled 1 Level INT2 is enabled
0	INT1	Interrupt 1 enable. INT1 enables or disables CPU interrupt level INT1. 0 Level INT1 is disabled 1 Level INT1 is enabled

6.6 External Interrupt Control Registers

Some devices support three masked external interrupts XINT1, XINT2, XINT13. XINT13 is multiplexed with one non-maskable interrupt XNMI. Each of these external interrupts can be selected for negative or positive edge triggered and can also be enabled or disabled (including XNMI). The masked interrupts also contain a 16-bit free running up counter that is reset to zero when a valid interrupt edge is detected. This counter can be used to accurately time stamp the interrupt.

Figure 6–12. External Interrupt 1 Control Register (XINT1CR) — Address 7070h

15	8
Reserved	
R-0	

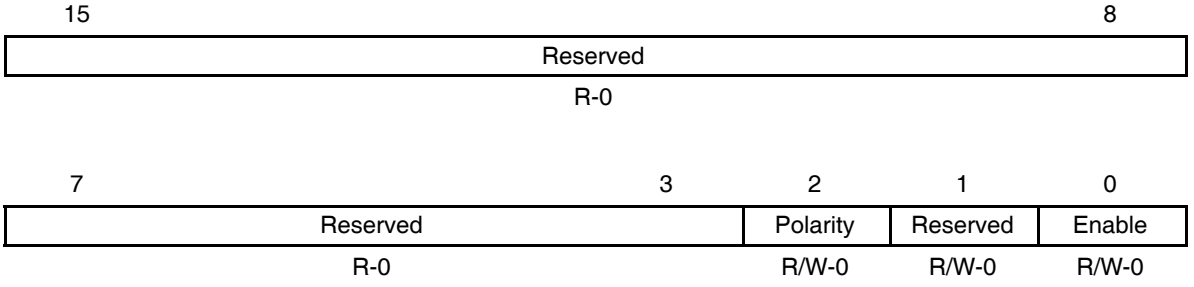
7	3	2	1	0
Reserved		Polarity	Reserved	Enable
R-0		R/W-0	R/W-0	R/W-0

Note: R = Read access, W = Write access, –0 = value after reset

Table 6–15. External Interrupt 1 Control Register (XINT1CR) Field Descriptions

Bits	Field	Description
15–3	Reserved	Reads return zero; writes have no effect.
2	Polarity	This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of a signal on the pin. 0 Interrupt generated on a falling edge (high-to-low transition) 1 Interrupt generated on a rising edge low-to-high transition)
1	Reserved	Reads return zero; writes have no effect
0	Enable	This read/write bit enables or disables external interrupt XINT1. 0 Disable interrupt 1 Enable interrupt

Figure 6–13. External Interrupt 2 Control Register (XINT2CR) — Address 7071h

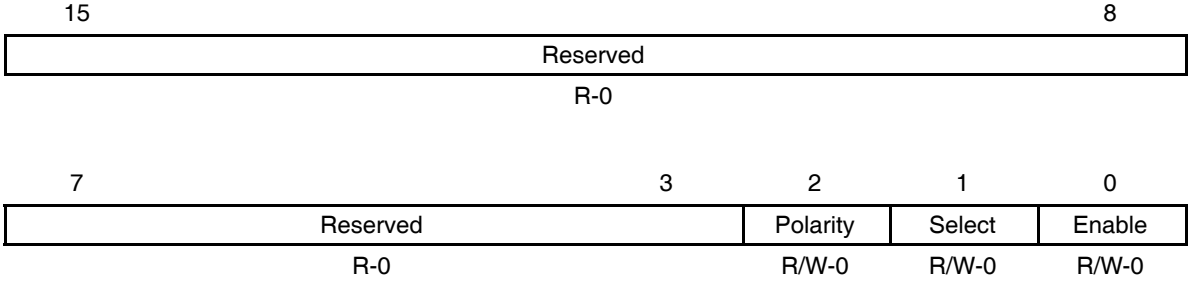


Note: R = Read access, W = Write access, –0 = value after reset

Table 6–16. External Interrupt 2 Control Register (XINT2CR) Field Descriptions

Bits	Field	Description
15–3	Reserved	Reads return zero; writes have no effect.
2	Polarity	This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of a signal on the pin. Interrupt generated on a falling edge (high-to-low transition) Interrupt generated on a rising edge low-to-high transition)
1	Reserved	Reads return zero; writes have no effect
0	Enable	This read/write bit enables or disables external interrupt XINT2. Disable interrupt Enable interrupt

Figure 6–14. External NMI Interrupt Control Register (XNMICR) — Address 7077h



Note: R = Read access, W = Write access, –0 = value after reset

Table 6–17. External NMI Interrupt Control Register (XNMICR) Field Descriptions

Bits	Field	Description
15–3	Reserved	Reads return zero; writes have no effect.
2	Polarity	This read/write bit determines whether interrupts are generated on the rising edge or the falling edge of the signal on the pin.
		0 Interrupt generated on a falling edge (high-to-low transition)
		1 Interrupt generated on a rising edge low-to-high transition)
1	Select	Select the source for INT13
		0 Timer 1 connected To INT13
		1 XNMI_XINT13 connected To INT13
0	Enable	This read/write bit enables or disables external interrupt NMI
		0 Disable XNMI interrupt
		1 Enable XNMI interrupt

The XNMI Control Register (XNMICR) can be used to enable or disable the NMI interrupt to the CPU. In additions, you can select the source for the INT13 CPU interrupt. As shown in Figure 6–4, the source of the INT13 interrupt can either the internal CPU Timer1 or the external XNMI_XINT13 signal.

On the F281x devices, CPU Timer1 is reserved for use by TI software. The INT13 interrupt can, however, still be connected to XNMI_XINT13 for customer use.

Table 6–18 shows the relationship between the XNMICR Register settings and the interrupt sources to the 28x CPU.

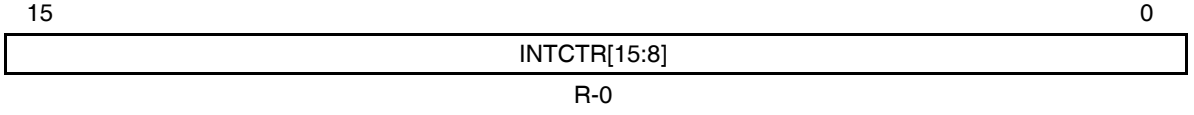
Table 6–18. XNMICR Register Settings and Interrupt Sources

XNMICR	Register Bits	28x CPU Interrupt		Timestamp
ENABLE	SELECT	NMI	INT13	(XNMICTR)
0	0	Disabled	CPU Timer 1	None
0	1	Disabled	XNMI_XINT13	None
1	0	XNMI_XINT13	CPU Timer 1	XNMI_XINT13
1	1	XNMI_XINT13	XNMI_XINT13	XNMI_XINT13

When ENABLE = 1 and SELECT = 1, both the NMI and INT13 CPU interrupts will respond to the XNMI_XINT13 signal.

For each external interrupt, there is also a 16-bit counter that is reset to 0x000 whenever an interrupt edge is detected. These counters can be used to accurately time stamp an occurrence of the interrupt.

Figure 6–15. External Interrupt 1 Counter (XINT1CTR) — Address 7078h

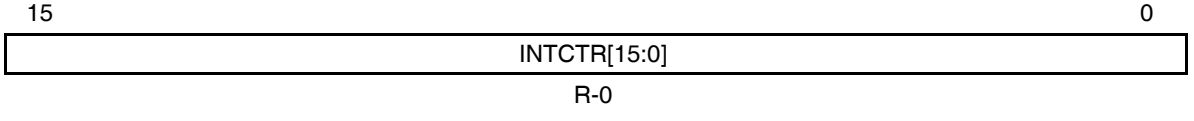


Note: R = Read access, –0 = value after reset

Table 6–19. External Interrupt 1 Counter (XINT1CTR) Field Descriptions

Bits	Field	Description
15–0	INTCTR	This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. When the interrupt is disabled, the counter stops. The counter is a free-running counter and wraps around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.

Figure 6–16. External Interrupt 2 Counter (XINT2CTR) — Address 7079h

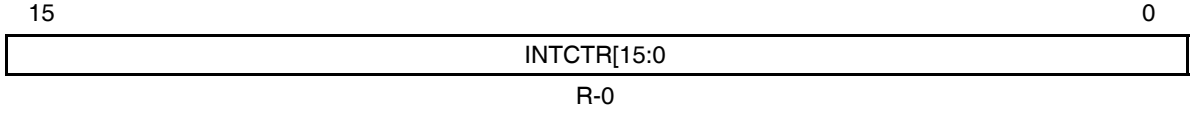


Note: R = Read access, –0 = value after reset

Table 6–20. External Interrupt 2 Counter (XINT2CTR) Field Descriptions

Bits	Field	Description
15–0	INTCTR	This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. When the interrupt is disabled, the counter stops. The counter is a free-running counter and wraps around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.

Figure 6–17. External NMI Interrupt Counter (XNMICTR) — Address 707Fh



Note: R = Read access, –0 = value after reset

Table 6–21. External NMI Interrupt Counter (XNMICTR) Field Descriptions

Bits	Field	Description
15–0	INTCTR	This is a free running 16-bit up-counter that is clocked at the SYSCLKOUT rate. The counter value is reset to 0x0000 when a valid interrupt edge is detected and then continues counting until the next valid interrupt edge is detected. When the interrupt is disabled, the counter stops. The counter is a free-running counter and wraps around to zero when the max value is reached. The counter is a read only register and can only be reset to zero by a valid interrupt edge or by reset.

This page intentionally left blank.

Appendix A

Revision History

This document was revised to SPRU078C from SPRU078B, which was released in October, 2004. The scope of the revisions was limited to technical changes as described in A.1. This appendix lists only revisions made in the most recent version.

A.1 Changes Made in This Revision

The following technical changes were made in this revision:

Page	Additions/Modifications/Deletions
2-4	Modified Table 2–2
2-4	Added a bullet on flash configuration registers to the list in Section 2.2
2-6	Modified Table 2–4, making passwords singular instead of plural
3-17	Modified description of WD check bits 5–3 in Table 3–13
5-10	Modified Table 5–12
5-11	Added part ID register as Figure 5–2
5-11	Added Table 5–14 (field descriptions for Part ID register)
6-11	Corrected reference to SPRC097 in Section 6.3.1, item 2
6-23	Modified note on Table 6–5 to clarify use of reserved interrupts

This page intentionally left blank.

Index

G

H

I

J

K

L

M

O

P

R

S

T

U

V

W

X

TMS320x281x DSP Boot ROM Reference Guide

Literature Number: SPRU095B
May 2003 − November 2004

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third−party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products & application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Contents

Figures

Tables

This page intentionally left blank.

Boot ROM

This reference guide is applicable for the Boot ROM found on the TMS320x281x generation of processors in the TMS320C2000 platform. This includes all Flash-based, ROM-based, and RAM-based devices within the 281x generation.

The Boot ROM is factory-programmed with boot-loading software. Boot-mode signals (general purpose I/Os) are used to tell the bootloader software what mode to use on power up. The 281x Boot ROM also contains standard math tables, such as SIN/COS waveforms for use in IQ math related algorithms.

This guide describes the purpose and features of the bootloader. It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

1 Boot ROM Overview

The boot ROM on the F281x, C281x, and R281x devices is a 4K x 16 block located in memory from 0x3F F000 − 0x3F FFC0. This memory block is mapped only when the MPNMC status bit in the XINTCNF2 register is 0. The same Boot ROM is included in Flash, ROM, and RAM 281x devices.

1.1 Effect of XMPNMC on the Boot ROM

In this document XMPNMC refers to the input signal to the device while MPNMC refers to the status bit in the external interface (XINTF) configuration register XINTFCNF2. On devices without an XINTF, the XMPNMC input signal is tied low internal to the device.

The MPNMC status bit in the XINTFCNF2 register switches the device between microprocessor and microcomputer mode. When high, XINTF Zone 7 is enabled on the external interface and the internal boot ROM is disabled. When low, XINTF Zone 7 is disabled from the external interface, and the on-chip boot ROM memory can be accessed instead. The XMPNMC input signal is latched into the XINTF configuration register XINTCNF2 on a reset. After reset, the state of the XMPNMC input signal is ignored and you can then modify the state of this mode in software.

On devices with an XINTF, like the F2812, the XMPNMC input signal is available externally and therefore, you can control this signal to boot from the internal boot ROM or from XINTF Zone 7. On devices without the XINTF, such as the F2810, XMPNMC is tied low internal to the device and, therefore, boot ROM is automatically enabled at reset.

The remainder of this document assumes that the XMPNMC input signal is pulled low at reset to boot from internal ROM unless stated otherwise.

1.2 On-Chip ROM Description

On the 281x devices, the 4K x 16 on-chip ROM is factory programmed with the boot-load routine and additional features. Appendix A contains the code for each of the following items:

- Bootloader functions
- Version number, release date and checksum
- Reset vector
- CPU vector table (Used for test purposes only)
- IQmath Tables

  3K x 16 of boot ROM memory is reserved for math tables and future upgrades. These math tables and in the future math functions are intended to help with performance and save RAM space.

The 281x boot ROM includes math tables that are used by the Texas Instruments™ TMS320C28x™ IQmath Library. The 28x IQmath Library is a collection of highly optimized and high precision mathematical functions for C/C++ programmers to seamlessly port a floating-point algorithm into fixed-point code on TMS320C28x devices.

These routines are typically used in computational-intensive real-time applications where optimal execution speed and high accuracy is critical. By using these routines you can achieve execution speeds that are considerably faster than equivalent code written in standard ANSI C language. In addition, by providing ready-to-use high precision functions, the TI IQmath Library can shorten significantly your DSP application development time. The *28x IQmath Library* (literature number SPRC087), can be downloaded from the TI website.

The following math tables are included in the 281x Boot ROM.

- Sin/Cos Table:

  Table size: 1282 words

  Q format: Q30

  Contents: 32-bit samples for one and a quarter period sin wave

  This is useful for accurate sin wave generation and 32-bit FFTs. This can also be used for 16-bit math, just skip over every second value.

- Normalized Inverse Table:

  Table size: 528 words

  Q format: Q29

  Contents: 32-bit normalized inverse samples plus saturation limits

  This table is used as an initial estimate in the Newton-Raphson inverse algorithm. By using a more accurate estimate the convergence is quicker and hence cycle time is faster.

- Normalized Square Root Table:

  Table size: 274 words

  Q format: Q30

  Contents: 32-bit normalized inverse square root samples plus saturation

  This table is used as an initial estimate in the Newton-Raphson square-root algorithm. By using a more accurate estimate the convergence is quicker and hence cycle time is faster.

- Normalized Arctan Table:

  Table size: 452 words

  Q format: Q30

  Contents 32-bit 2nd order coefficients for line of bset fit plus normalization table

  This table is used as an initial estimate in the Arctan iterative algorithm. By using a more accurate estimate the convergence is quicker and hence cycle time is faster.

- Rounding and Saturation Table:

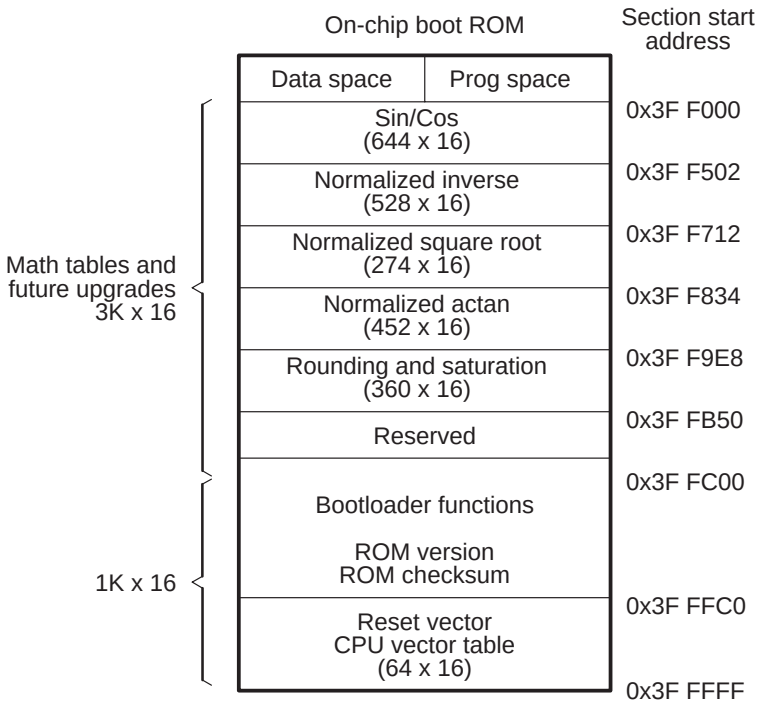
  Table size: 360 words

  Q format: Q30

  Contents 32-bit rounding and saturation limits for various Q values

Figure 1 shows the memory map of the on-chip ROM for the F2810/12. The memory block is 4Kx16 in size and is located at 0x3F F000 − 0x3F FFFF in both program and data space when the MPNMC status bit is low.

Figure 1. Memory Map of On-Chip ROM



2 Boot ROM Version and Checksum Information

The boot ROM contains its own version number located at address 0x3F FFBA. This version number starts at 1 and will be incremented any time the boot ROM code is modified. The next address, 0x3F FFBB contains the month and year (MM/YY in decimal) that the boot code was released. The next four memory locations contain a checksum value for the boot ROM. Taking a 64-bit summation of all addresses within the ROM, except for the checksum locations, generates this checksum.

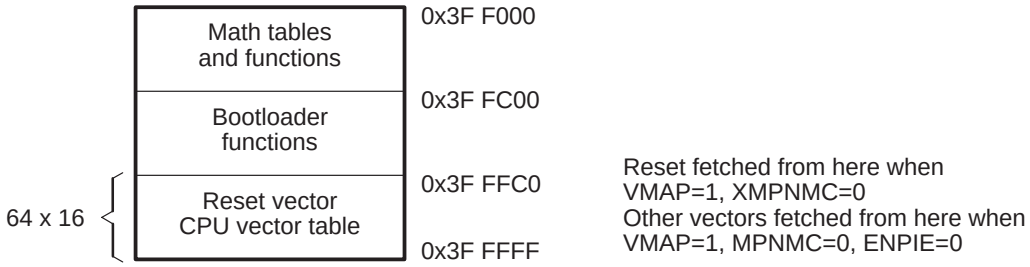
Table 1. Memory Addresses

Address	Contents
0x3F FFBA	Boot ROM Version Number
0x3F FFBB	MM/YY of release (in decimal)
0x3F FFBC	Least significant word of checksum
0x3F FFBD	...
0x3F FFBE	...
0x3F FFBF	Most significant word of checksum

3 CPU Vector Table

A CPU vector table resides in boot ROM memory from address 0x3F FFC0 − 0x3F FFFF. This vector table is active when VMAP = 1, ENPIE = 0 (PIE vector table disabled) and MPNMC = 0 (internal Boot ROM memory enabled, XINTF Zone 7 disabled).

Figure 2. Vector Table Map



Notes: 1) The VMAP bit is located in Status Register 1 (ST1). On the 281x devices, VMAP is always 1 on reset. It can be changed after reset by software, however the normal operating mode will be to leave VMAP = 1.

2) On the 2812, XMPNMC is available externally; on the 2810 and 2811, it is tied low internally. On reset, the state of the XMPNMC input signal is latched into the MPNMC status bit in the XINTCNF2 register where it can be changed by software.

3) The ENPIE bit is located in the PIECTRL register. The default state of this bit at reset is 0, which disables the Peripheral Interrupt Expansion block (PIE).

The only vector that will normally be handled from the internal boot ROM memory is the reset vector located at 0x3F FFC0. The reset vector is factory programmed to point to the InitBoot function. This function starts the boot load process. A series of checking operations is performed on General Purpose I/O (GPIO I/O) pins to determine which boot mode to use. This boot mode selection is described in the next section of this document.

The remaining vectors in the ROM are not used during normal operation on the 281x devices. After the boot process is complete, you should initialize the Peripheral Interrupt Expansion (PIE) vector table and enable the PIE block. From that point on, all vectors, except reset, will be fetched from the PIE module and not the CPU vector table shown here.

For TI silicon debug and test purposes the vectors located in the boot ROM memory point to locations in the M0 SARAM block as described in the following table. During silicon debug, you can program the specified locations in M0 with branch instructions to catch any vectors fetched from boot ROM. This is not required for normal device operation.

Table 2. Vector Locations

Vector	Location in Boot ROM	Contents (i.e., points to)	Vector	Location in Boot ROM	Contents (ie points to)
RESET	0x3F FFC0	InitBoot (0x3F FC00)	RTOSINT	0x3F FFE0	0x00 0060
INT1	0x3F FFC2	0x00 0042	Reserved	0x3F FFE2	0x00 0062
INT2	0x3F FFC4	0x00 0044	NMI	0x3F FFE4	0x00 0064
INT3	0x3F FFC6	0x00 0046	ILLEGAL	0x3F FFE6	0x00 0066
INT4	0x3F FFC8	0x00 0048	USER1	0x3F FFE8	0x00 0068
INT5	0x3F FFCA	0x00 004A	USER2	0x3F FFEA	0x00 006A
INT6	0x3F FFCC	0x00 004C	USER3	0x3F FFEC	0x00 006C
INT7	0x3F FFCE	0x00 004E	USER4	0x3F FFEE	0x00 006E
INT8	0x3F FFD0	0x00 0050	USER5	0x3F FFF0	0x00 0070
INT9	0x3F FFD2	0x00 0052	USER6	0x3F FFF2	0x00 0072
INT10	0x3F FFD4	0x00 0054	USER7	0x3F FFF4	0x00 0074
INT11	0x3F FFD6	0x00 0056	USER8	0x3F FFF6	0x00 0076
INT12	0x3F FFD8	0x00 0058	USER9	0x3F FFF8	0x00 0078
INT13	0x3F FFDA	0x00 005A	USER10	0x3F FFFA	0x00 007A
INT14	0x3F FFDC	0x00 005C	USER11	0x3F FFFC	0x00 007C
DLOGINT	0x3F FFDE	0x00 005E	USER12	0x3F FFFE	0x00 007E

4 Bootloader Features

This section describes in detail the boot mode selection process, as well as the specifics of boot loader operation.

4.1 Bootloader Functional Operation

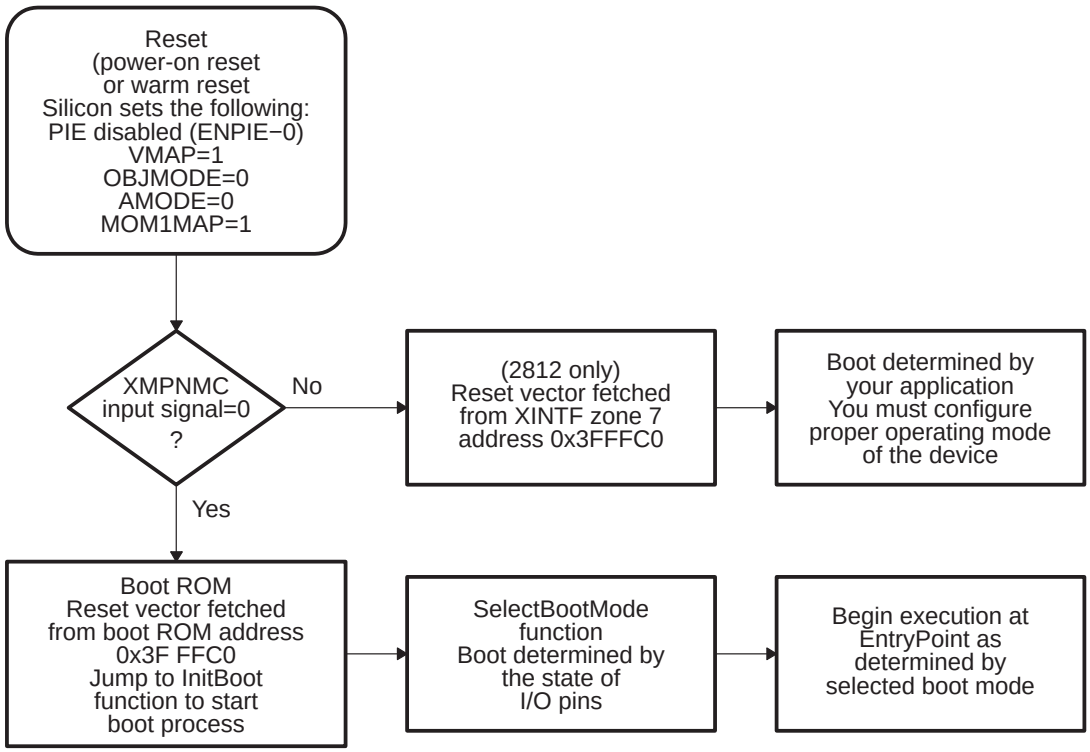
The 281x bootloader is the program located in the 281x ROM that is executed following a reset when the device is in microcomputer mode.

The bootloader is used to transfer code from an external source into internal or external interface (XINTF) memory following power up. This allows code to reside in slow non-volatile memory externally, and be transferred to high-speed memory to be executed.

The bootloader provides a variety of different ways to download code to accommodate different system requirements. These modes are only available if the processor boots in microcomputer mode (XMPNMC device input signal = low).

The bootloader uses various GPIO signals to determine which boot mode to use. The boot mode selection process as well as the specifics of each boot loader operation are described in the remainder of this document. Figure 3 shows the basic bootloader flow.

Figure 3. Bootloader Flow Diagram



Note: On the F2810 the XMPNMC input signal is tied low internally on the device, and therefore boot from reset is always from the internal boot ROM.

At reset, the value of the XMPNMC pin is sampled. The state of this pin determines whether boot ROM or XINTF Zone 7 is enabled at reset.

If XMPNMC = 1 (Micro-Processor Mode) then Zone 7 is enabled and the reset vector will be fetched from external memory. In this case, you must ensure that the reset vector points to a valid memory location for code execution. This option is only available on devices with an XINTF.

If XMPNMC = 0 (Micro-Computer Mode) then the boot ROM memory is enabled and XINTF Zone 7 is disabled. In this case, the reset vector is fetched from the internal boot ROM memory. All devices without an XINTF module have the XMPNMC signal tied low internal to the device such that the boot ROM memory is always enabled at reset.

The reset vector in boot ROM redirects program execution to the InitBoot function. After performing device initialization the boot loader will check the state of GPIO pins to determine which boot mode you want to execute. Options include: Jump to Flash, Jump to H0 SARAM, Jump to OTP or call one of the on-chip boot loading routines.

After the selection process and if required boot loading is complete, the processor will continue execution at an entry point determined by the boot mode selected. If a boot loader was called, then the input stream loaded by the peripheral determines this entry address. This data stream is described in section 2.4.3. If, instead, you choose to boot directly to Flash, OTP, or H0 SARAM, the entry address is predefined for each of these memory blocks.

The following sections discuss in detail the different boot modes available and the process used for loading data code into the device.

4.2 Bootloader Device Configuration

At reset, any 28x™ CPU-based device is in 27x™ object-compatible mode. It is up to the application to place the device in the proper operating mode before execution proceeds.

On the 28x devices, when booting from the internal boot ROM (XMPNMC = 0), the device is configured for 28x operating mode by the boot ROM software. You are responsible for any additional configuration required.

For example:

- If your application includes C2xLP™ source, then you are responsible for configuring the device for C2xLP source compatibility prior to execution of code generated from C2xLP source.
- If you boot from external memory (MPNMC = 1) then the application must configure the device for 28x operating mode or C2xLP source compatible mode as appropriate.

The configuration required for each operating mode is summarized in Table 3.

27x, 28x, and C2xLP are trademarks of Texas Instruments.

Table 3. Configuration for Device Modes

	281x C27x Mode (Reset)	28x Mode	C2xLP Source Compatible Mode
OBJMODE	0	1	1
AMODE	0	0	1
PAGE0	0	0	0
M0M1MAP†	1	1	1
Other Settings			SXM = 1 C = 1 SPM = 0

† Normally for 27x compatibility, the M0M1MAP would be 0. On the 281x; however, it is tied off high internally. Thus at reset M0M1MAP is always configured for 28x mode on these devices.

4.2.1 PLL Multiplier Selection

The Boot ROM does not change the state of the PLL. Note that the PLL multiplier is not affected by a reset from the debugger. Therefore, a boot that is initialized from a reset from Code Composer Studio™ may be at a different speed than booting by pulling the external reset line ($\overline{\text{XRS}}$) low.

4.2.2 Watchdog Module

When branching directly to flash, H0 single-access RAM (SARAM), or one-time-programmable (OTP) memory, the watchdog will not be touched. In the other boot modes, the watchdog will be disabled before booting and then re-enabled and cleared before branching to the final destination address.

4.2.3 PIE Configuration

The boot modes do not enable the PIE. It is left in its default state, which is disabled.

4.2.4 Reserved Memory

The first 80 words of the M1 memory block (address 0x400 − 0x450) are reserved for stack use during the boot load process. If code is bootloaded into this region there is no error checking to prevent it from corrupting the Boot ROM stack.

4.3 Bootloader Modes

To accommodate different system requirements, the 281x boot ROM offers a variety of different boot modes. This section describes the different boot

modes and gives brief summary of their functional operation. The state of four GPIO pins are used to determine the boot mode desired as shown in Table 4.

Table 4. Boot Mode Selection

GPIOF4	**GPIOF12**	**GPIOF3**	**GPIOF2**	
(SCITXDA)	(MDXA)	(SPISTEA)	(SPICLK)	
PU	No PU	No PU	No PU	Mode Selected
1	x	x	x	Jump to Flash address 0x3F 7FF6 You must have programmed a branch instruction here prior to reset to re-direct code execution as desired.
0	1	x	x	Call SPI_Boot to load from an external serial SPI EEPROM
0	0	1	1	Call SCI_Boot to load from SCI-A
0	0	1	0	Jump to H0 SARAM address 0x3F 8000
0	0	0	1	Jump to OTP address 0x3D 7800
0	0	0	0	Call Parallel_Boot to load from GPIO Port B

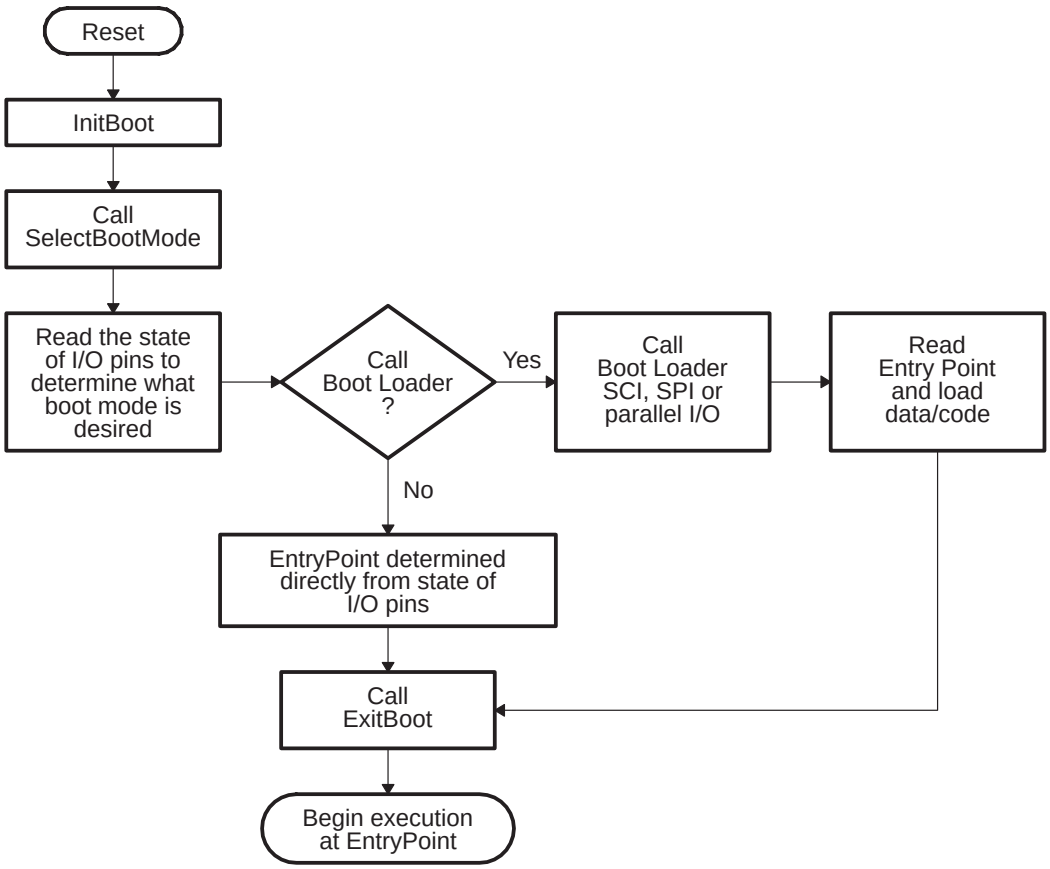
Notes: 1) PU = pin has an internal pullup No PU = pin does not have an internal pullup

2) You must take extra care due to any affect toggling SPICLK in order to select a boot mode may have on external logic.

3) If the boot mode selected is Flash, H0 or OTP, then no external code is loaded by the bootloader.

Figure 4 shows an overview of the boot process. Each step is described in greater detail in following sections.

Figure 4. Boot ROM Function Overview



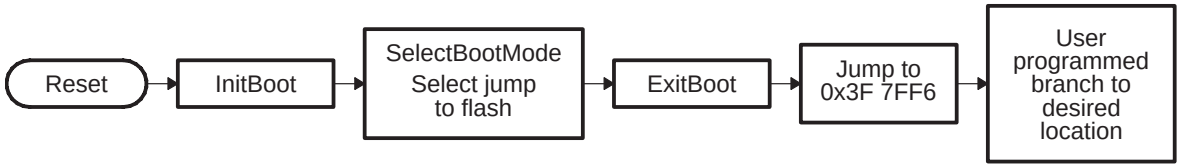
The following boot modes do not call a boot loader. Instead, they jump to a predefined location in memory:

- Jump to branch instruction in Flash Memory:

  In this mode, the boot ROM software will configure the device for ‘28x operation and then branch directly to location 0x3F 7FF6 in Flash memory. This location is just before the 128-bit code security module (CSM) password locations. You are required to have previously programmed a branch instruction at location 0x3F 7FF6 that will redirect code execution to either a custom boot-loader or the application code.

  On 281x RAM devices, the boot-to-Flash option jumps to reserved memory and should not be used. On 281x ROM devices, the boot-to-Flash option jumps to the location 0x3F 7FF6 in ROM.

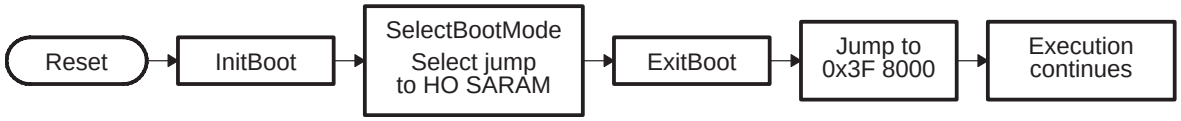
Figure 5. Jump to Flash Flow Diagram



- Jump to H0 SARAM

  In this mode, the boot ROM software will configure the device for ‘28x operation and then branch directly to 0x3F 8000; the first address in the H0 SARAM memory block.

Figure 6. Flow Diagram of Jump to H0 SARAM

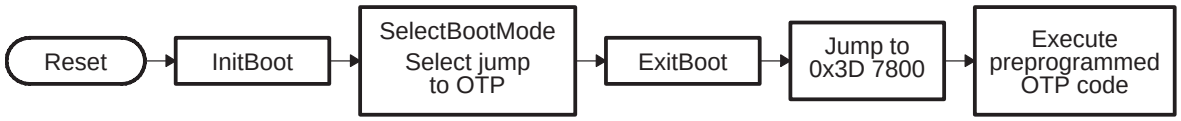


- Jump to OTP Memory

  In this mode, the boot ROM software will configure the device for C28x operation and then branch directly to at 0x3D 7800; the first address in the OTP memory block.

  On 281x ROM devices, the boot-to-OTP option jumps to address 0x3D 7800 in OTP. On 281x RAM devices, the boot-to-OTP option jumps to reserved memory and should not be used.

Figure 7. Flow Diagram of Jump to OTP Memory



- Standard Serial Boot Mode (SCI):

  In this mode, the boot ROM will load code to be executed into on-chip memory via the SCI-A port.

- SPI EEPROM Boot Mode:

  In this mode, the boot ROM will load code and data into on-chip memory from an external EEPROM via the SPI port.

- Boot from GPIO Port:

  In this mode, the boot ROM uses a GPIO port B to load code and data from an external source. This mode supports both 8-bit and 16-bit data

streams. Since this mode requires a number of GPIO pins, it would typically be used for downloading code only for flash programming when the device is connected to a platform explicitly for flash programming and not a target board.

4.4 Bootloader Data Stream Structure

The following two tables and associated examples show the structure of the data stream incoming to the boot loader. The basic structure is the same for all the boot loaders and is based on the C54x source data stream generated by the C54x hex utility. The C28x hex utility has been updated to support this structure. All values in the data stream structure are in hex.

The first 16-bit word in the data stream is known as the key value. The key value is used to tell the boot loader the width of the incoming stream: 8 or 16 bits. Note that not all boot loaders will accept both 8 and 16-bit streams. Please refer to the detailed information on each loader for the valid data stream width. For an 8-bit data stream, the key value is 0x08AA and for a 16-bit stream it is 0x10AA. If a boot loader receives an invalid key value, then the load is aborted. In this case, the entry point for the Flash memory will be used.

The next 8 words are used to initialize register values or otherwise enhance the boot loader by passing values to it. If a boot loader does not use these values then they are reserved for future use and the boot loader simply reads the value and then discards it. Currently only the SPI boot loader uses one word to initialize registers.

The next 10^{th} and 11^{th} words comprise the 22-bit entry point address. This address is used to initialize the PC after the boot load is complete. This address is most likely the entry point of the program downloaded by the boot loader.

The twelfth word in the data stream is the size of the first data block to be transferred. The size of the block is defined for both 8 and 16-bit data stream formats as the number of 16-bit words in the block. For example, to transfer a block of 20 8-bit data values from an 8-bit data stream, the block size would be 0x000A to indicate 10 16-bit words.

The next two words tell the loader the destination address of the block of data. Following the size and address will be the 16-bit words that makeup that block of data.

This pattern of block size/destination address repeats for each block of data to be transferred. Once all the blocks have been transferred, a block size of 0x0000 signals to the loader that the transfer is complete. At this point the

loader will return the entry point address to the calling routine which in turn will cleanup and exit. Execution will then continue at the entry point address as determined by the input data stream contents.

Table 5. General Structure Of Source Program Data Stream In 16-Bit Mode

Word	Contents
1	10AA (KeyValue for memory width = 16bits)
2	Register initialization value or reserved for future use
3	Reserved for future use
	…
9	Reserved for future use
10	Entry point PC[22:16]
11	Entry point PC[15:0]
12	Block size (number of words) of the first block of data to load. If the block size is 0, this indicates the end of the source program. Otherwise another section follows.
13	Destination address of first block Addr[31:16]
14	Destination address of first block Addr[15:0]
15	First word of the first block in the source being loaded
.	…
.	…
.	Last word of the first block of the source being loaded
.	Block size of the 2nd block to load.
.	Destination address of second block Addr[31:16]
.	Destination address of second block Addr[15:0]
.	First word of the second block in the source being loaded
.	…
.	…
.	Last word of the second block of the source being loaded
	…
	…

Table 5. General Structure Of Source Program Data Stream In 16-Bit Mode (Continued)

Word	Contents
.	Block size of the last block to load
.	Destination address of last block Addr[31:16]
.	Destination address of last block Addr[15:0]
.	First word of the last block in the source being loaded
.	…
.	…
.	Last word of the last block of the source being loaded
n	0000h − indicates end of the source program

Example 1. Data stream structure 16bit:

```
10AA   ; 0x10AA 16-bit key value
0000   ;      8 reserved words
0000
0000
0000
0000
0000
0000
0000
003F   ; 0x003F8000 EntryAddr, starting point after boot load completes
8000
0005   ; 0x0005 – First block consists of 5 16-bit words
003F   ; 0x003F9010 – First block will be loaded starting at 0x3F9010
9010
0001   ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
0002
0003
0004
0005
0002   ; 0x0002 – 2nd block consists of 2 16-bit words
003F   ; 0x003F8000 – 2nd block will be loaded starting at 0x3F8000
8000
7700   ; Data loaded = 0x7700 0x7625
7625
0000   ; 0x0000 – Size of 0 indicates end of data stream
```

After load has completed the following memory values will have been initialized as follows:

```
Location  Value
0x3F9010  0x0001
0x3F9011  0x0002
0x3F9012  0x0003
0x3F9013  0x0004
0x3F9014  0x0005
0x3F8000  0x7700
0x3F8001  0x7625
PC    Begins execution at 0x3F8000
```

4.5 General Structure of Source Program Data Stream in 8-Bit Mode

In 8-bit mode, the LSB of the word is sent first followed by the MSB. The boot loaders take this into account when loading an 8-bit data stream.

Table 6. LSB/MSB Loading Sequence in 8-Bit Data Stream

Byte	Contents
1	LSB = AA (KeyValue for memory width = 8 bits)
2	MSB = 08h (KeyValue for memory width = 8 bits)
3	LSB = Register initialization value or reserved for future use
4	MSB= Register initialization value or reserved for future use
…	…
17	LSB = reserved for future use
18	MSB= reserved for future use
19	LSB: Upper half of Entry point PC[23:16]
20	MSB: Upper half of Entry point PC[31:24] (Note: Always 0x00)
21	LSB: Lower half of Entry point PC[7:0]
22	MSB: Lower half of Entry point PC[15:8]
23	LSB: Block size in words of the first block to load. If the block size is 0, this indicates the end of the source program. Otherwise another block follows. For example, a block size of 0x000A would indicate 10 words or 20 bytes in the block.
24	MSB: block size
25	LSB: Upper half of Destination address of first block Addr[23:16]
26	MSB: Upper half of Destination address of first block Addr[31:24]
27	LSB: Lower half of Destination address of first block Addr[7:0]
28	MSB: Lower half of Destination address of first block Addr[15:8]
29	LSB: First word of the first block being loaded
30	MSB: First word of the first block being loaded
.	…
	…
.	LSB: Last word of the first block of the source being loaded

Table 6. LSB/MSB Loading Sequence in 8-Bit Data Stream (Continued)

Byte	Contents
.	MSB: Last word of the first block of the source being loaded
.	LSB: Block size of the second block
.	MSB: Block size of the second block
.	LSB: Upper half of Destination address of second block Addr[23:16]
.	MSB: Upper half of Destination address of second block Addr[31:24]
.	LSB: Lower half of Destination address of second block Addr[7:0]
.	MSB: Lower half of Destination address of second block Addr[15:8]
.	LSB: First word of the second block being loaded
.	MSB: First word of the second block being loaded
.	…
.	…
.	LSB: Last word of the second block of the source being loaded
.	MSB: Last word of the second block of the source being loaded
.	…
.	…
.	…
.	LSB: Block size of the last block
.	MSB: Block size of the last block
.	LSB: Upper half of Destination address of last block Addr[23:16]
.	MSB: Upper half of Destination address of last block Addr[31:24]
.	LSB: Lower half of Destination address of last block Addr[7:0]
.	MSB: Lower half of Destination address of last block Addr[15:8]
.	LSB: First word of the last block being loaded
.	MSB: First word of the last block being loaded…
.	…
.	…

Table 6. LSB/MSB Loading Sequence in 8-Bit Data Stream (Continued)

Byte	Contents
.	LSB: Last word of the last block of the source being loaded
.	MSB: Last word of the last block of the source being loaded
n	LSB: 00h
n+1	MSB: 00h − indicates the end of the source

Example 2. Example 1: Data stream structure 8 bit:

```
AA      ; 0x08AA 8-bit key value
08
00      ; 8 reserved words
00
00
00
00
00
00
00
00
00
00
00
00
00
3F      ; 0x003F8000 EntryAddr, starting point after boot load completes
00
00
80
05      ; 0x0005 – First block consists of 5 16-bit words
00
3F      ; 0x003F9010 – First block will be loaded starting at 0x3F9010
00
10
90
01      ; Data loaded = 0x0001 0x0002 0x0003 0x0004 0x0005
00
02
00
03
00
04
00
05
00
```

```
02     ; 0x0002 – 2nd block consists of 2 16-bit words
00
3F     ; 0x003F8000 – First block will be loaded starting at 0x3F8000
00
00
80
00     ; Data loaded = 0x7700 0x7625
77
25
76
00     ; 0x0000 – Size of 0 indicates end of data stream
00
```

After load has completed the following memory values will have been initialized as follows:

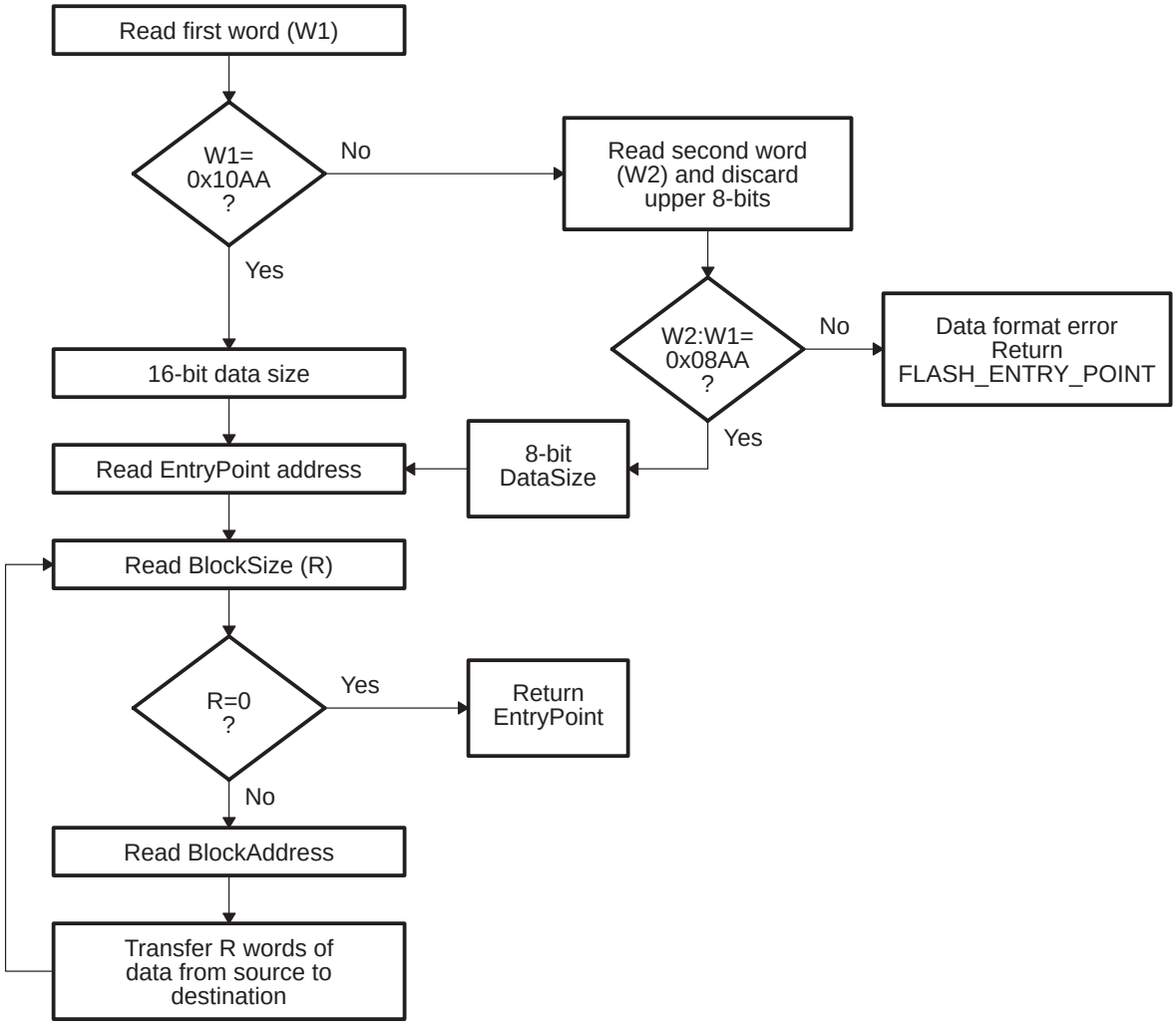
```
Location  Value
0x3F9010  0x0001
0x3F9011  0x0002
0x3F9012  0x0003
0x3F9013  0x0004
0x3F9014  0x0005
0x3F8000  0x7700
0x3F8001  0x7625
PC        Begins execution at 0x3F8000
```

4.6 Basic Transfer Procedure

Figure 8 illustrates the basic process a boot loader uses to determine whether 8-bit or 16-bit data stream has been selected, transfer that data, and start program execution. This process occurs after the boot loader finds the valid boot mode selected by the state of GPIO pins.

The loader first compares the first value sent by the host against the 16-bit key value of 0x10AA. If the value fetched does not match then the loader will read a second value. This value will be combined with the first value to form a word. This will then be checked against the 8-bit key value of 0x08AA. If the loader finds that the header does not match either the 8-bit or 16-bit key value, or if the value is not valid for the given boot mode then the load will abort. In this case the loader will return the entry point address for the flash to the calling routine.

Figure 8. F2810/12 Boot Loader Basic Transfer Procedure



Notes: 1) 8-bit and 16-bit transfers are not valid for all boot modes. Refer to the info specific to a particular boot loader for any limitations.

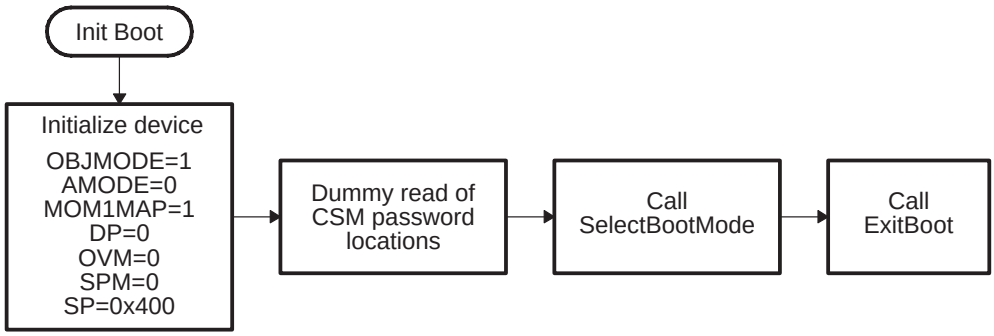
2) In 8-bit mode the LSB of the 16-bit word is read first followed by the MSB.

4.7 InitBoot Assembly Routine

The first routine called after reset is the InitBoot assembly routine. This routine initializes the device for operation in C28x object mode. Next it performs a dummy read of the Code Security Module (CSM) password locations. If the CSM passwords are erased (all 0xFFFFs) then this has the effect of unlocking the CSM. Otherwise the CSM will remain locked and this dummy read of the password locations will have no effect. This can be useful if you have a new device that you want to boot load.

After the dummy read of the CSM password locations, the InitBoot routine calls the SelectBootMode function. This function will then determine the type of boot mode desired by the state of certain GPIO pins. Once the boot is complete, the SelectBootMode function passes back the EntryAddr to the InitBoot function. InitBoot then calls the ExitBoot routine that then restores CPU registers to their reset state and exits to the EntryAddr that was determined by the boot mode.

Figure 9. Overview of InitBoot Assembly Function



4.8 SelectBootMode Function

To determine the desired boot mode, the SelectBootMode function examines the state of 4 GPIO pins as shown in Table 7. These pins are all part of GPIO Port F and are normally output pins when used in their peripheral function (shown in parenthesis).

Table 7. GPIO Pin Status

GPIOF4	GPIOF12	GPIOF3	GPIOF2	
(SCITXDA)	(MDXA)	(SPISTEA)	(SPICLK)	
PU	No PU	No PU	No PU	Mode Selected
1	x	x	x	Jump to Flash address 0x3F 7FF6 You must have programmed a branch instruction here prior to reset to redirect code execution as desired
0	1	x	x	Call SPI_Boot to load from external EEPROM
0	0	1	1	Call SCI_Boot to load from SCI-A
0	0	1	0	Jump to H0 SARAM address 0x3F 8000
0	0	0	1	Jump to OTP address 0x3D 7800
0	0	0	0	Call Parallel_Boot to load from GPIO Port B

Notes:
1) When booting directly to Flash is assumed that you have previously programmed a branch statement at 0x3F 7FF6 to redirect program flow as desired.
2) When booting directly to OTP or H0, it is assumed that you have previously programmed or loaded code starting at the entry point location.
3) x = don't care
4) You must take extra care due to any affect toggling SPICLK in order to select a boot mode may have on external logic.
5) PU = pin has an internal pull-up resister. No PU = pin does not have an internal pull-up resistor

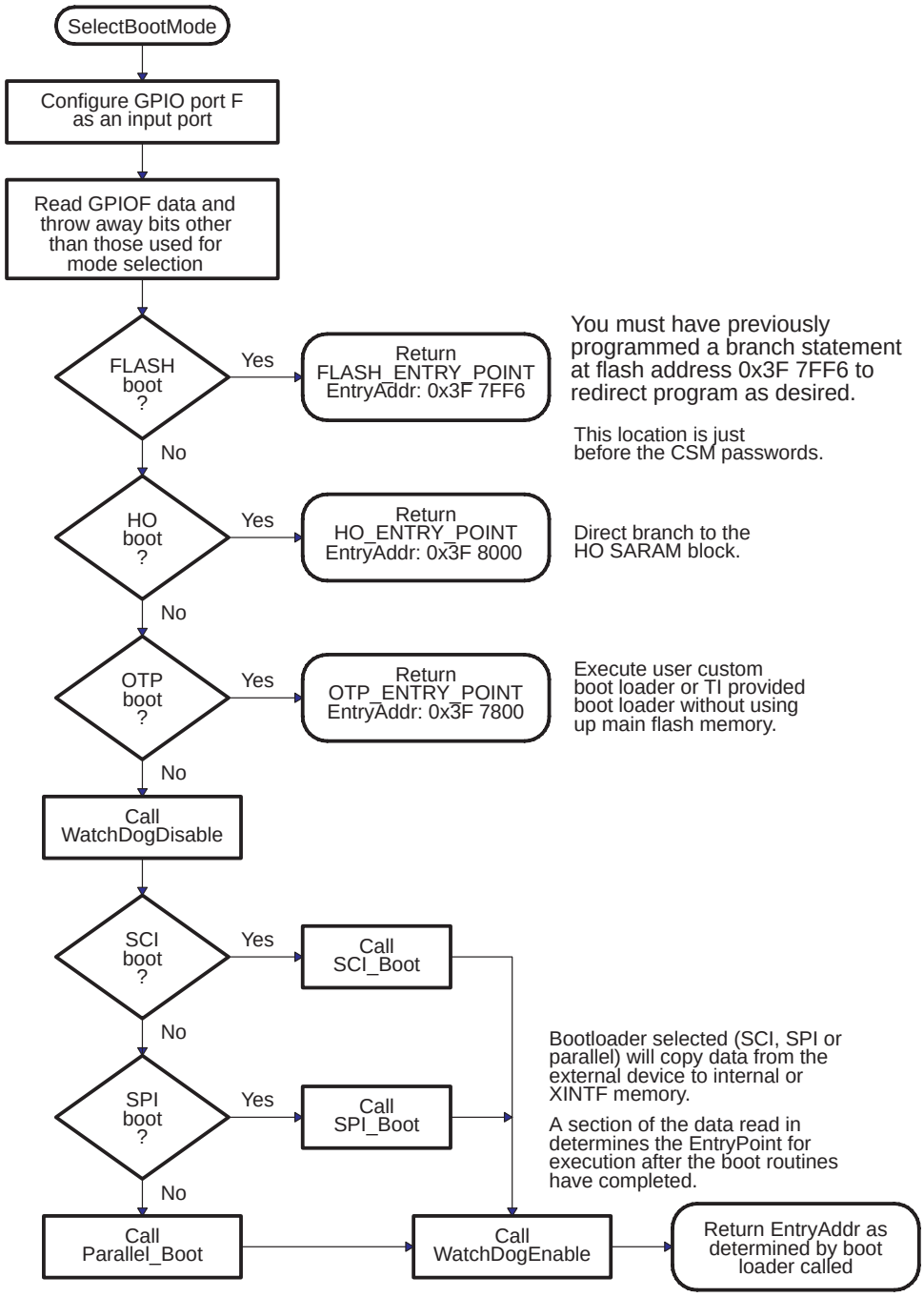
For a boot mode to be selected, the pins corresponding to the desired boot mode have to be pulled low or high until the selection process completes. Note that the state of the selection pins is not latched at reset; they are sampled some cycles later in the SelectBootMode function.

The SelectBootMode routine disables the watchdog before calling the SCI, SPI or Parallel boot loader. If a boot loader is not going to be called, then the watchdog is left untouched. **The boot loaders do not service the watchdog and assume that it is disabled.** Before exiting the SelectBootMode routine will re-enable the watchdog and reset its timer.

When selecting a boot mode, the pins should be pulled high or low through a weak pulldown or weak pull-up such that the DSP can drive them to a new state when required. For example, if you wanted to boot from the SCI one of the pins you pull low is the SCITXDA pin. This pulldown must be weak so that when the SCI boot process begins the DSP will be able to properly transmit through the TX pin. Likewise for the remaining boot mode selection pins.

You must take extra care if you use SPICLK to select a boot mode. Toggling of this signal may have an affect on external logic and this must be taken into account.

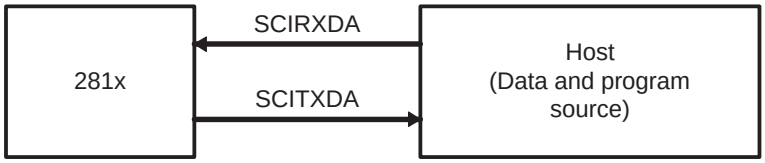
Figure 10. Overview of the SelectBootMode Function



4.9 SCI_Boot Function

The SCI boot mode asynchronously transfers code from SCI-A to internal or XINTF memory. This boot mode only supports an incoming 8-bit data stream and follows the same data flow as outlined in Example 1.

Figure 11. Overview of SCI Boot Loader Operation



The F2810/12 communicates with the external host device by communication through the SCI-A Peripheral. The autobaud feature of the SCI port is used to lock baud rates with the host. For this reason the SCI loader is very flexible and you can use a number of different baud rates to communicate with the DSP.

After each data transfer, the DSP will echo back the 8-bit character received to the host. In this manner, the host can perform checks that each character was received by the DSP.

At higher baud rates, the slew rate of the incoming data bits can be effected by transceiver and connector performance. While normal serial communications may work well, this slew rate may limit reliable auto-baud detection at higher baud rates (typically beyond 100kbaud) and cause the auto-baud lock feature to fail. To avoid this, the following is recommended:

1) Achieve a baud-lock between the host and 28x SCI boot loader using a lower baud rate.

2) Load the incoming 28x application or custom loader at this lower baud rate.

3) The host may then handshake with the loaded 28x application to set the SCI baud rate register to the desired high baud rate.

Figure 12. Overview of SCI_Boot Function

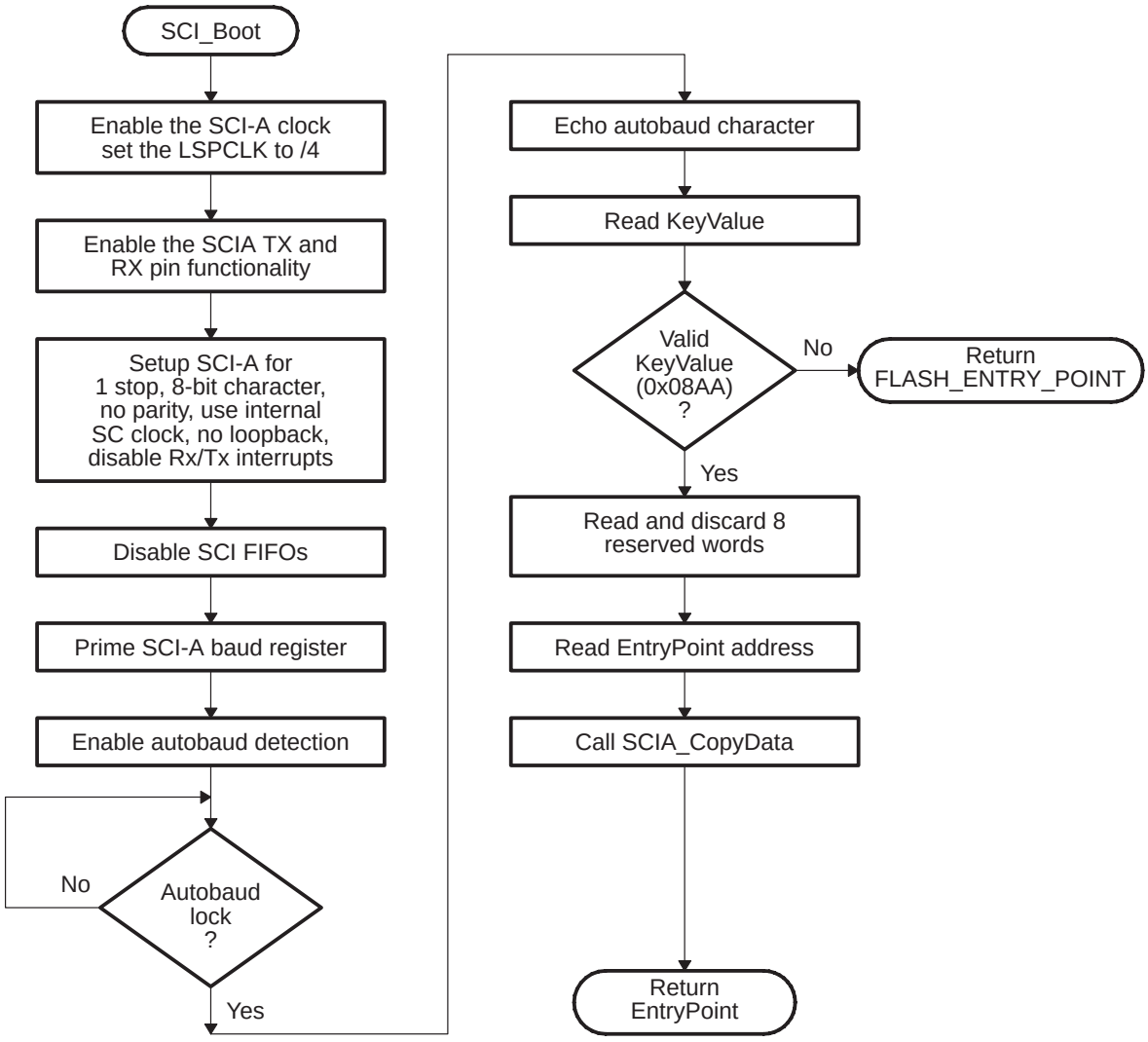


Figure 13. Overview of SCIA_CopyData Function

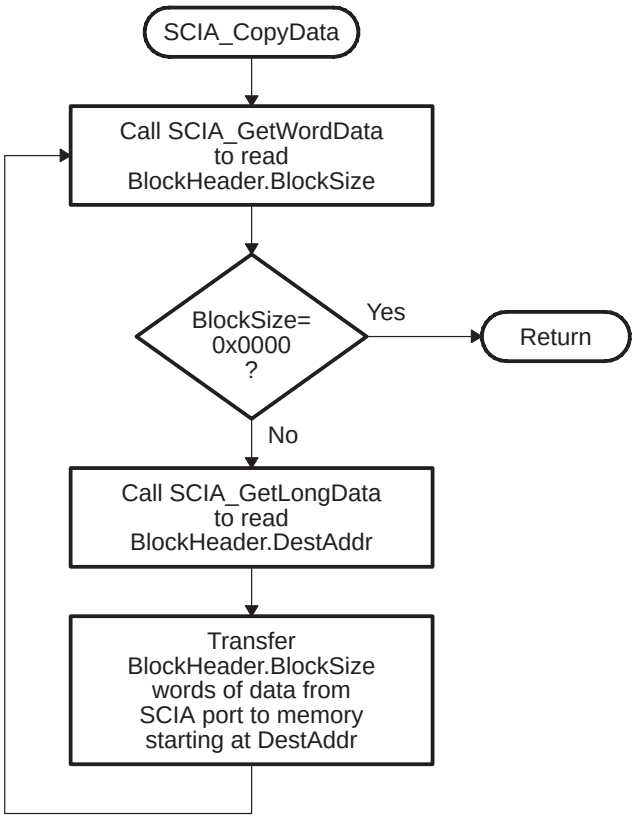
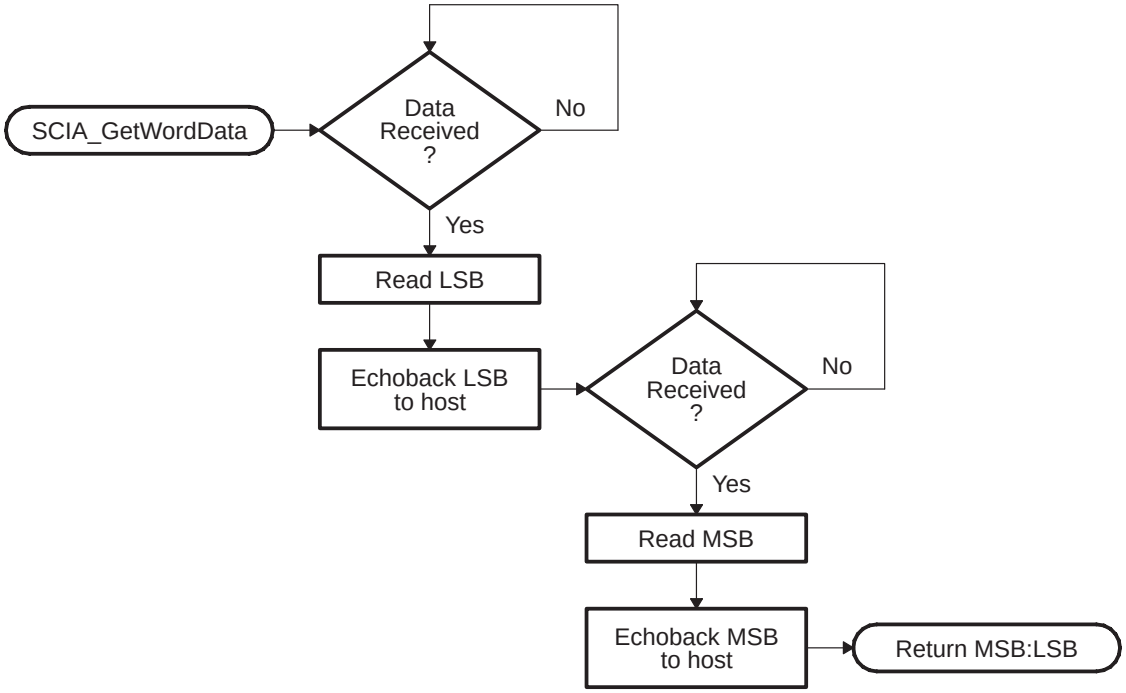


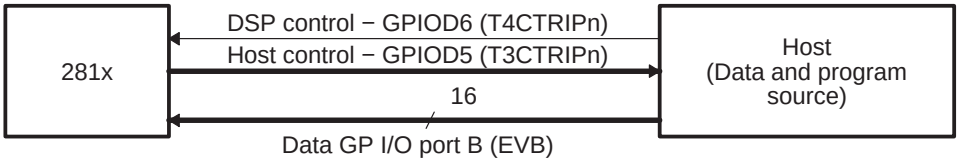
Figure 14. Overview of SCI_GetWordData Function



4.10 Parallel_Boot Function (GPIO)

The parallel general purpose I/O (GPIO) boot mode asynchronously transfers code from GPIO port B to internal or XINTF memory. Each value can be 16 bits or 8 bits long and follows the same data flow as outlined in Data Stream Structure.

Figure 15. Overview of Parallel GPIO Boot Loader Operation



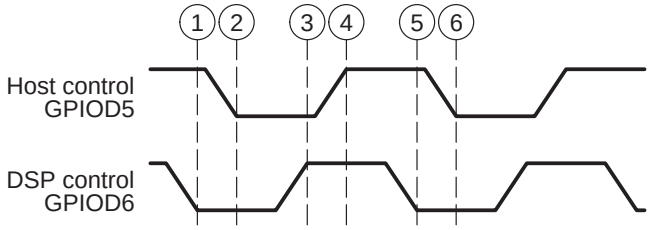
The 28x communicates with the external host device by polling/driving the GPIOD5 and GPIOD6 lines. The handshake protocol shown in Figure 16 must be used to successfully transfer each word via GPIO port B. This protocol is very robust and allows for a slower or faster host to communicate with the 281x device.

If the 8-bit mode is selected, two consecutive 8-bit words are read to form a single 16-bit word. The most significant byte (MSB) is read first followed by the

least significant byte (LSB). In this case, data is read from the lower eight lines of GPIO port B ignoring the higher byte.

The DSP first signals the host that the DSP is ready to begin data transfer by pulling the GPIOD6 pin low. The host load then initiates the data transfer by pulling the GPIOD5 pin low. The complete protocol is shown in the diagram below:

Figure 16. Parallel GPIO Boot loader Handshake Protocol



1) The DSP indicates it is ready to start receiving data by pulling the GPIOD6 pin low.

2) The boot loader waits until the host puts data on GPIO port B. The host signals to the DSP that data is ready by pulling the GPIOD5 pin low.

3) The DSP reads the data and signals the host that the read is complete by pulling GPIOD6 high.

4) The Boot loader waits until the Host acknowledges the DSP by pulling GPIOD5 high.

5) The DSP again indicates it is ready for more data by pulling the GPIOD6 pin low.

This process is repeated for each data value to be sent.

Figure 17 shows an overview of the Parallel GPIO boot loader flow.

Figure 17. Parallel GPIO Mode Overview

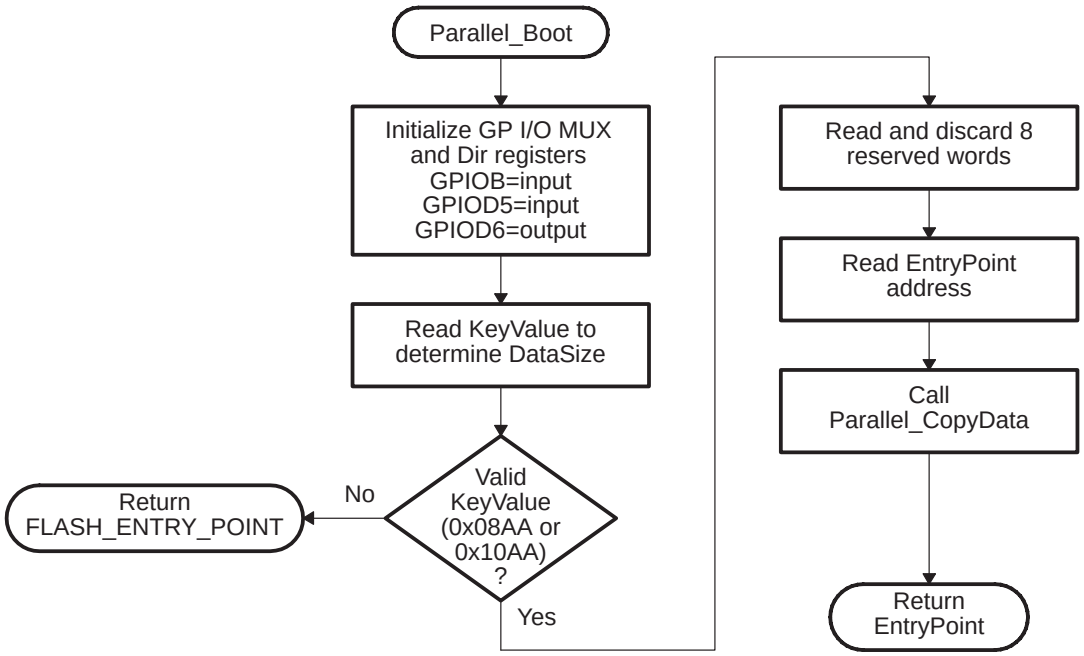


Figure 18 shows the transfer flow from the Host side. The operating speed of the CPU and Host are not critical in this mode as the host will wait for the DSP and the DSP will in turn wait for the host. In this manner the protocol will work with both a host running faster and a host running slower then the DSP.

Figure 18. Parallel GPIO Mode − Host Transfer Flow

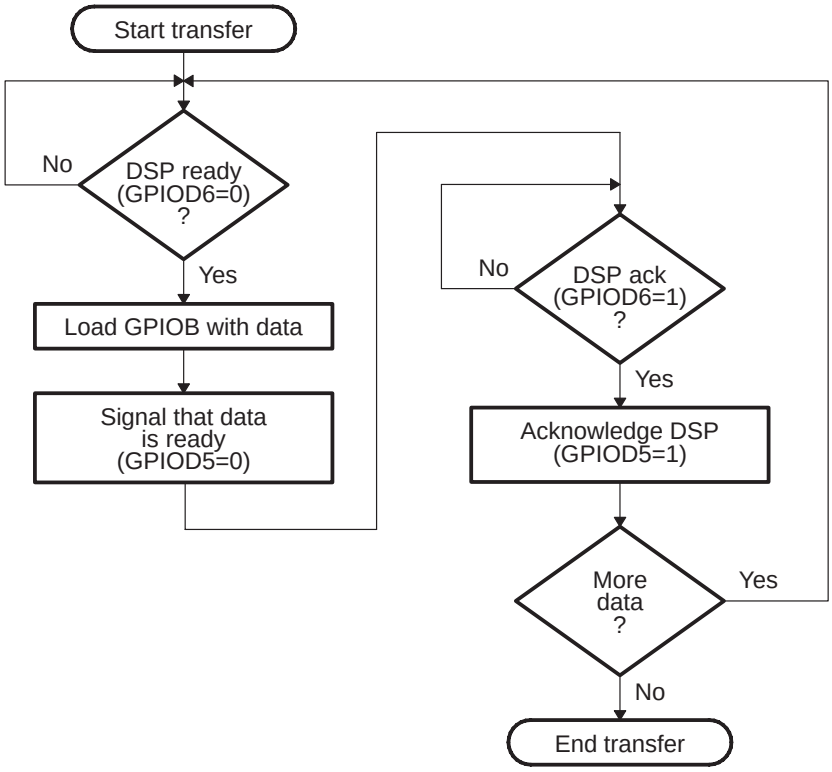


Figure 19. Overview of Parallel_CopyData Function

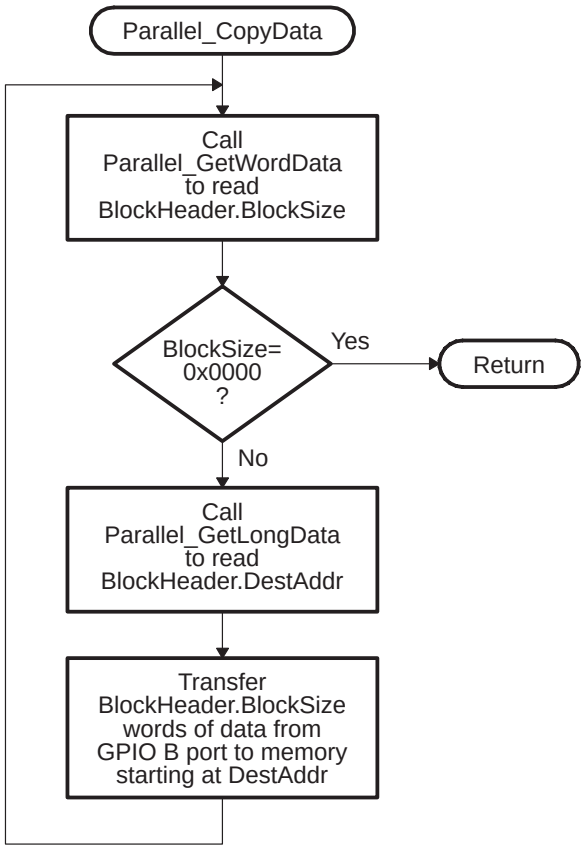
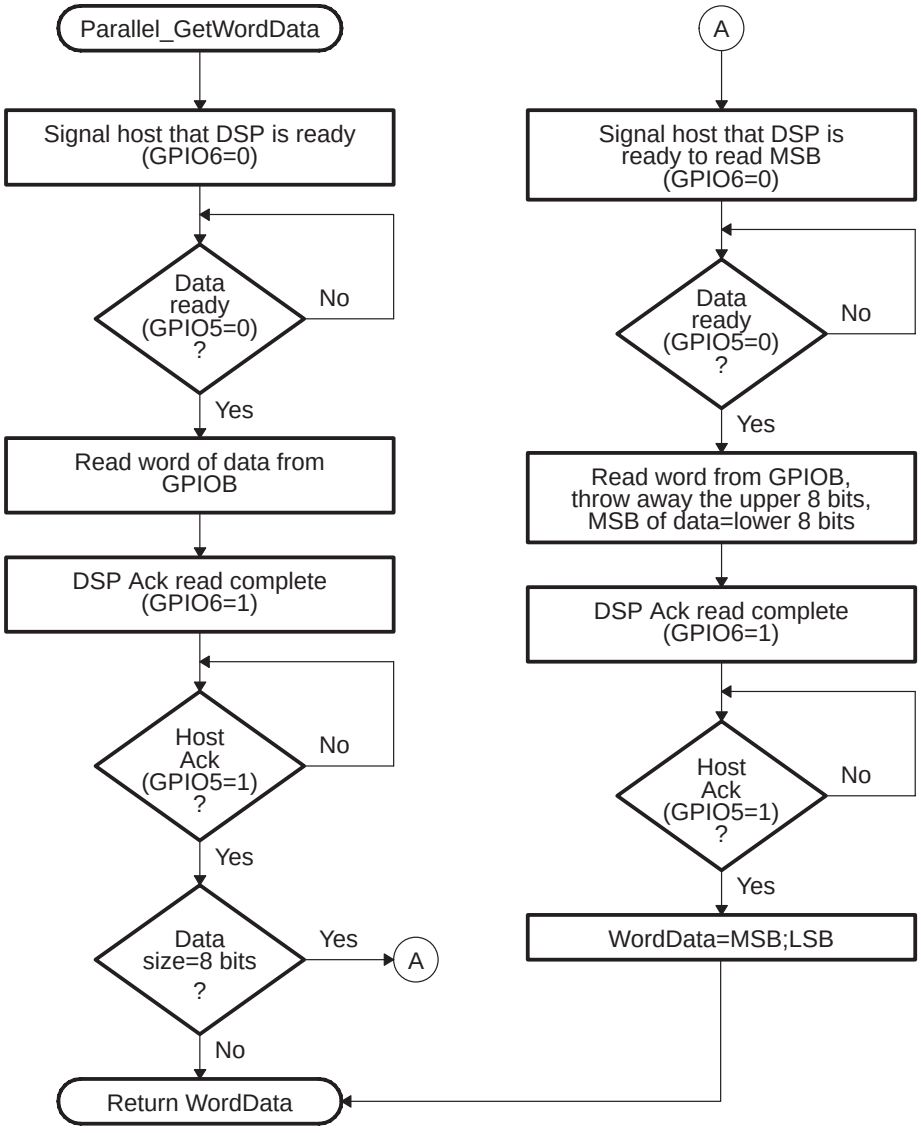


Figure 20 shows the flow for fetching a single word of data from the parallel port.

The routine is passed a DataSize parameter of 8 bits or 16 bits. The routine follows the previously defined protocol flow to read 16 bits from the GPIO B port. If the DataSize is 16 bits, then the routine will pass this 16-bit value back to the calling routine.

If the DataSize parameter is 8 bits, then the routine will and discard the upper 8 bits of the first read and treat the lower 8 bits as the least significant byte (LSB) of the word to be fetched. The routine will then perform a 2nd fetch to read the most significant byte (MSB). It then combines the MSB and LSB into a single 16-bit value to be passed back to the calling routine.

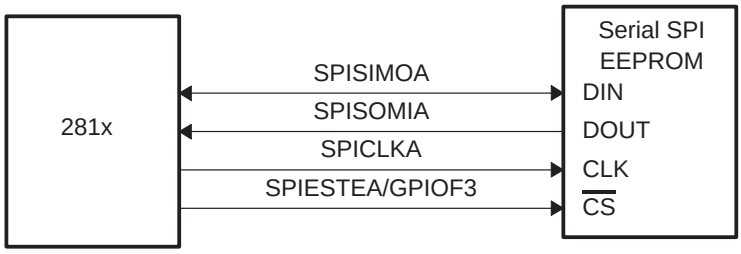
Figure 20. Parallel GPIO Boot Loader Word Fetch



4.11 SPI_Boot Function

The SPI loader expects an 8-bit wide SPI-compatible serial EEPROM device to be present on the SPI pins as indicated in Figure 21. The SPI bootloader does not support a 16-bit data stream.

Figure 21. SPI Loader



The SPI boot ROM loader initializes the SPI module to interface to a serial SPI EEPROM. Devices of this type include, but are not limited to, the Xicor X25320 (4Kx8) and Xicor X25256 (32Kx8) SPI serial SPI EEPROMs.

The SPI boot ROM loader initializes the SPI with the following settings: FIFO enabled, 8-bit character, internal SPICLK master mode and talk mode, clock phase = 0, polarity = 0, using the slowest baud rate.

If the download is to be preformed from an SPI port on another device, then that device must be setup to operate in the slave mode and mimic a serial SPI EEPROM. Immediately after entering the SPI_Boot function, the pin functions for the SPI pins are set to primary and the SPI is initialized. The initialization is done at the slowest speed possible. Once the SPI is initialized and the key value read, you could specify a change in baud rate or low speed peripheral clock.

Table 8. SPI 8-Bit Data Stream

Byte	Contents
1	LSB = AA (KeyValue for memory width = 8-bits)
2	MSB = 08h (KeyValue for memory width = 8-bits)
3	LSB = LOSPCP
4	MSB= SPIBRR
5	LSB = reserved
6	MSB = reserved

Table 8. SPI 8-Bit Data Stream (Continued)

Byte	Contents
17	LSB = reserved for future use
18	MSB= reserved for future use
19	LSB: Upper half of Entry point PC[23:16]
20	MSB: Upper half of Entry point PC[31:24] (Note: Always 0x00)
21	LSB: Lower half of Entry point PC[7:0]
22	MSB: Lower half of Entry point PC[15:8]
	Blocks of data in the format size/destination address/data as shown in the generic data stream description
	LSB: 00h
	MSB: 00h − indicates the end of the source

The data transfer is done in “burst” mode from the serial SPI EEPROM. The transfer is carried out entirely in byte mode (SPI at 8 bits/character). A step-by step description of the sequence follows:

1) The SPI-A port is initialized
2) The GPIOF3 pin is now used as a chip-select for the serial SPI EEPROM
3) The SPI-A outputs a read command for the serial SPI EEPROM
4) The SPI-A sends the serial SPI EEPROM an address 0x0000; that is, the host requires that the EEPROM must have the downloadable packet starting at address 0x0000 in the EEPROM.
5) The next word fetched must match the key value for an 8-bit data stream (0x08AA).

   The least significant byte of this word is the byte read first and the most significant byte is the next byte fetched. This is true of all word transfers on the SPI.

   If the key value does not match then the load is aborted and the entry point for the Flash (0x3F 7FF6) is returned to the calling routine.
6) The next 2 bytes fetched can be used to change the value of the low speed peripheral clock register (LOSPCP) and the SPI Baud rate register (SPIBRR). The first byte read is the LOSPCP value and the 2nd byte read is the SPIBRR value.

   The next 7 words are reserved for future enhancements. The SPI boot loader reads these 7 words and discards them.

7) The next 2 words makeup the 32-bit entry point address where execution will continue after the boot load process is complete. This is typically the entry point for the program being downloaded through the SPI port.

8) Multiple blocks of code and data are then copied into memory from the external serial SPI EEPROM through the SPI port. The blocks of code are organized in the standard data stream structure presented earlier. This is done until a block size of 0x0000 is encountered. At that point in time the entry point address is returned to the calling routine that then exits the boot loader and resumes execution at the address specified.

Figure 22. Data Transfer From EEPROM Flow

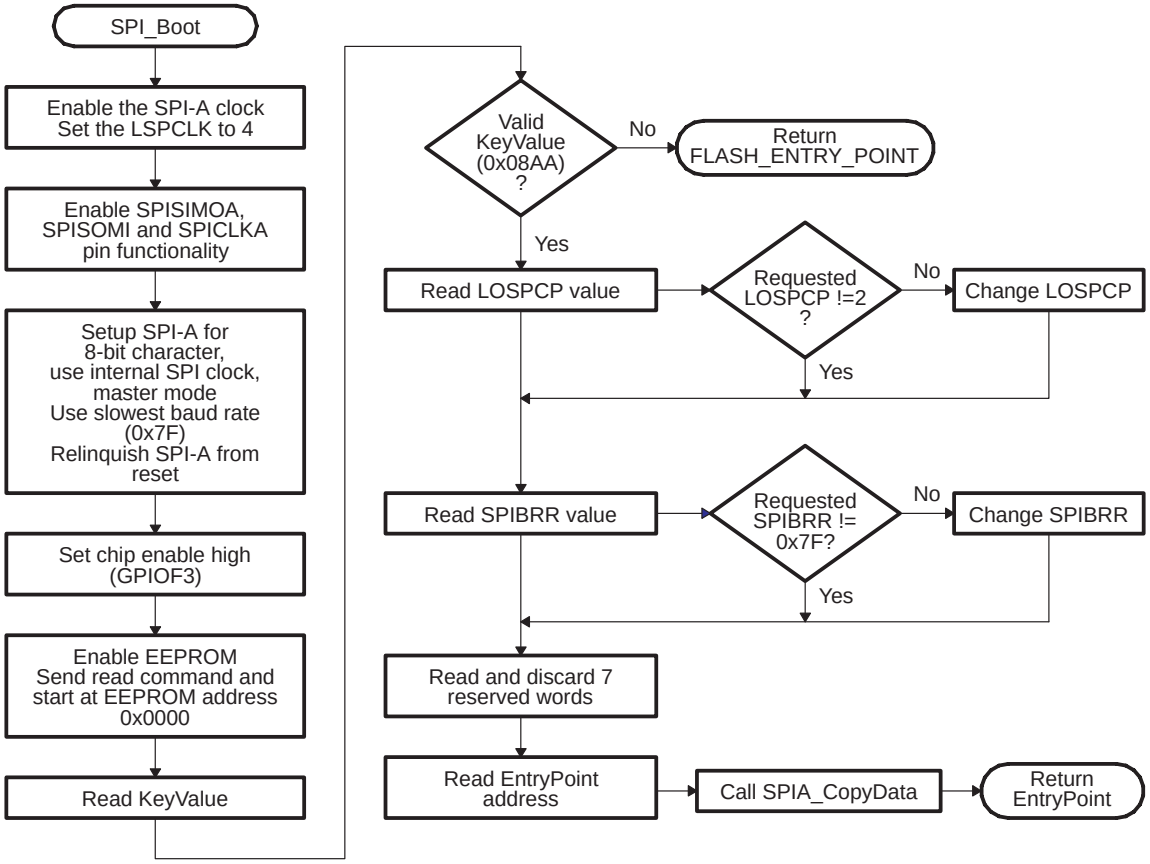


Figure 23. Overview of SPIA_CopyData Function

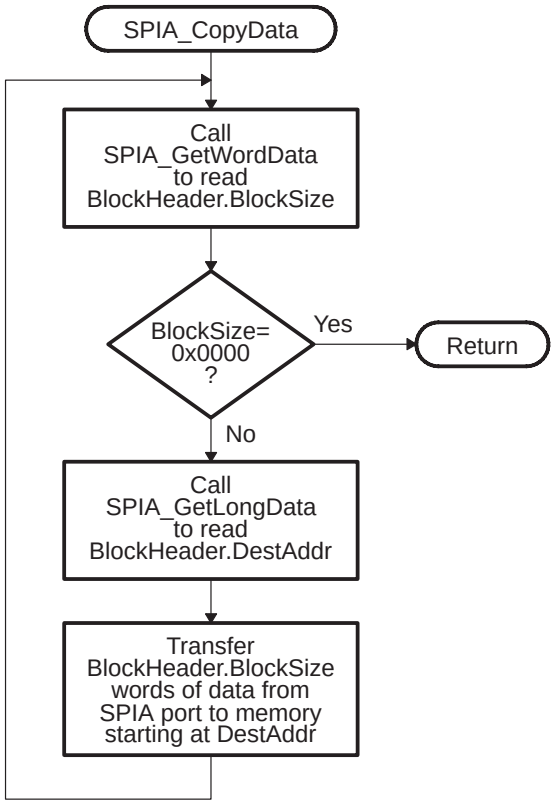
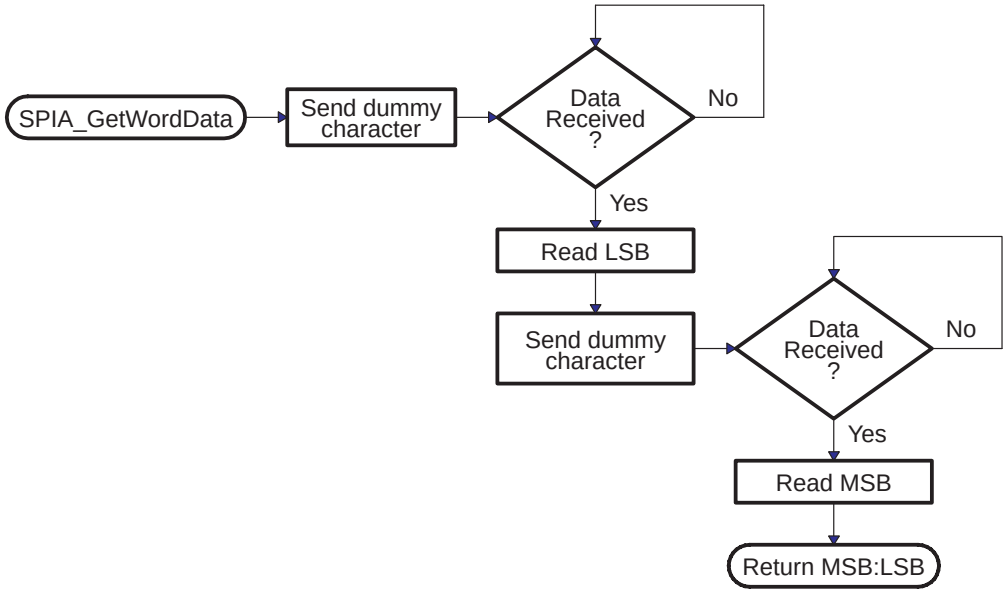


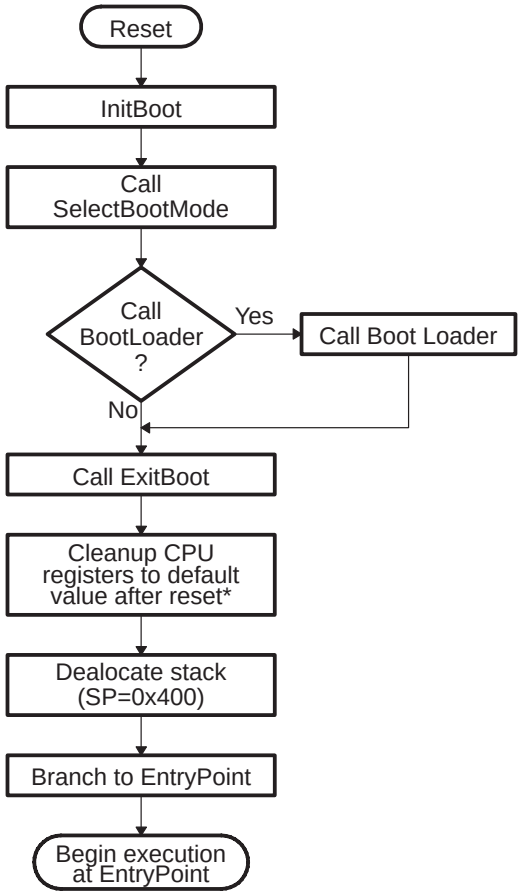
Figure 24. Overview of SPIA_GetWordData Function



4.12 ExitBoot Assembly Routine

The 281x Boot Rom includes a ExitBoot routine that restores the CPU registers to their default state at reset. This is performed on all registers with one exception. The OBJMODE bit in ST1 is left set so that the device remains configured for C28x operation. This flow is detailed in the following diagram:

Figure 25. ExitBoot Procedure Flow



The following CPU registers are restored to their default values:

ACC = 0x0000 0000

RPC = 0x0000 0000

P = 0x0000 0000

XT = 0x0000 0000

ST0 = 0x0000

ST1 = 0x0A0B

XAR0 = XAR7 = 0x0000 0000

After the ExitBoot routine completes and the program flow is redirected to the entry point address, the CPU registers will have the following values:

Table 9. CPU Register Values

Register	Value	Register	Value
ACC	0x0000 0000	P	0x0000 0000
XT	0x0000 0000	RPC	0x00 0000
XAR0–XAR7	0x0000 0000	DP	0x0000
ST0	0x0000 15:10 OVC = 0 9:7 PM = 0 6 V = 0 5 N = 0 4 Z = 0 3 C = 0 2 TC = 0 1 OVM = 0 0 SXM = 0	ST1	0x0A0B 15:13 ARP = 0 12 XF = 0 11 M0M1MAP = 1 10 reserved 9 OBJMODE = 1 8 AMODE = 0 7 IDLESTAT = 0 6 EALLOW = 0 5 LOOP = 0 4 SPA = 0 3 VMAP = 1 2 PAGE0 = 0 1 DBGM = 1 0 INTM = 1

5 Building the Boot Table

To use the features of the 281x bootloader, you must first generate a boot table that contains the complete data stream the bootloader needs. The hex conversion utility tool included with the 28x code generation tools generates the boot table. The contents of the boot table vary slightly depending on the boot mode and the options selected when running the hex conversion utility.

The hex utility supports creation of the boot table required for the SCI, SPI, and parallel I/O loaders. That is, the hex utility adds the required information to the file such as the key value, reserved bits, entry point, address, block start address, block length and terminating value. The actual file format required by the host (ASCII, binary, hex, etc.) will differ from one specific application to another and some additional conversion may be required.

To build the 281x boot table, follow these steps:

Step 1: Assemble (or compile) the code.

Step 2: Link the file. Each block of the boot table data corresponds to an initialized section in the COFF file. Uninitialized sections are not converted by the hex conversion utility.

Step 3: Run the hex conversion utility. Choose the appropriate options for the desired boot mode and run the hex conversion utility to convert the COFF file produced by the linker to a boot table.

This table summarizes the hex conversion utility options available for the boot loader**.** See the *TMS320C28x Assembly Language Tools User's Guide* (SPRU513) for a detailed description for the procedure for generating a boot table and the required options.

Table 10. Boot-Loader Options

Option	Description
−boot	Convert all sections into bootable form (use instead of a SECTIONS directive)
−sci8	Specify the source of the boot loader table as the SCI-A port, 8-bit mode
−spi8	Specify the source of the boot loader table as the SPI-A port, 8-bit mode
−gpio8	Specify the source of the boot loader table as the GP I/O port, 8-bit mode
−gpio16	Specify the source of the boot loader table as the GP I/O port, 16-bit mode
−bootorg value	Specify the source address of the boot loader table
−lospcp value	Specify the initial value for the LOSPCP register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
−spibrr value	Specify the initial value for the SPIBRR register. This value is used only for the spi8 boot table format and ignored for all other formats. If the value is greater than 0x7F, the value is truncated to 0x7F.
−e value	Specify the entry point at which to begin execution after boot loading. The value can be an address or a global symbol. This value is optional. The entry point can be defined at compile time using the linker −e option to assign the entry point to a global symbol. The entry point for a C program is normally −c.intOO unless defined otherwise by the −e linker option.

6 Bootloader Code Listing

```
//###########################################################################
//
// FILE:      F2812_Boot.h
//
// TITLE:     F2812 Boot ROM Definitions.
//
//###########################################################################
//
//  Ver | dd mmm yyyy |     Who      | Description of changes
// =====|=============|==============|======================================
//  0.1 | 30 Jan 2002 | LH           | Original Release.
// -----|-------------|--------------|--------------------------------------
//      |             |              |
//
//###########################################################################
#ifndef F2812_BOOT_H
#define F2812_BOOT_H
//---------------------------------------------------------------------------
// Fixed boot entry points:
//
#define FLASH_ENTRY_POINT 0x3F7FF6
#define OTP_ENTRY_POINT   0x3D7800
#define H0_ENTRY_POINT    0x3F8000
#define PASSWORD_LOCATION 0x3F7FF8
// Misc definitions
#define ERROR                   1
#define NO_ERROR                0
#define EIGHT_BIT               8
#define SIXTEEN_BIT            16
#define EIGHT_BIT_HEADER   0x08AA
#define SIXTEEN_BIT_HEADER 0x10AA
//---------------------------------------------------------------------------
// Common CPU Definitions:
//
#define        EALLOW      asm(" EALLOW");
#define        EDIS        asm(" EDIS");
typedef int                i16;
typedef long               i32;
typedef unsigned int       ui16;
typedef unsigned long      ui32;
//---------------------------------------------------------------------------
// Include Peripheral Header Files:
//
#include "SysCtrl_Boot.h"
#include "SPI_Boot.h"
#include "SCI_Boot.h"
#include "Parallel_Boot.h"
#endif  // end of DSP28_DEVICE_H definition

;;###########################################################################
;;
;; FILE:    Init_Boot.asm
```

```
;;
;; TITLE:   F2810/12 Boot Rom Initialization and Exit routines.
;;
;; Functions:
;;
;;     _InitBoot
;;     _ExitBoot
;;
;; Notes:
;;
;;############################################################################
;;
;;  Ver | dd mmm yyyy |     Who      | Description of changes
;; =====|=============|==============|======================================
;;  1.0 | 12 Mar 2002 | LH           | PG Release.
;;      |             |              |
;;############################################################################
    .def _InitBoot
    .ref _SelectBootMode
    .sect ".Version"
    .word 0x0001      ; F2810/12 Boot ROM Version 1
    .word 0x0302      ; Month/Year: 3/02

    .sect ".Checksum" ; 64-bit Checksum
    .long 0x70F3099C  ; least significant 32-bits
    .long 0x00000402  ; most significant 32-bits

    .sect ".InitBoot"
;---------------------------------------------
; _InitBoot
;---------------------------------------------
;---------------------------------------------
; This function performs the initial boot routine
; for the F2810/12 boot ROM.
;
; This module performs the following actions:
;
;     1) Initializes the stack pointer
;     2) Sets the device for C28x operating mode
;     3) Calls the main boot functions
;     4) Calls an exit routine
;---------------------------------------------
_InitBoot:
; Initialize the stack pointer.
__stack:    .usect ".stack",0
    MOV SP, #__stack ; Initialize the stack pointer
                     ; Initialize the device for running in C28x mode.
    C28OBJ           ; Select C28x object mode
    C28ADDR          ; Select C27x/C28x addressing
    C28MAP           ; Set blocks M0/M1 for C28x mode
    CLRC PAGE0       ; Always use stack addressing mode
    MOVW DP,#0       ; Initialize DP to point to the low 64 K
    CLRC OVM
```

```
; Set PM shift of 0
    SPM 0
; Read the password locations - this will unlock the
; CSM only if the passwords are erased. Otherwise it
; will not have an effect.
  MOVL XAR1,#0x3F7FF8;
  MOVL XAR0,*XAR1++
  MOVL XAR0,*XAR1++
  MOVL XAR0,*XAR1++
  MOVL XAR0,*XAR1
; Decide which boot mode to use
    LCR  _SelectBootMode
; Cleanup and exit.  At this point the EntryAddr
; is located in the ACC register
    BF  _ExitBoot,UNC
;-----------------------------------------------
; _ExitBoot
;-----------------------------------------------
;-----------------------------------------------
;This module cleans up after the boot loader
;
; 1) Make sure the stack is deallocated.
;    SP = 0x400 after exiting the boot
;    loader
; 2) Push 0 onto the stack so RPC will be
;    0 after using LRETR to jump to the
;    entry point
; 2) Load RPC with the entry point
; 3) Clear all XARn registers
; 4) Clear ACC, P and XT registers
; 5) LRETR - this will also clear the RPC
;    register since 0 was on the stack
;-----------------------------------------------
_ExitBoot:
;-----------------------------------------------
;   Ensure that the stack is deallocated
;-----------------------------------------------
    MOV SP,#__stack
;-----------------------------------------------
; Clear the bottom of the stack.  This will endup
; in RPC when we are finished
;-----------------------------------------------
    MOV  *SP++,#0
    MOV  *SP++,#0
;-----------------------------------------------
; Load RPC with the entry point as determined
; by the boot mode.  This address will be returned
; in the ACC register.
;-----------------------------------------------
    PUSH ACC
    POP  RPC
;-----------------------------------------------
; Put registers back in their reset state.
```

```
;
; Clear all the XARn, ACC, XT, and P and DP
; registers
;
; NOTE: Leave the device in C28x operating mode
;       (OBJMODE = 1, AMODE = 0)
;-----------------------------------------------
    ZAPA
    MOVL  XT,ACC
    MOVZ  AR0,AL
    MOVZ  AR1,AL
    MOVZ  AR2,AL
    MOVZ  AR3,AL
    MOVZ  AR4,AL
    MOVZ  AR5,AL
    MOVZ  AR6,AL
    MOVZ  AR7,AL
    MOVW  DP, #0
;-----------------------------------------------
;   Restore ST0 and ST1.  Note OBJMODE is
;   the only bit not restored to its reset state.
;   OBJMODE is left set for C28x object operating
;   mode.
;
;  ST0 = 0x0000     ST1 = 0x0A0B
;  15:10 OVC = 0    15:13      ARP = 0
;   9: 7  PM = 0       12       XF = 0
;      6   V = 0       11  M0M1MAP = 1
;      5   N = 0       10  reserved
;      4   Z = 0        9  OBJMODE = 1
;      3   C = 0        8    AMODE = 0
;      2  TC = 0        7 IDLESTAT = 0
;      1 OVM = 0        6   EALLOW = 0
;      0 SXM = 0        5     LOOP = 0
;                       4      SPA = 0
;                       3     VMAP = 1
;                       2    PAGE0 = 0
;                       1     DBGM = 1
;                       0     INTM = 1
;-----------------------------------------------
    MOV  *SP++,#0
    MOV  *SP++,#0x0A0B
    POP  ST1
    POP  ST0
;-----------------------------------------------
;   Jump to the EntryAddr as defined by the
;   boot mode selected and continue execution
;-----------------------------------------------
    LRETR

;eof ----------

//###########################################################################
```

```
//
// FILE:    SelectMode_Boot.c
//
// TITLE:   F2810/12 Boot Mode selection routines
//
// Functions:
//
//      ui32   SelectBootMode(void)
//      inline void SelectMode_GPOISelect(void)
//
// Notes:
//
//###########################################################################
//
//  Ver | dd mmm yyyy |     Who      | Description of changes
// =====|=============|==============|======================================
//  1.0 | 12 Mar 2002 | LH           | PG Release.
//      |             |              |
//###########################################################################
#include "F2812_Boot.h"
inline void SelectMode_GPIOSelect(void);
// Define mask for mode selection
// all mode select pins are on GPIOF
// These definitions define which pins
// are used:
//GPIOF4     GPIOF12  GPIOF3    GPIOF2
//(SCITXDA)  (MDXA)  (SPISTEA)  (SPICLKA) Mode Selected
//   1          x       x         x       Jump to Flash address 0x3F 7FF6
//   0          1       x         x       Call SPI_Boot
//   0          0       1         1       Call SCI_Boot
//   0          0       1         0       Jump to H0 SARAM
//   0          0       0         1       Jump to OTP
//   0          0       0         0       Call  Parallel_Boot
#define MODE_PIN1   0x0010  // GPIOF4
#define MODE_PIN2   0x1000  // GPIOF12
#define MODE_PIN3   0x0008  // GPIOF3
#define MODE_PIN4   0x0004  // GPIOF2
// On GP I/O port F use only the following pins to determine the boot mode
#define MODE_MASK    (MODE_PIN1 | MODE_PIN2 | MODE_PIN3 | MODE_PIN4)
// Define which pins matter to which modes.  Note that some modes
// do not check the state of all 4 pins.
#define FLASH_MASK     (MODE_PIN1)
#define SPI_MASK       (MODE_PIN1 | MODE_PIN2)
#define SCI_MASK       (MODE_PIN1 | MODE_PIN2 | MODE_PIN3 | MODE_PIN4)
#define H0_MASK        (MODE_PIN1 | MODE_PIN2 | MODE_PIN3 | MODE_PIN4)
#define OTP_MASK       (MODE_PIN1 | MODE_PIN2 | MODE_PIN3 | MODE_PIN4)
#define PARALLEL_MASK  (MODE_PIN1 | MODE_PIN2 | MODE_PIN3 | MODE_PIN4)
// Define which pins (out of the ones being examined) must be set
// to boot to a particular mode
#define FLASH_MODE     (MODE_PIN1)
#define SPI_MODE       (MODE_PIN2)
#define SCI_MODE       (MODE_PIN3 | MODE_PIN4)
#define H0_MODE        (MODE_PIN3)
```

```
#define OTP_MODE       (MODE_PIN4)
#define PARALLEL_MODE  0x0000
ui32 SelectBootMode()
{
  ui32 EntryAddr;
  ui16 BootMode;

  SelectMode_GPIOSelect();
  BootMode = GPIODataRegs.GPFDAT.all & MODE_MASK;

  // First check for modes which do not require
  // a boot loader (Flash/H0/OTP)

  if( (BootMode & FLASH_MASK) == FLASH_MODE ) return FLASH_ENTRY_POINT;
  if( (BootMode & H0_MASK) == H0_MODE) return H0_ENTRY_POINT;
  if( (BootMode & OTP_MASK) == OTP_MODE) return OTP_ENTRY_POINT;

  // Otherwise, disable the watchdog and check for the
  // other boot modes that requre loaders
  WatchDogDisable();
  if( (BootMode & SCI_MASK) == SCI_MODE) EntryAddr = SCI_Boot();
  else if( (BootMode & SPI_MASK) == SPI_MODE) EntryAddr = SPI_Boot();
  else EntryAddr = Parallel_Boot();
   WatchDogEnable();
   return EntryAddr;
}
//#####################################################
// inline void SelectMode_GPIOSelect(void)
//-----------------------------------------
// Enable GP I/O port F as an input port.
//-----------------------------------------
inline void SelectMode_GPIOSelect()
{
    EALLOW;

    // GPIO Port F is all I/O pins
    // 0 = I/O pin  1 = Peripheral pin
     GPIOMuxRegs.GPFMUX.all = 0x0000;

    // Port F is all input
    // 0 = input   1 = output
    GPIOMuxRegs.GPFDIR.all = 0x0000;
    EDIS;
}
;EOF ----------
//###########################################################################
//
// FILE:    SCI_Boot.c
//
// TITLE:   F2810/12 SCI Boot mode routines
//
// Functions:
//
```

```
//     ui32 SCI_Boot(void)
//     inline void SCIA_GPIOSelect(void)
//     inline void SCIA_SysClockEnable(void)
//     inline void SCIA_Init(void)
//     inline void SCIA_AutobaudLock(void)
//     inline ui16 SCIA_CheckKeyVal(void)
//     inline void SCIA_ReservedFn(void)
//     ui32 SCIA_GetLongData(void)
//     ui32 SCIA_GetWordData(void)
//     void SCIA_CopyData(void)
//
// Notes:
//
//###########################################################################
//
//  Ver | dd mmm yyyy |     Who      | Description of changes
// =====|=============|==============|======================================
//  1.0 | 12 Mar 2002 | LH           | PG Release.
//      |             |              |
//###########################################################################
#include "F2812_Boot.h"
// Private functions
inline void SCIA_GPIOSelect(void);
inline void SCIA_Init(void);
inline void SCIA_AutobaudLock(void);
inline ui16 SCIA_CheckKeyVal(void);
inline void SCIA_ReservedFn(void);
inline void SCIA_SysClockEnable(void);
ui32 SCIA_GetLongData(void);
ui16 SCIA_GetWordData(void);
void SCIA_CopyData(void);
// Data section where SCIA control registers
// reside
#pragma DATA_SECTION(SCIARegs,".SCIARegs");
volatile struct SCI_REGS SCIARegs;
//###########################################################
// ui32 SCI_Boot(void)
//-------------------------------------------
// This module is the main SCI boot routine.
// It will load code via the SCI-A port.
//
// It will return a entry point address back
// to the InitBoot routine which in turn calls
// the ExitBoot routine.
//-------------------------------------------
ui32 SCI_Boot()
{
   ui32 EntryAddr;
   ui16 ErrorFlag;
   SCIA_SysClockEnable();
   SCIA_GPIOSelect();
   SCIA_Init();
   SCIA_AutobaudLock();
```

```
   // If the KeyValue was invalid, abort the load
   // and return the flash entry point.
   ErrorFlag = SCIA_CheckKeyVal();
   if (ErrorFlag == ERROR) return FLASH_ENTRY_POINT;
   SCIA_ReservedFn();
   EntryAddr = SCIA_GetLongData();
   SCIA_CopyData();
   return EntryAddr;
}
//###########################################################
// void SCIA_GPIOSelect(void)
//-----------------------------------------
// Enable pins for the SCI-A module for SCI
// peripheral functionality.
//-----------------------------------------
inline void SCIA_GPIOSelect()
{
    EALLOW
    GPIOMuxRegs.GPFMUX.all = 0x0030;
    EDIS
}
//###########################################################
// void SCIA_Init(void)
//-------------------------------------------
// Initialize the SCI-A port for communications
// with the host.
//-------------------------------------------
inline void SCIA_Init()
{
    // Enable FIFO reset bit only
    SCIARegs.SCIFFTX.all=0x8000;
    // 1 stop bit, No parity, 8-bit character
    // No loopback
    SCIARegs.SCICCR.all = 0x0007;
    // Enable TX, RX, Use internal SCICLK
    SCIARegs.SCICTL1.all = 0x0003;
    // Disable RxErr, Sleep, TX Wake,
    // Diable Rx Interrupt, Tx Interrupt
    SCIARegs.SCICTL2.all = 0x0000;
    // Relinquish SCI-A from reset
    SCIARegs.SCICTL1.all = 0x0023;
    return;
}
//###########################################################
// void SCIA_AutobaudLock(void)
//-----------------------------------------------
// Perform autobaud lock with the host.
// Note that if autobaud never occurs
// the program will hang in this routine as there
// is no timeout mechanism included.
//-----------------------------------------------
inline void SCIA_AutobaudLock()
{
```

```
    ui16 byteData;
    // Must prime baud register with >= 1
    SCIARegs.SCILBAUD = 1;
    // Prepare for autobaud detection
    // Set the CDC bit to enable autobaud detection
    // and clear the ABD bit
    SCIARegs.SCIFFCT.all = 0x2000;
    // Wait until we correctly read an
    // 'A' or 'a' and lock
    while(SCIARegs.SCIFFCT.bit.ABD != 1) {}
    // After autobaud lock, clear the CDC bit
    SCIARegs.SCIFFCT.bit.CDC = 0;
    while(SCIARegs.SCIRXST.bit.RXRDY != 1) { }
    byteData = SCIARegs.SCIRXBUF.bit.RXDT;
    SCIARegs.SCITXBUF = byteData;
    return;
}
//###################################################
// ui16 SCIA_CheckKeyVal(void)
//---------------------------------------------------
// The header of the datafile should have a proper
// key value of 0x08 0xAA.  If it does not, then
// we either have a bad data file or we are not
// booting properly.  If this is the case, return
// an error to the main routine.
//---------------------------------------------------
inline ui16 SCIA_CheckKeyVal()
{
    ui16 wordData;
    wordData = SCIA_GetWordData();
    if(wordData != EIGHT_BIT_HEADER) return ERROR;
    // No error found
    return NO_ERROR;
}
//###################################################
// void SCIA_ReservedFn(void)
//---------------------------------------------------
// This function reads 8 reserved words in the header.
// None of these reserved words are used by the
// SCI boot loader at this time, they may be used in
// future devices for enhancments.
//---------------------------------------------------
inline void SCIA_ReservedFn()
{
    ui16 i;
    // Read and discard the 8 reserved words.
    for(i = 1; i <= 8; i++)
    {
       SCIA_GetWordData();
    }
    return;
}
//###################################################
```

```
// ui32 SCIA_GetLongData(void)
//--------------------------------------------
// This routine fetches two words from the SCI-A
// port and puts them together to form a single
// 32-bit value.  It is assumed that the host is
// sending the data in the form MSW:LSW.
//--------------------------------------------
ui32 SCIA_GetLongData()
{
    ui32 longData = (ui32)0x00000000;
    // Fetch the upper 1/2 of the 32-bit value
    longData = ( (ui32)SCIA_GetWordData() << 16);
    // Fetch the lower 1/2 of the 32-bit value
    longData |= (ui32)SCIA_GetWordData();
    return longData;
}
//################################################
// ui16 SCIA_GetWordData(void)
//--------------------------------------------
// This routine fetches two bytes from the SCI-A
// port and puts them together to form a single
// 16-bit value.  It is assumed that the host is
// sending the data in the order LSB followed by MSB.
//--------------------------------------------
ui16 SCIA_GetWordData()
{
   ui16 wordData;
   ui16 byteData;

   wordData = 0x0000;
   byteData = 0x0000;

   // Fetch the LSB and verify back to the host
   while(SCIARegs.SCIRXST.bit.RXRDY != 1) { }
   wordData =  (ui16)SCIARegs.SCIRXBUF.bit.RXDT;
   SCIARegs.SCITXBUF = wordData;
   // Fetch the MSB and verify back to the host
   while(SCIARegs.SCIRXST.bit.RXRDY != 1) { }
   byteData =  (ui16)SCIARegs.SCIRXBUF.bit.RXDT;
   SCIARegs.SCITXBUF = byteData;

   // form the wordData from the MSB:LSB
   wordData |= (byteData << 8);
   return wordData;
}
//################################################
// void SCIA_CopyData(void)
//--------------------------------------------------
// This routine copies multiple blocks of data from the host
// to the specified RAM locations.  There is no error
// checking on any of the destination addresses.
// That is it is assumed all addresses and block size
// values are correct.
```

```
//
// Multiple blocks of data are copied until a block
// size of 00 00 is encountered.
//
//-----------------------------------------------------
void SCIA_CopyData()
{
   struct HEADER {
     ui16 BlockSize;
     ui32 DestAddr;
   } BlockHeader;

   ui16 wordData;
   ui16 i;

   // Get the size in words of the first block
   BlockHeader.BlockSize = SCIA_GetWordData();

   // While the block size is > 0 copy the data
   // to the DestAddr.  There is no error checking
   // as it is assumed the DestAddr is a valid
   // memory location

   while(BlockHeader.BlockSize != (ui16)0x0000)
   {
      BlockHeader.DestAddr = SCIA_GetLongData();
      for(i = 1; i <= BlockHeader.BlockSize; i++)
      {
          wordData = SCIA_GetWordData();
          *(ui16 *)BlockHeader.DestAddr++ = wordData;
      }

      // Get the size of the next block
      BlockHeader.BlockSize = SCIA_GetWordData();
   }
   return;
}
//###################################################
// inline void SCIA_SysClockEnable(void)
//-----------------------------------------------------
// This routine enables the clocks to the SCIA Port.
//-----------------------------------------------------
inline void SCIA_SysClockEnable()
{
    EALLOW;
    SysCtrlRegs.PCLKCR.bit.SCIAENCLK=1;
    SysCtrlRegs.LOSPCP.all = 0x0002;
    EDIS;
}
// EOF-------

//===========================================================================
//
```

```
// FILE:    Parallel_Boot.c
//
// TITLE:   F2810/12 Parallel Port I/O boot routines
//
// Functions:
//
//       ui32 Parallel_Boot(void)
//       inline void Parallel_GPIOSelect(void)
//       inline ui16 Parallel_CheckKeyVal(void)
//       invline void Parallel_ReservedFn(void)
//       ui32 Parallel_GetLongData(ui16 DataSize)
//       ui16 Parallel_GetWordData(ui16 DataSize)
//       void Parallel_CopyData(ui16 DataSize)
//       void Parallel_WaitHostRdy(void)
//       void Parallel_HostHandshake(void)
// Notes:
//
//############################################################################
//
//  Ver | dd mmm yyyy |     Who      | Description of changes
// =====|=============|==============|======================================
//  1.0 | 12 Mar 2002 | LH           | PG Release.
//      |             |              |
//############################################################################
#include "F2812_Boot.h"
// Private function definitions
inline void Parallel_GPIOSelect(void);
inline ui16 Parallel_CheckKeyVal(void);
inline void Parallel_ReservedFn();
ui32 Parallel_GetLongData(ui16 DataSize);
ui16 Parallel_GetWordData(ui16 DataSize);
void Parallel_CopyData(ui16 DataSize);
void Parallel_WaitHostRdy(void);
void Parallel_HostHandshake(void);
#define HOST_DATA_NOT_RDY  GPIODataRegs.GPDDAT.bit.GPIOD5!=0
#define WAIT_HOST_ACK      GPIODataRegs.GPDDAT.bit.GPIOD5!=1
// Set (DSP_ACK) or Clear (DSP_RDY) GPIOD6
#define DSP_ACK            GPIODataRegs.GPDSET.all   = 0x0040
#define DSP_RDY            GPIODataRegs.GPDCLEAR.all = 0x0040
#define DATA               GPIODataRegs.GPBDAT.all
// Data section where GPIO control and data registers
// reside
#pragma DATA_SECTION(GPIODataRegs,".GPIODataRegs");
volatile struct GPIO_DATA_REGS GPIODataRegs;
#pragma DATA_SECTION(GPIOMuxRegs,".GPIOMuxRegs");
volatile struct GPIO_MUX_REGS GPIOMuxRegs;
#endif

//###################################################
// ui32 Parallel_Boot(void)
//---------------------------------------
// This module is the main Parallel boot routine.
// It will load code via GP I/O port B.
```

```
//
// This boot mode accepts 8-bit or 16-bit data.
// 8-bit data is expected to be the order LSB
// followed by MSB.
//
// This function returns a entry point address back
// to the InitBoot routine which in turn calls
// the ExitBoot routine.
//-----------------------------------------
ui32 Parallel_Boot()
{
   ui32 EntryAddr;
   ui16 DataSize;

   Parallel_GPIOSelect();
   DataSize = Parallel_CheckKeyVal();
   if (DataSize == ERROR) return FLASH_ENTRY_POINT;
   Parallel_ReservedFn(DataSize);

   EntryAddr = Parallel_GetLongData(DataSize);
   Parallel_CopyData(DataSize);

   return EntryAddr;
}
//#################################################
// void Parallel_GPIOSelect(void)
//-----------------------------------------
// Enable pins for GP I/O on Port B.  Also enable
// the control pins for host ack and DSP ready.
//-----------------------------------------
inline void Parallel_GPIOSelect()
{
    EALLOW;

    // GPIO Port B is all I/O pins
    // 0 = I/O pin  1 = Peripheral pin
     GPIOMuxRegs.GPBMUX.all = 0x0000;

    // GPIO Port D pin 5 and 6 are I/O pins
    GPIOMuxRegs.GPDMUX.all &= 0xFF9F;

    // Port B is all input
    // D5 is an input control from the Host Ack/Rdy
    // D6 is an output for DSP Ack/Rdy
    // 0 = input   1 = output
    GPIOMuxRegs.GPDDIR.bit.GPIOD6 = 1;
    GPIOMuxRegs.GPDDIR.bit.GPIOD5 = 0;
    GPIOMuxRegs.GPBDIR.all = 0x0000;

    EDIS;
}
//#################################################
// void Parallel_CheckKeyVal(void)
```

```
//---------------------------------------
// Determine if the data we are loading is in
// 8-bit or 16-bit format.
// If neither, return an error.
//
// Note that if the host never responds then
// the code will be stuck here.  That is there
// is no timeout mechanism.
//---------------------------------------
inline ui16 Parallel_CheckKeyVal()
{
   ui16 wordData;

   // Fetch a word from the parallel port and compare
   // it to the defined 16-bit header format, if not check
   // for a 8-bit header format.

   wordData = Parallel_GetWordData(SIXTEEN_BIT);
   if(wordData == SIXTEEN_BIT_HEADER) return SIXTEEN_BIT;
   // If not 16-bit mode, check for 8-bit mode
   // Call Parallel_GetWordData with 16-bit mode
   // so we only fetch the MSB of the KeyValue and not
   // two bytes. We will ignore the upper 8-bits and combine
   // the result with the previous byte to form the
   // header KeyValue.

   wordData = wordData & 0x00FF;
   wordData |= Parallel_GetWordData(SIXTEEN_BIT) << 8;
   if(wordData == EIGHT_BIT_HEADER) return EIGHT_BIT;
   // Didn't find a 16-bit or an 8-bit KeyVal header so return an error.
   else return ERROR;
}
//#################################################
// void Parallel_ReservedFn(void)
//-------------------------------------------------
// This function reads 8 reserved words in the header.
// None of these reserved words are used by the
// Parallel boot loader at this time, they may be used in
// future devices for enhancments.
//-------------------------------------------------
inline void Parallel_ReservedFn(ui16 DataSize)
{
    ui16 i;
    // Read and discard the 8 reserved words.
    for(i = 1; i <= 8; i++)
    {
       Parallel_GetWordData(DataSize);
    }
    return;
}
//#################################################
// void Parallel_CopyData(void)
//-------------------------------------------------
```

```
// This routine copies multiple blocks of data from the host
// to the specified RAM locations. There is no error
// checking on any of the destination addresses.
// That is it is assumed all addresses and block size
// values are correct.
//
// Multiple blocks of data are copied until a block
// size of 00 00 is encountered.
//
//-----------------------------------------------------
void Parallel_CopyData(ui16 DataSize)
{
   struct HEADER {
     ui16 BlockSize;
     ui32 DestAddr;
   } BlockHeader;

   ui16 wordData;
   ui16 i;

   // Get the size in words of the first block
   BlockHeader.BlockSize = Parallel_GetWordData(DataSize);

   // While the block size is > 0 copy the data
   // to the DestAddr.  There is no error checking
   // as it is assumed the DestAddr is a valid
   // memory location

   while(BlockHeader.BlockSize != (ui16)0x0000)
   {
      BlockHeader.DestAddr = Parallel_GetLongData(DataSize);
      for(i = 1; i <= BlockHeader.BlockSize; i++)
      {
          wordData = Parallel_GetWordData(DataSize);
          *(ui16 *)BlockHeader.DestAddr++ = wordData;
      }

      // Get the size of the next block
      BlockHeader.BlockSize = Parallel_GetWordData(DataSize);
   }
   return;
}
//###################################################
// ui16 Parallel_GetWordData(ui16 DataSize)
//-----------------------------------------------
// This routine fetches a 16-bit word from the
// GP I/O port. The function is passed a DataSize
// value. If the DataSize is 16, then the input
// stream is 16-bits and the function fetches a
// signle word and returns it to the host.
//
// If the DataSize is 8, then the input stream is
// an 8-bit input stream and the upper 8-bits of the
```

```
// GP I/O port are ignored. In the 8-bit case the
// first fetches the LSB and then the MSB from the
// GPIO port.  These two bytes are then put together to
// form a single 16-bit word that is then passed back
// to the host. Note that in this case, the input stream
// from the host is in the order LSB followed by MSB
//-------------------------------------------------
ui16 Parallel_GetWordData(ui16 DataSize)
{
   ui16 wordData;

   // Get a word of data.  If we are in
   // 16-bit mode then we are done.

   Parallel_WaitHostRdy();
   wordData = DATA;
   Parallel_HostHandshake();
   // If we are in 8-bit mode then the first
   // fetch was only the LSB. Fetch the MSB.

   if(DataSize == EIGHT_BIT) {
      wordData = wordData & 0x00FF;
      Parallel_WaitHostRdy();
      wordData |= (DATA << 8);
      Parallel_HostHandshake();
   }
   return wordData;
}
//#################################################
// ui32 Parallel_GetLongData(ui16 DataSize)
//-------------------------------------------------
// This routine fetches two words from the GP I/O
// port and puts them together to form a single
// 32-bit value.  It is assumed that the host is
// sending the data in the form MSW:LSW.
//-------------------------------------------------
ui32 Parallel_GetLongData(ui16 DataSize)
{
   ui32 longData;
   longData = ( (ui32)Parallel_GetWordData(DataSize) )<< 16;
   longData |=  (ui32)Parallel_GetWordData(DataSize);
   return longData;
}
//#################################################
// void Parallel_WaitHostRdy(void)
//-------------------------------------------------
// This routine tells the host that the DSP is ready to
// recieve data. The DSP then waits for the host to
// signal that data is ready on the GP I/O port.e
//-------------------------------------------------
void Parallel_WaitHostRdy()
{
   DSP_RDY;
```

```
   while(HOST_DATA_NOT_RDY) { }
}
//#################################################
// void Parallel_HostHandshake(void)
//-------------------------------------------------
// This routine tells the host that the DSP has recieved
// the data. The DSP then waits for the host to acknowledge
// the reciept before continuing.
//-------------------------------------------------
void Parallel_HostHandshake()
{
   DSP_ACK;
   while(WAIT_HOST_ACK) { }
}
// EOF --------
//###########################################################################
//
// FILE:    SPI_Boot.c
//
// TITLE:   F2810/12 SPI Boot mode routines
//
// Functions:
//
//     ui32 SPI_Boot(void)
//     inline void SPIA_GPIOSelect(void)
//     inline void SPIA_SysClockEnable(void)
//     inline void SPIA_Init(void)
//     inline void SPIA_Transmit(u16 cmdData)
//     inline ui16 SPIA_CheckKeyVal(void)
//     inline void SPIA_ReservedFn(void);
//     ui32 SPIA_GetLongData(void)
//     ui32 SPIA_GetWordData(void)
//     void SPIA_CopyData(void)
//
// Notes:
//
//###########################################################################
//
//  Ver | dd mmm yyyy |     Who      | Description of changes
// =====|=============|==============|======================================
//  1.0 | 12 Mar 2002 | SS           | PG Release.
//      |             |              |
//###########################################################################
#include "F2812_Boot.h"
#pragma DATA_SECTION(SPIARegs,".SPIARegs");
volatile struct SPI_REGS SPIARegs;
// Private functions
inline void SPIA_GPIOSelect(void);
inline void SPIA_Init(void);
inline ui16 SPIA_Transmit(ui16 cmdData);
inline ui16 SPIA_CheckKeyVal(void);
inline void SPIA_ReservedFn(void);
inline void SPIA_SysClockEnable(void);
```

```
ui32 SPIA_GetLongData(void);
ui16 SPIA_GetWordData(void);
void SPIA_CopyData(void);
//#################################################
// ui32 SPI_Boot(void)
//-----------------------------------------
// This module is the main SPI boot routine.
// It will load code via the SPI-A port.
//
// It will return a entry point address back
// to the ExitBoot routine.
//-----------------------------------------
ui32 SPI_Boot()
{
   ui32 EntryAddr;
   ui16 ErrorFlag;

   SPIA_SysClockEnable();
   SPIA_GPIOSelect();
   SPIA_Init();
   // 1. Enable EEPROM chip enable - low - Bit 5 for the IOPORT
   // Chip enable - high
    GPIODataRegs.GPFCLEAR.bit.GPIOF3 =1;
   // 2. Enable EEPROM and send EEPROM Read Command
   SPIA_Transmit(0x0300);
   // 3. Send Starting for the EEPROM address 16bit
   // Sending 0x0000,0000 will work for address and data packets
   SPIA_GetWordData();
   // 4. Check for 0x08AA data header, else go to flash
   ErrorFlag = SPIA_CheckKeyVal();
   if (ErrorFlag != 0) return FLASH_ENTRY_POINT;
   // 4a.Check for Clock speed change and reserved words
   SPIA_ReservedFn();
   // 5. Get point of Entry address after load
   EntryAddr = SPIA_GetLongData();
   // 6. Receive and copy one or more code sections to  destination addresses
   SPIA_CopyData();
   // 7. Disable EEPROM chip enable - high
   // Chip enable - high
    GPIODataRegs.GPFSET.bit.GPIOF3 =1;
   return EntryAddr;
}
//#################################################
// void SPIA_GPIOSelect(void)
//-----------------------------------------
// Enable pins for the SPI-A module for SPI
// peripheral functionality.
//-----------------------------------------
inline void SPIA_GPIOSelect()
{
  EALLOW
    // Enable SPISIMO/SPISOMI/SPICLK pins
    GPIOMuxRegs.GPFMUX.all = 0x0007;
```

```
    //IOPORT as output pin instead of SPISTE
    GPIOMuxRegs.GPFDIR.all = 0x0008;
    //Set IOPORT high GPIOF3, EEPROM CS enable high
    GPIODataRegs.GPFDAT.all = 0x0008;

    EDIS
}
//###################################################
// void SPIA_Init(void)
//--------------------------------------------
// Initialize the SPI-A port for communications
// with the host.
//--------------------------------------------
inline void SPIA_Init()
{
    // Enable FIFO reset bit only
    SPIARegs.SPIFFTX.all=0x8000;
    // 8-bit character
    SPIARegs.SPICCR.all = 0x0007;
    // Use internal SPICLK master mode and Talk mode
    SPIARegs.SPICTL.all = 0x000E;
    // Use the slowest baud rate
    SPIARegs.SPIBRR     = 0x007f;
    // Relinquish SPI-A from reset
    SPIARegs.SPICCR.all = 0x0087;
    return;
}
//###################################################
// void SPIA_Transmit(void)
//--------------------------------------------
// Send a byte/words through SPI transmit channel
//--------------------------------------------
inline ui16 SPIA_Transmit(ui16 cmdData)
{
    ui16 recvData;
    // Send Read command/dummy word to EEPROM to fetch a byte
    SPIARegs.SPITXBUF = cmdData;
    while( (SPIARegs.SPISTS.bit.INT_FLAG) !=1);
    // Clear SPIINT flag and capture received byte
    recvData = SPIARegs.SPIRXBUF;

    return recvData;

}
//###################################################
// ui16 SPIA_CheckKeyVal(void)
//--------------------------------------------
// The header of the datafile should have a proper
// key value of 0x08 0xAA. If it does not, then
// we either have a bad data file or we are not
// booting properly. If this is the case, return
// an error to the main routine.
//--------------------------------------------
```

```
inline ui16 SPIA_CheckKeyVal()
{
    ui16 wordData;

    wordData = SPIA_GetWordData();
    if(wordData != 0x08AA) return ERROR;
    // No error found
    return NO_ERROR;
}
//#################################################
// void SPIA_ReservedFn(void)
//-------------------------------------------------
// This function reads 8 reserved words in the header.
// The first word has parameters for LOSPCP
// and SPIBRR register 0xMSB:LSB, LSB = is a three
// bit field for LOSPCP change MSB = is a 6bit field
// for SPIBRR register update
//
// If either byte is the default value of the register
// then no speed change occurs. The default values
// are LOSPCP = 0x02 and SPIBRR = 0x7F
// The remaining reserved words are read and discarded
// and then returns to the main routine.
//-------------------------------------------------
inline void SPIA_ReservedFn()
{
    ui16 speedData;
    ui16 i;

    speedData = (ui16)0x0000;
    // update LOSPCP register if 1st reserved byte is not the default
    // value of 0x0002
    speedData = SPIA_Transmit((ui16)0x0000);
    if(speedData != (ui16)0x0002)
    {
        EALLOW;
        SysCtrlRegs.LOSPCP.all = speedData;
        EDIS;
       // Dummy cycles
       asm(" RPT #0x0F ||NOP");
    }

    // update SPIBRR register if 2nd reserved byte is not the default
    // value of 0x7F
    speedData = SPIA_Transmit((ui16)0x0000);

    if(speedData != (ui16) 0x007F)
    {
       SPIARegs.SPIBRR  = speedData;
       // Dummy cycles
       asm(" RPT #0x0F ||NOP");
    }
    // Read and discard the next 7 reserved words.
```

```
    for(i = 1; i <= 7; i++)
    {
       SPIA_GetWordData();
    }
    return;
}
//###########################################################
// ui32 SPIA_GetLongData(void)
//-----------------------------------------------
// This routine fetches two words from the SPI-A
// port and puts them together to form a single
// 32-bit value. It is assumed that the host is
// sending the data in the form MSW:LSW.
// ----------------------------------------------
ui32 SPIA_GetLongData()
{
    ui32 longData = (ui32)0x00000000;
    // Fetch the upper 1/2 of the 32-bit value
    longData = ( (ui32)SPIA_GetWordData() << 16);

    // Fetch the lower 1/2 of the 32-bit value
    longData |= (ui32)SPIA_GetWordData();
    return longData;
}
//###########################################################
// ui16 SPIA_GetWordData(void)
//-----------------------------------------------
// This routine fetches two bytes from the SPI-A
// port and puts them together to form a single
// 16-bit value.  It is assumed that the host is
// sending the data in the form MSB:LSB.
//-----------------------------------------------
ui16 SPIA_GetWordData()
{
   ui16 wordData;
   ui16 byteData;

   wordData = 0x0000;

   // Fetch the LSB and verify back to the host
   wordData =  SPIA_Transmit(0x0000);
   // Fetch the MSB and verify back to the host
   byteData =  SPIA_Transmit(0x0000);
   // Shift the upper byte to be the MSB
    wordData |= (byteData << 8);
   return wordData;
}
//###########################################################
// void SPIA_CopyData(void)
//-----------------------------------------------------
// This routine copies multiple blocks of data from the host
// to the specified RAM locations. There is no error
// checking on any of the destination addresses.
```

```
// That is it is assumed all addresses and block size
// values are correct.
//
// Multiple blocks of data are copied until a block
// size of 00 00 is encountered.
//
//-----------------------------------------------------
void SPIA_CopyData()
{
   struct HEADER {
     ui16 BlockSize;
     ui32 DestAddr;
   } BlockHeader;

   ui16 wordData;
   ui16 i;

   // Get the size in words of the first block
   BlockHeader.BlockSize = SPIA_GetWordData();

   // While the block size is > 0 copy the data
   // to the DestAddr. There is no error checking
   // as it is assumed the DestAddr is a valid
   // memory location

   while(BlockHeader.BlockSize != (ui16)0x0000)
   {
      BlockHeader.DestAddr = SPIA_GetLongData();
      for(i = 1; i <= BlockHeader.BlockSize; i++)
      {
          wordData = SPIA_GetWordData();
          *(ui16 *)BlockHeader.DestAddr++ = wordData;
      }

      // Get the size of the next block
      BlockHeader.BlockSize = SPIA_GetWordData();
   }
   return;
}
//###################################################
// inline void SPIA_SysClockEnable(void)
//-----------------------------------------------------
// This routine enables the clocks to the SPIA Port.
//-----------------------------------------------------
inline void SPIA_SysClockEnable()
{
    EALLOW;
    SysCtrlRegs.PCLKCR.bit.SPIAENCLK=1;
    SysCtrlRegs.LOSPCP.all = 0x0002;
    EDIS;
}
```

Appendix A

Revision History

This document was revised to SPRU095B from SPR095A. The scope of the revisions was limited to technical changes as described in Section A.1. This appendix lists only revisions made in the most recent version.

A.1 Changes Made in This Revision

The following changes were made in this revision:

Global change − Name changed to reflect new devices.

Page	Additions/Modifications/Deletions
7	Added a paragraph to explain the title change and new devices
18	Changed title of Table 4 from GPIO Pin Status to Boot Mode Selection
19	Added paragraph to “Jump to branch instruction in Flash Memory:” bullet under Figure 4
20	Added paragraph to “Jump to OTP Memory” bullet under Figure 6
31	Modified paragraph under Table 7
44	Modified step 5 in the description of the sequence of data transfer by switching least and most significant byte
51	Added to the description of −e value in Table 10

TMS320F20x/F24x DSP Embedded Flash Memory Technical Reference

This document contains preliminary data current as of publication date and is subject to change without notice.

Literature Number: SPRU282
September 1998

IMPORTANT NOTICE

Texas Instruments and its subsidiaries (TI) reserve the right to make changes to their products or to discontinue any product or service without notice, and advise customers to obtain the latest version of relevant information to verify, before placing orders, that information being relied on is current and complete. All products are sold subject to the terms and conditions of sale supplied at the time of order acknowledgement, including those pertaining to warranty, patent infringement, and limitation of liability.

TI warrants performance of its semiconductor products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

CERTAIN APPLICATIONS USING SEMICONDUCTOR PRODUCTS MAY INVOLVE POTENTIAL RISKS OF DEATH, PERSONAL INJURY, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE ("CRITICAL APPLICATIONS"). TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS. INCLUSION OF TI PRODUCTS IN SUCH APPLICATIONS IS UNDERSTOOD TO BE FULLY AT THE CUSTOMER'S RISK.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards must be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance or customer product design. TI does not warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used. TI's publication of information regarding any third party's products or services does not constitute TI's approval, warranty or endorsement thereof.

Read This First

About This Manual

This reference guide describes the operation of the embedded flash EEPROM module on the TMS320F20x/F24x digital signal processor (DSP) devices and provides sample code that you can use in developing your own software. The performance specifications of the embedded flash memory have been evaluated using the algorithms and techniques described in this guide. TI does not recommend deviation from these algorithms and techniques, since doing so could affect device performance. The book does not describe the use of any specific flash programming tool nor does it describe the external interface to the DSP. For information about any aspect of the TMS320F20x/F24x devices other than the embedded flash EEPROM module, see *Related Documentation from Texas Instruments* on page v.

How to Use This Manual

There are several stand-alone flash programming tools for TMS320F20x/F24x generation of DSPs. Using one of these stand-alone tools with the TMS320F20x/F24x requires only a basic understanding of the flash operations. More information about these flash programming tools is available on the TI web page, http://www.ti.com. This guide is intended to provide a complete understanding of the flash operations. This level of understanding is necessary for making modifications to existing flash programming tools or for developing alternative programming schemes.

If you are looking for information about:	Turn to these locations:
Algorithms	Chapter 3, *Algorithm Implementations and Software Considerations*
Erasing the flash array	Section 1.1, *Basic Concepts of Flash Memory Technology* Section 2.1, *Modifying the Contents of the TMS320F20x/F24x Flash Array* Section 2.6, *Erase Operation* Section 3.3, *Erase Algorithm*

If you are looking for information about:	Turn to these locations:
Over-erasure (depletion) and recovery	Section 1.1, *Basic Concepts of Flash Memory Technology* Section 2.7, *Recovering From Over-Erasure (Flash-Write Operation)* Section 3.4, *Flash-Write Algorithm*
Programming the flash array	Section 1.1, *Basic Concepts of Flash Memory Technology* Section 2.1, *Modifying the Contents of the TMS320F20x/F24x Flash Array* Section 2.5, *Program Operation* Section 3.2, *Programming Algorithm*
Sample code	Appendix A, *Assembly Source Listings and Program Examples*

Notational Conventions

This document uses the following conventions.

- The flash EEPROM is referred to as flash memory or the flash module. The term flash array refers to the actual memory array within the flash module. The flash module includes the flash memory array and the associated control circuitry.
- The DSP generation and devices are abbreviated as follows:
  - TMS320F20x/24x generation: ’F20x/24x
  - TMS320F20x devices: ’F20x
  - TMS320F24x devices: ’F24x
- Program listings and code examples are shown in a `special typeface`.

  Here is a sample program listing:

```
0011  0005  0001          .field    1, 2
0012  0005  0003          .field    3, 4
0013  0005  0006          .field    6, 3
0014  0006                .even
```

Related Documentation From Texas Instruments

The following books describe the ’F20x/24x and related support tools. To obtain a copy of any of these TI documents, call the Texas Instruments Literature Response Center at (800) 477–8924. When ordering, please identify the book by its title and literature number.

TMS320C24x DSP Controllers Reference Set, Volume 1: CPU, System, and Instruction Set (literature number SPRU160) describes the TMS320C24x 16-bit, fixed-point, digital signal processor controller. Covered are its architecture, internal register structure, data and program addressing, and instruction set. Also includes instruction set comparisons and design considerations for using the XDS510 emulator.

TMS320C24x DSP Controllers Reference Set Volume 2: Peripheral Library and Specific Devices (literature number SPRU161) describes the peripherals available on the TMS320C24x digital signal processor controllers and their operation. Also described are specific device configurations of the ’C24x family.

TMS320C240, TMS320F240 DSP Controllers (literature number SPRS042) data sheet contains the electrical and timing specifications for these devices, as well as signal descriptions and pinouts for all of the available packages.

TMS320C2x/C2xx/C5x Optimizing C Compiler User’s Guide (literature number SPRU024) describes the ’C2x/C2xx/C5x C compiler. This C compiler accepts ANSI standard C source code and produces TMS320 assembly language source code for the ’C2x, ’C2xx, and ’C5x generations of devices.

TMS320F206 Digital Signal Processor (literature number SPRS050) data sheet contains the electrical and timing specifications for the ’F206 device, as well as signal descriptions and the pinout.

TMS320F241, TMS320C241, TMS320C242 DSP Controllers (literature number SPRS063) data sheet contains the electrical and timing specifications for the ’F241, ’C241, and ’C242 devices, as well as signal descriptions and pinouts.

TMS320F243 DSP Controller (literature number SPRS064) data sheet contains the electrical and timing specifications for the ’F243 device, as well as signal descriptions and the pinout.

TMS320C2xx User’s Guide (literature number SPRU127) discusses the hardware aspects of the ’C2xx 16-bit, fixed-point digital signal processors. It describes the architecture, the instruction set, and the on-chip peripherals.

TMS320C2xx C Source Debugger User's Guide (literature number SPRU151) tells you how to invoke the 'C2xx emulator and simulator versions of the C source debugger interface. This book discusses various aspects of the debugger interface, including window management, command entry, code execution, data management, and breakpoints. It also includes a tutorial that introduces basic debugger functionality.

If You Need Assistance . . .

❑ World-Wide Web Sites

TI Online	http://www.ti.com
Semiconductor Product Information Center (PIC)	http://www.ti.com/sc/docs/pic/home.htm
DSP Solutions	http://www.ti.com/dsps
320 Hotline On-line™	http://www.ti.com/sc/docs/dsps/support.htm

❑ North America, South America, Central America

Product Information Center (PIC)	(972) 644-5580		
TI Literature Response Center U.S.A.	(800) 477-8924		
Software Registration/Upgrades	(214) 638-0333	Fax: (214) 638-7742	
U.S.A. Factory Repair/Hardware Upgrades	(281) 274-2285		
U.S. Technical Training Organization	(972) 644-5580		
DSP Hotline	(281) 274-2320	Fax: (281) 274-2324	Email: dsph@ti.com
DSP Modem BBS	(281) 274-2323		

DSP Internet BBS via anonymous ftp to ftp://ftp.ti.com/pub/tms320bbs

❑ Europe, Middle East, Africa

European Product Information Center (EPIC) Hotlines:

Multi-Language Support	+33 1 30 70 11 69	Fax: +33 1 30 70 10 32

Email: epic@ti.com

Deutsch	+49 8161 80 33 11 or	+33 1 30 70 11 68
English		+33 1 30 70 11 65
Francais		+33 1 30 70 11 64
Italiano		+33 1 30 70 11 67
EPIC Modem BBS		+33 1 30 70 11 99
European Factory Repair		+33 4 93 22 25 40
Europe Customer Training Helpline		Fax: +49 81 61 80 40 10

❑ Asia-Pacific

Literature Response Center	+852 2 956 7288	Fax: +852 2 956 2200
Hong Kong DSP Hotline	+852 2 956 7268	Fax: +852 2 956 1002
Korea DSP Hotline	+82 2 551 2804	Fax: +82 2 551 2828
Korea DSP Modem BBS	+82 2 551 2914	
Singapore DSP Hotline		Fax: +65 390 7179
Taiwan DSP Hotline	+886 2 377 1450	Fax: +886 2 377 2718
Taiwan DSP Modem BBS	+886 2 376 2592	

Taiwan DSP Internet BBS via anonymous ftp to ftp://dsp.ee.tit.edu.tw/pub/TI/

❑ Japan

Product Information Center	+0120-81-0026 (in Japan)	Fax: +0120-81-0036 (in Japan)
	+03-3457-0972 or (INTL) 813-3457-0972	Fax: +03-3457-1259 or (INTL) 813-3457-1259
DSP Hotline	+03-3769-8735 or (INTL) 813-3769-8735	Fax: +03-3457-7071 or (INTL) 813-3457-7071
DSP BBS via Nifty-Serve	Type “Go TIASP”	

❑ Documentation

When making suggestions or reporting errors in documentation, please include the following information that is on the title page: the full title of the book, the publication date, and the literature number.

Mail: Texas Instruments Incorporated
Technical Documentation Services, MS 702
P.O. Box 1443
Houston, Texas 77251-1443

Email: dsph@ti.com

Note: When calling a Literature Response Center to order documentation, please specify the literature number of the book.

Contents

Figures

Tables

Chapter 1

Introduction

The TMS320F20x/F24x digital signal processors (DSPs) contain on-chip flash EEPROM (electrically-erasable programmable read-only memory). The embedded flash memory provides an attractive alternative to masked program ROM. Like ROM, flash memory is nonvolatile, but it has an advantage over ROM: *in-system* reprogrammability.

This chapter discusses basic flash memory technology, introduces the flash memory module of the ’F20x/F24x DSP, and lists the benefits of flash memory embedded in a DSP chip.

Topic **Page**

1.1 Basic Concepts of Flash Memory Technology

The term flash in this EEPROM technology refers to the speed of some of the operations performed on the memory (these operations will be described in greater detail later in this document). An entire block of bits is affected simultaneously in a *block* or *flash operation*, rather than being affected one bit at a time. In contrast, writing data to the flash memory cannot be a block operation, since normally a selection of ones and zeroes are written (all bits are not the same value). Writing selected bits to create a desired pattern is known as programming the flash memory, and a written bit is called a programmed bit.

Several different types of program and erase operations are performed on the flash memory in order to properly produce the desired pattern of ones and zeroes in the memory. It should be noted that, under some conditions, flash memory may become overerased, resulting in a condition known as depletion. The ’F20x/F24x algorithms avoid overerasure by using an approach that erases in small increments until complete erasure is achieved.

The ’F20x/F24x flash EEPROM includes a special operation, flash-write, that is used only to recover from over-erasure. Because of the implementation of the flash memory, when over-erasure occurs, any particular bit in depletion mode is difficult to identify. For this reason, the ’F20x/F24x simply writes an entire block of bits simultaneously; hence, the name flash-write.

The program and erase operations in flash memory must provide sufficient charge margin on 1s and 0s to ensure data retention, so the ’F20x/F24x flash module includes a hardware mechanism that provides margin for erasing or programming. This mechanism implements voltage reference levels which ensure this logic level margin when modifying the contents of the flash memory.

1.2 TMS320F20x/F24x Flash Module

The ’F20x/F24x flash EEPROM is implemented with one or two independent flash memory modules of 8K or 16K words. Each flash module is composed of a flash memory array, four control registers, and circuitry that produces analog voltages for programming and erasing. The flash array size of the TMS320F206 and TMS320F240 is 16K × 16 bits, while the TMS320F241 and TMS320F243 incorporate an 8K × 16-bit flash array (see Table 1–1). Unlike most discrete flash memories, the ’F20x/F24x flash module does not require a dedicated state machine, because the algorithms for programming and erasing the flash are executed in software by the DSP core. The use of these sophisticated, adaptive programming algorithms results in reduced chip size and greater programming flexibility. In addition, the application code can manage the use of the flash memory without the requirement of external programming equipment.

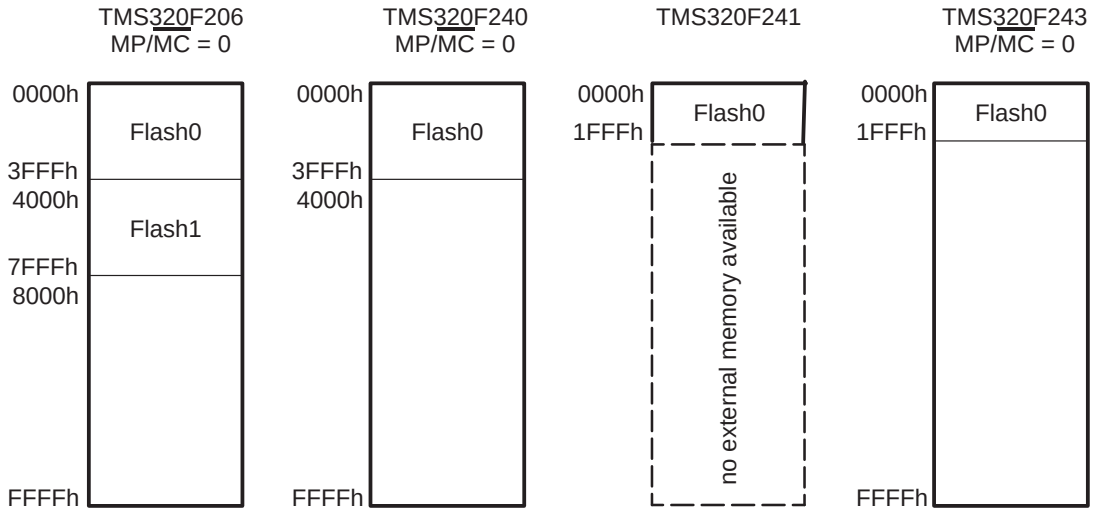
Table 1–1. TMS320 Devices With On-Chip Flash EEPROM

Device	Array Size	Total Flash Memory
TMS320F206	16K	32K†
TMS320F240	16K	16K
TMS320F241	8K	8K
TMS320F243	8K	8K

† Each array can be independently erased.

Simplified memory maps for the program space of the TMS320F20x/F24x devices are shown in Figure 1–1 to illustrate the location of the flash modules.

Figure 1–1. TMS320F20x/F24x Program Space Memory Maps



1.3 Benefits of Embedded Flash Memory in a DSP System

The circuitry density of flash memory is about half that of conventional EEPROM memory, making it possible to approach DRAM densities with flash memory. This increased density allows flash memory to be integrated with a CPU and other peripherals in a single ’F20x/F24x DSP chip. Embedded flash memory expands the capabilities of the ’F20x/F24x DSPs in the areas of prototyping, integrated solutions, and field upgradeable designs.

Embedded flash memory facilitates system development and early field testing. Throughout the development process, the system software can be updated and reprogrammed into the flash memory for testing at various stages. Since flash is a non-volatile memory type, the resulting standalone prototype can be tested in the appropriate environment without the need for battery backup. In addition to its nonvolatile nature, embedded flash memory has the advantage of in-system programming. Unlike some discrete flash or EEPROM chips, embedded flash memory can be programmed without removing the device from the system board. In fact, the embedded flash memory of ’F20x/F24x DSPs can be programmed using hardware emulators which are already an integral part of the DSP development process; no external programming equipment is required.

The embedded flash memory of ’F20x/F24x DSPs also makes these devices ideal for highly integrated, low-cost systems. The initial investment involved with making a ROM memory is not justifiable for certain low-cost applications. Accordingly, when on-chip ROM is not an option, DSP system designers usually resort to using expensive static RAM (SRAM), to store system software and data. The SRAM provides the fast access times required by the DSP, but has the disadvantage of being a volatile memory type. To address the issue of memory volatility, designers often use a low-cost EPROM or flash device to load the SRAM after system power-up. This approach is very expensive, and the increased chip count is often prohibitive. The ’F20x/F24x DSPs, with their on-chip flash memory modules, provide a single chip solution with nonvolatile memory that supports full speed DSP access rates.

Another benefit of embedded flash memory in a DSP system is remote reprogrammability. Field upgradeability is an extremely useful feature for embedded systems. For example, many modem manufacturers offer algorithm upgrades remotely, without requiring the modem to be removed from the host computer system. The same type of feature is also being offered for many handheld consumer products. Adding this capability to a product requires the addition of EEPROM or flash devices, which increase chip count and system cost. Since no external equipment is required to program the embedded flash memory of the ’F20x/F24x DSPs, these devices enable field upgradeability without impacting system cost.

Chapter 2

Flash Operations and Control Registers

The operations that modify the contents of the ’F20x/F24x flash array are performed in software through the use of dedicated programming algorithms. This chapter introduces the operations performed by these algorithms and explains the role of the control registers in this process. The actual algorithms are discussed in Chapter 3.

Topic **Page**

2.1 Operations that Modify the Contents of the 'F20x/F24x Flash Array

Operations that modify the contents of the flash array are generically referred to as either "programming," which drives one or more bits toward the logic zero state, or "erasing," which drives all bits towards the logic one state. It should be noted that since these operations are performed incrementally, a single "programming" or "erasing" operation does not ALWAYS result in a valid logic one or zero. The result of each of these types of operations depends on the initial state of the bit(s) prior to the operation. This is described in more detail below.

Within these two basic types of operations (which are related to the fact that there are only two valid logic levels in the F20x/F24x device) are four distinctly different types of functions which are actually performed.

In the category of "programming" operations, there are three actual types of functions that are performed:

- ❏ Clear – which is used to write ALL array bits to a zero state,
- ❏ Program – which is used to write SELECTED array bits to zero, and
- ❏ Flash-Write – which is used to recover ALL array bits from depletion

In the category of "erase" operations, there is only one type of operation:

- ❏ Erase – which is used to write ALL array bits to a one state.

Clear, Program, Flash-Write, and Erase are the only four functions that are used to modify the flash array.

Assuming that the intent of a modification of the contents of the flash array is to program the array with a selection of ones and zeroes, the following sequence of operations must be performed for proper operation of the flash memory:

1) The array is first CLEARED to all zeroes.
2) The array is then ERASED to all ones.
3) The array is then checked for depletion and recovered using FLASH-WRITE if necessary (note that if Flash-Write is used to recover from depletion, this sequence must be started over again with the Clear and Erase functions).
4) Once the array is properly cleared and erased, and verified not to be in depletion, the array is then PROGRAMMED with the desired selection of zero bits.

This procedure is discussed in complete detail in Chapter 3.

During these operations that are used to modify the contents of the flash array, three special read modes, and a corresponding set of reference voltage levels, are used when reading back data values to verify programming and erase operations.

These read modes and reference levels are:

- VER0 – which is used to verify the logic zero level including margin,
- VER1 – which is used to verify the logic one level including margin, and
- Inverse Erase – which is used to verify depletion recovery.

These concepts are illustrated graphically in Figure 2–1 and summarized in Table 2–1.

Note that ONLY the Erase and the Flash-Write functions are truly “flash” in the sense that these functions actually affect all bits in the array simultaneously. In contrast, bit programming levels in the Program and Clear functions can be controlled individually on a bit-by-bit basis.

Therefore, when using the Erase or Flash-Write functions, the whole array is modified, and then the whole array is read, word by word, to verify whether all words have reached the same value (if not, further iterations of the Erase or Flash-Write functions continue).

In these cases, as mentioned previously, all the bits in the array are modified simultaneously, but some bits may react more quickly, potentially resulting in variation in actual levels on different bits. Therefore, when performing an Erase, it is possible that some bits may reach depletion even before other bits reach the logic one reference level (VER1).

The reason that it is critical to clear the array to a consistent zero level before erasing the array is to give maximum immunity to depletion when erasing. Note, however, that even when following this sequence, some flash arrays may experience depletion, and may require recovery using the Flash-Write function.

In contrast to the true “flash” operations Erase and Flash-Write, after each incremental Program or Clear operation, each bit is tested against the VER0 reference level to determine the exact point at which it has reached the proper value, following which, no further incremental adjustment of the level is made on that bit. Therefore, when the Program or Clear operation is complete, all bits are at the same zero level, which greatly increases proper data retention and depletion immunity for the device. Again, note that the programming and erase operations are discussed in complete detail in Chapter 3.

Figure 2–1. Flash Memory Logic Levels During Programming and Erasing

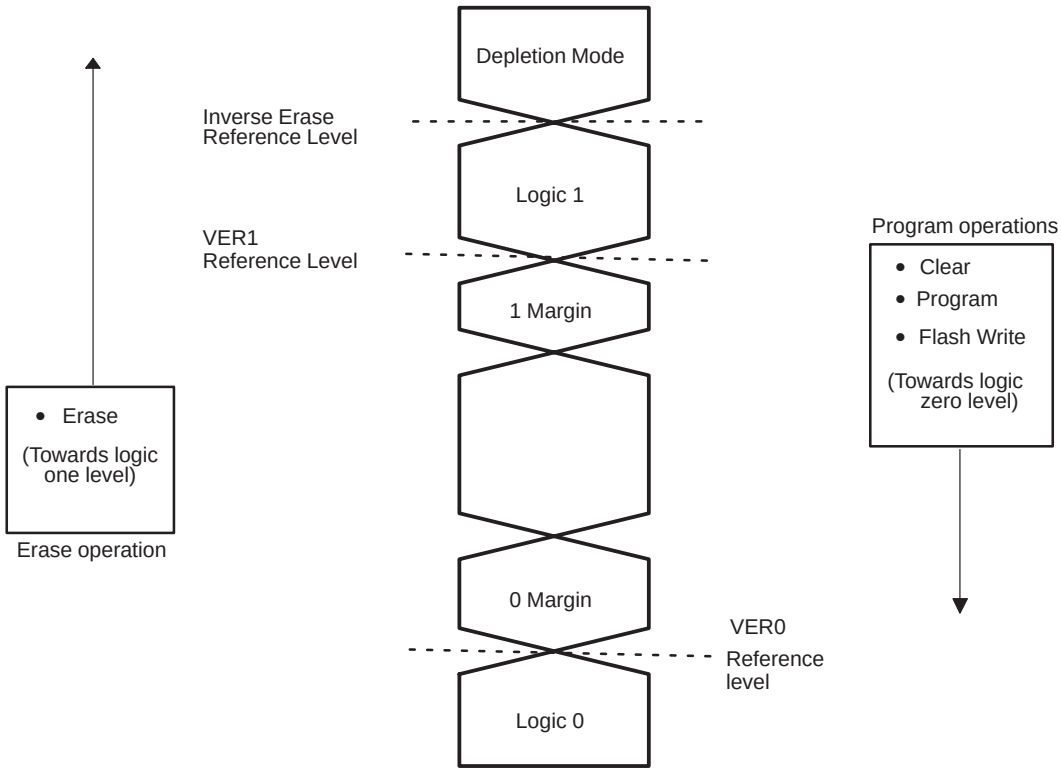


Table 2–1. Operations that Modify the Contents of the Flash Array

Change in Bit Level			
Towards Logic 1		Towards Logic 0	
Function	Reference Level	Function	Reference Level
Erase (all bits)	VER1	Program (selected bits)	VER0
		Clear (all bits)	VER0
		Flash-Write (all bits)	Inverse Erase

2.2 Accessing the Flash Module

In addition to the flash memory array, each flash module has four registers that control operations on the flash array. These registers are:

- Segment control register (SEG_CTR)
- Test register (TST)
- Write address register (WADRS)
- Write data register (WDATA)

The flash module operates in one of two modes: one in which the flash memory is accessed directly by the CPU, and one in which the memory array cannot be accessed directly, but the four control registers are accessible. This mode is used for programming. Each flash module has a flash access-control register that selects between these two access modes. The register is a single-bit, I/O-mapped register.

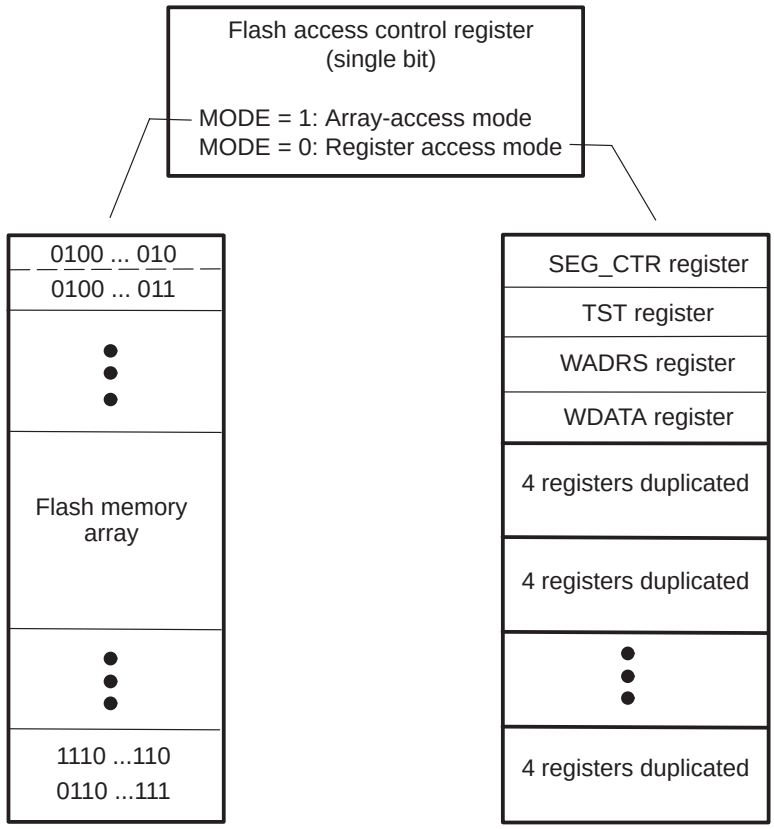
The two access modes are summarized as follows:

- Array-access mode. You can access the flash array in the memory space decoded for the flash module. The flash module remains in this mode most of the time, because it allows the DSP core to read from the memory array.
- Register-access mode. You can access the four control registers in the memory space decoded for the flash module. This mode is used for programming. When the flash module is in register-access mode, the registers are repeated every four address locations within the flash module's address range.

The flash array is not directly accessible as memory in register-access mode, and the control registers are not directly accessible in array-access mode.

Figure 2–2 shows memory maps of the flash array in register and array access modes.

Figure 2–2. Memory Maps in Register and Array Access Modes



2.2.1 TMS320F206 Flash Access-Control Register

Because each flash module has an access-control register associated with it, the ’F206 has two access-control registers. These registers are standard I/O-mapped registers that can be read with an IN instruction and must be modified with an OUT instruction.

- ❏ F_ACCESS0 is mapped in I/O space at 0FFE0h.
- ❏ F_ACCESS1 is mapped in I/O space at 0FFE1h.

The MODE bit (bit 0) of the access-control register selects the access mode:

MODE = 0	Register-access mode
MODE = 1	Array-access mode

Bits 15–1 of each access-control register are always read as 0 and are unaffected by writes.

Although the function is the same, the access control registers of the ’F206 device are mapped at different addresses from that of the ’F24x devices, and their values are modified in a different way.

2.2.2 TMS320F24x Flash Access-Control Register

The access-control register of the ’F24x devices is a special type of I/O-mapped register that cannot be read. The register is mapped at I/O address 0FF0Fh, and it functions as indicated below.

Note:

For both the IN and OUT instructions, the data operand (dummy) is not used, and can be any valid memory location.

An OUT instruction using the register address as an I/O port places the flash module in register-access mode.

For example:

```
OUT    dummy, 0FF0Fh ;Selects register-access mode
```

An IN instruction using the register address as an I/O port places the flash module in array-access mode.

The data operand (dummy) is not used, and can be any valid memory location.

For example:

```
IN     dummy, 0FF0Fh ;Selects array-access mode
```

2.3 Flash Module Control Registers

Table 2–2 lists the control registers and their relative addresses within the four locations that repeat throughout the module's address range.

Table 2–2. Flash Module Control Registers

Relative Address	Register Name	Description	Described in ... Section	Page
0	SEG_CTR	Segment control register. The eight MSBs enable specific segments for programming. Setting a bit to 1 enables the segment. The eight LSBs control the program, erase, and verify operations of the module.	2.3.1	2-5
1	TST	Test register. Reserved for test; not accessible to the user.	2.3.2	2-8
2	WADRS	Write address register. Holds the address for a write operation.	2.3.3	2-8
3	WDATA	Write data register. Holds the data for a write operation.	2.3.4	2-8

2.3.1 Segment Control Register (SEG_CTR)

SEG_CTR is a 16-bit register that initiates and monitors the programming and erasing of the flash array. This register contains the bits that initiate the active operations (the WRITE/ERASE field and EXE bit), those used for verification (VER0 and VER1), and those used for protection (KEY0, KEY1, and SEG7–SEG0). All bits of SEG_CTR register are cleared to 0 upon reset.

SEG_CTR is shown in Figure 2–3 and the fields are described in Table 2–3.

Figure 2–3. Segment Control Register (SEG_CTR)

15	14	13	12	11	10	9	8	7	6	5	4	3	2 1	0
SEG7	SEG6	SEG5	SEG4	SEG3	SEG2	SEG1	SEG0	Res	KEY1	KEY0	VER0	VER1	WRITE/ ERASE	EXE
RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0	X	RW–0	RW–0	RW–0	RW–0	RW–0	RW–0

Legend: R = read
W = write
–0 = value after reset
X = don't care

Table 2–3. Segment Control Register Field Descriptions

Bits	Name	Description
15–8	SEG7–SEG0	Segment enable bits. Each of these bits protects the specified segment against programming or enables programming for the specified segment in the array. Any number of segments (from 0 to 7 in any combination) can be enabled at any one time. See Table 2–4 for segment address ranges. EXE must be cleared to modify the SEGx bits. SEGx = 1 enables programming of the corresponding segment. SEGx = 0 protects the segment from programming.
7	Reserved	This bit is not affected by writes, and reads of this bit are undefined.
6–5	KEY1, KEY0	Execute key bits. A binary value of 10 must be written to these bits in the same DSP core access in which the EXE bit is set for the selected operation (erase, program, or flash-write) to start. KEY1 and KEY0 must be cleared in the same write access that clears EXE. These bits are used as additional protection against inadvertent programming or erasure of the array. These bits are read as 0s.
4–3	VER0, VER1	Verify bits. These bits select special read modes used to verify proper erasure or programming. Possible values: 00: Normal read mode 01: Verify 1s (VER1) read mode to verify margin of 1s for proper erasure 10: Verify 0s (VER0) read mode to verify margin of 0s for proper programming 11: Inverse-read mode; tests for bits erased into depletion
2–1	WRITE/ERASE	Write/erase enable field. These bits select the program, erase, or flash-write operation. However, modification of the array data does not actually start until the EXE bit is set. Reset clears these bits to zero. Possible values: 00: Read operation is enabled. These bit values are required to read the array. 01: Erase operation is enabled 10: Write operation is enabled 11: Flash-write operation is enabled
0	EXE	Execute bit. In conjunction with WRITE/ERASE, KEY1, and KEY0, this bit controls the program, erase, and flash-write operations. Setting EXE starts and stops programming and erasing of the flash array. The KEY1 and KEY0 bits must be written in the same write access that sets EXE, and EXE must be cleared in the same write access that clears KEY1 and KEY0. EXE must be cleared to modify the SEGx bits.

Note: The segment enable bits are not intended for protection during the erase or flash-write operations. During these operations, *all* segments must be enabled.

Table 2–4. Flash Array Segments Summary

SEG7–SEG0 Bits								'F206/F240 Flash Module†		'F241/F243 Flash Module	Array Segment Enabled
15	14	13	12	11	10	9	8	Flash0	Flash1		
0	0	0	0	0	0	0	1	0000–07FFh	4000–47FFh	0000–03FFh	0
0	0	0	0	0	0	1	0	0800–0FFFh	4800–4FFFh	0400–07FFh	1
0	0	0	0	0	1	0	0	1000–17FFh	5000–57FFh	0800–0BFFh	2
0	0	0	0	1	0	0	0	1800–1FFFh	5800–5FFFh	0C00–0FFFh	3
0	0	0	1	0	0	0	0	2000–27FFh	6000–67FFh	1000–13FFh	4
0	0	1	0	0	0	0	0	2800–2FFFh	6800–6FFFh	1400–17FFh	5
0	1	0	0	0	0	0	0	3000–37FFh	7000–77FFh	1800–1BFFh	6
1	0	0	0	0	0	0	0	3800–3FFFh	7800–7FFFh	1C00–1FFFh	7

† The TMS320F206 has two flash modules. The TMS320F240 device uses the address ranges shown for Flash0.

Although segmentation is not supported during erase (i.e., the entire array must be erased simultaneously), the segment enable bits can be used to protect portions of the array against unintentional programming. This is useful for applications in which different portions of the array are programmed at different times. For example, an application might program the flash module with a large table in 2K $\times$ 16 blocks. Some time after the first block is programmed, the next block is programmed. The segment enable bits can be used to prevent corruption of the first block while the second block is being programmed.

2.3.2 Flash Test Register (TST)

The flash test register (TST) is a 5-bit register used during manufacturing test of the flash array. This register is not accessible to the DSP core.

2.3.3 Write Address Register (WADRS)

The write address register (WADRS) is a 16-bit register that holds the latched write address for a programming operation. In array-access mode, this register is loaded with the value on the address bus when you are writing a data value to the flash module. It can be loaded directly in register-access mode by writing to it.

2.3.4 Write Data Register (WDATA)

The write data register (WDATA) is a 16-bit register that contains the latched write data for a programming operation. In array-access mode, this register can be loaded by writing a data value to the flash module. It can be loaded directly in register-access mode by writing to it. The WDATA register must be loaded with the value FFFFh before an erase operation starts.

2.4 Read Modes

The ’F20x/F24x flash module uses four read modes and corresponding sets of reference levels:

- ❑ Standard
- ❑ Verify 0s (VER0)
- ❑ Verify 1s (VER1)
- ❑ Inverse-erase

Read mode selection is accomplished through the verify bits (bits 3 and 4) in SEG_CTR during execution of the algorithms.

In the standard read mode of the ’F20x/F24x flash module, the supply voltage (V_{DD}) is internally applied to the cell to select it for reading. The VER0, VER1, and inverse-erase read modes differ from the standard read mode in the internal voltage level applied to the flash cell.

Because the program and erase operations must provide sufficient margin on 1s and 0s to ensure data retention, the verify 0s (VER0) and verify 1s (VER1), are provided on the flash module to check for sufficient margin.

The VER0 and VER1 read modes provide a method for adjusting the level on the cells during programming or erasing, beyond the point required for reading a 0 or a 1, creating the required logic level margin. In VER0 mode, a voltage closer to an ideal logic zero level than necessary to read a logic zero is internally applied to the cell to select it for reading. This is the worst-case condition for reading a programmed cell, and if a cell can be read as 0 in VER0 mode, then it can also be read as 0 in standard read mode. Similarly, in the VER1 read mode, a voltage closer to an ideal logic one level than necessary to read a logic one is internally applied to the cell to select it for reading. This is the worst-case condition for reading an erased cell, and if a cell can be read as 1 in the VER1 mode, then it can be read as 1 in standard read mode.

The inverse-erase read mode detects flash bits that are in depletion mode. This read mode applies a voltage to all array cells so that all cells are deselected. The entire array can be tested for bits in depletion mode by reading the first row (32 words) of the array in inverse-erase read mode. If there are no bits in depletion mode, all 32 words are read as 0000h.

2.5 Program Operation

The program operation of the ’F20x/F24x flash module loads the application-specific data (a pattern of 0s) into the flash array. The basis of the operation is applying a program pulse to a single word of flash memory. The term *program pulse* refers to the time during the program operation between the setting and the clearing of the EXE bit (bit 0 of SEG_CTR). During the program pulse, charge is added to the addressed bits via the programming mechanism. Several program pulses may be required to fully program the bits of a word, and the application of program pulses is controlled by the programming algorithm.

The flash location to be programmed is specified by the address in the WADRS register, and the data pattern to be programmed is loaded into the WDATA register. Only the bits that contain a 0 are programmed; any bit positions containing a 1 remain unchanged. (See sections 2.3.3 and 2.3.4 for information about how to load the WADRS and WDATA registers.)

To assure that the 0 bits are programmed with enough margin, the reads associated with programming are performed using the VER0 read mode. After a program pulse has been applied, the byte is read back in VER0 mode to assure that programmed bits can be read as 0 over the entire operating range of the device.

The flash module supports programming of up to eight bits of data. Therefore, although the flash bits are addressed on 16-bit word boundaries, only eight bits can be programmed at a time. The algorithm must limit the programming to eight bits by masking the word to be programmed before writing it to the WDATA register. For example, to mask off the upper byte while programming the lower byte, the data value is logically 0Red with 0FF00h in software. When a program pulse is applied, only the selected bits are programmed.

2.6 Erase Operation

The erase operation of the ’F20x/F24x flash module prepares the flash array for programming and enables reprogrammability of the flash array. Before the array can be erased, all bits must be programmed to 0s. This procedure of programming all array locations in preparation for the erase is called *clearing the array*. During the erase, all bits in the array are changed from 0s to 1s. After the erase is finished, a depletion mode test is made to determine whether any bits have been over-erased. If over-erased bits are detected, they must be recovered with the flash-write algorithm, and the clear and erase algorithms must be repeated.

An *erase pulse* is the time during the erase operation between the setting and the clearing of the EXE bit (bit 0 of SEG_CTR). During the erase pulse, the level on all array bits is modified via the erase mechanism.

Erasing the flash array is a block operation. During the erase pulse, all array bits are affected simultaneously. (See Figure 2–1, *Flash Memory Logic Levels During Programming and Erasing*, on page 2-4 for an illustration of this mechanism.) Multiple erase pulses may be required to fully erase all bits in the array, and the application of erase pulses is controlled by the erase algorithm.

The erase operation uses the VER1 read mode to determine when erasure is complete. After erasure is complete, the inverse-erase read mode is used to determine if any bits are over-erased. For more information about these read modes, see section 2.4, *Read Modes*, on page 2-12.

2.7 Recovering From Over-Erasure (Flash-Write Operation)

Generally, not all bits in the flash array have the same amount of charge removed with each erase pulse. By the time all bits have reached the VER1 read margin (and erase is complete), some of the bits in the array may be over-erased. They are said to be in depletion mode. If even one single flash cell is over-erased into depletion mode, it is always read as logic 1 and can corrupt the reading of other bits. This condition must be detected and corrected, because it also inhibits reprogramming of the flash array.

The ’F20x/F24x flash array employs the flash-write operation to recover bits that are erased into depletion mode. The flash-write operation is similar to the erase operation in that it affects all bits in the array simultaneously. This enables recovery of multiple bits from depletion mode, but requires the flash-write operation to be followed by the clear and erase operations to restore the erase margin on all bits.

A *flash-write pulse* is the time during the flash-write operation between the setting and the clearing of the EXE bit (bit 0 of SEG_CTR). During the flash-write pulse, all array bits are affected simultaneously. (See Figure 2–1, *Flash Memory Logic Levels During Programming and Erasing*, on page 2-4 for an illustration of this mechanism.) Multiple flash-write pulses may be required to fully recover all bits in the array, and the application of flash-write pulses is controlled by the flash-write algorithm.

The flash-write operation uses the inverse-erase read mode and inverse-erase reference level to detect bits that are in depletion mode. For more information about the inverse-erase read mode, see section 2.4, *Read Modes*, on page 2-12.

2.8 Reading From the Flash Array

Once the array is programmed, it is read in the same manner as other memory devices on the DSP memory interface. The flash module operates with zero wait states. When you are reading the flash module, the flash segment control register (SEG_CTR) bits should be 0 and the flash array must be in the array-access mode.

2.9 Protecting the Array

After the flash memory array is programmed, it is desirable to protect the array against corruption. The flash module of the ’F20x/F24x DSPs includes several protection mechanisms to prevent unintentional modification of the array.

Flash programming is facilitated via the supply voltage connected to the VCCP pin. If this pin is grounded, the program operation will not modify the flash array. Note, that grounding the VCCP pin does not prevent the erase operation; other protection mechanisms for the erase operation are discussed below.

The control registers provide the following mechanisms for protecting the flash array from unintentional modification.

- ❏ Segment enable bits
- ❏ EXE, KEY0, and KEY1 bits
- ❏ WDATA register

An array segment is prevented from being programmed when the corresponding segment enable bit in the SEG_CTR is cleared to zero. Additionally, all segment enable bits are cleared by reset, making unintentional programming less likely. Even if the segment enable bits are set to one, the program, erase, and flash-write operations are not initiated unless the appropriate values are set in the EXE, KEY0, and KEY1 bits of the SEG_CTR.

At the start of an operation, the KEY1 and KEY0 bits must be written in the same write access that sets EXE. When the program pulse, erase pulse, or flash-write pulse is finished, EXE must be cleared in the same write that clears KEY1 and KEY0. The data and address latches are locked whenever the EXE bit is set, and all attempts to read from or write to the array are ignored (read data is indeterminate). Once the EXE bit is set, all register bits are latched and protected. You must clear EXE to modify the SEGx bits. This protects the array from inadvertent change. Unprotected segments cannot be masked in the same register load with the deactivation of EXE. Additional security is provided by a function of the WDATA register to prevent unintentional erasure. The WDATA register must be loaded with FFFFh before the erase operation is initiated. If the register is not loaded with this value, the array will not be modified.

Chapter 3

Algorithm Implementations and Software Considerations

This chapter discusses the implementations of the algorithms for performing the operations described in the previous chapter. It also discusses items you must consider when incorporating the algorithms into your ’F20x/F24x DSP application code.

Topic **Page**

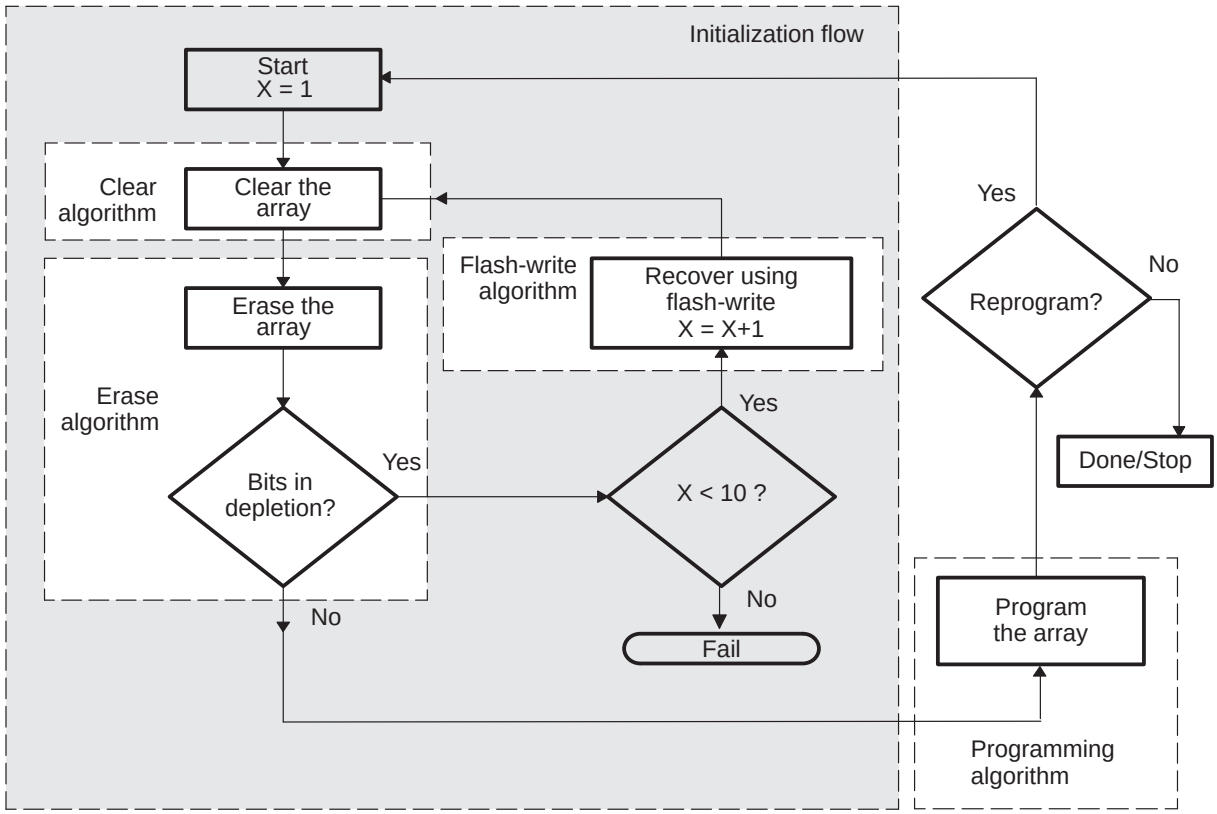
3.1 How the Algorithms Fit Into the Program-Erase-Reprogram Flow

The algorithms discussed in this chapter can be used to reprogram the ’F20x/F24x flash module multiple times. The clear algorithm, erase algorithm, and flash-write algorithm are used to prepare the flash memory for programming, while the programming algorithm is used to write a desired pattern of 0s to the array (program the array).

The programming algorithm and the clear algorithm are both implementations of the program operation. The difference between the two is the data that is written: the programming algorithm programs the user data, while the clear algorithm uses all 0s. All of the algorithms can be viewed as portions of a single flow diagram, as shown in Figure 3–1.

Note that in the algorithm flowcharts, the variable X represents the number of attempts at depletion recovery using the flash-write algorithm. It has been shown that if flash-write is not successful in depletion recovery after ten attempts, depletion recovery is not possible, and a device failure has occurred. Therefore, if ten flash-write attempts at depletion recovery are not successful, the algorithm returns a device failure error message.

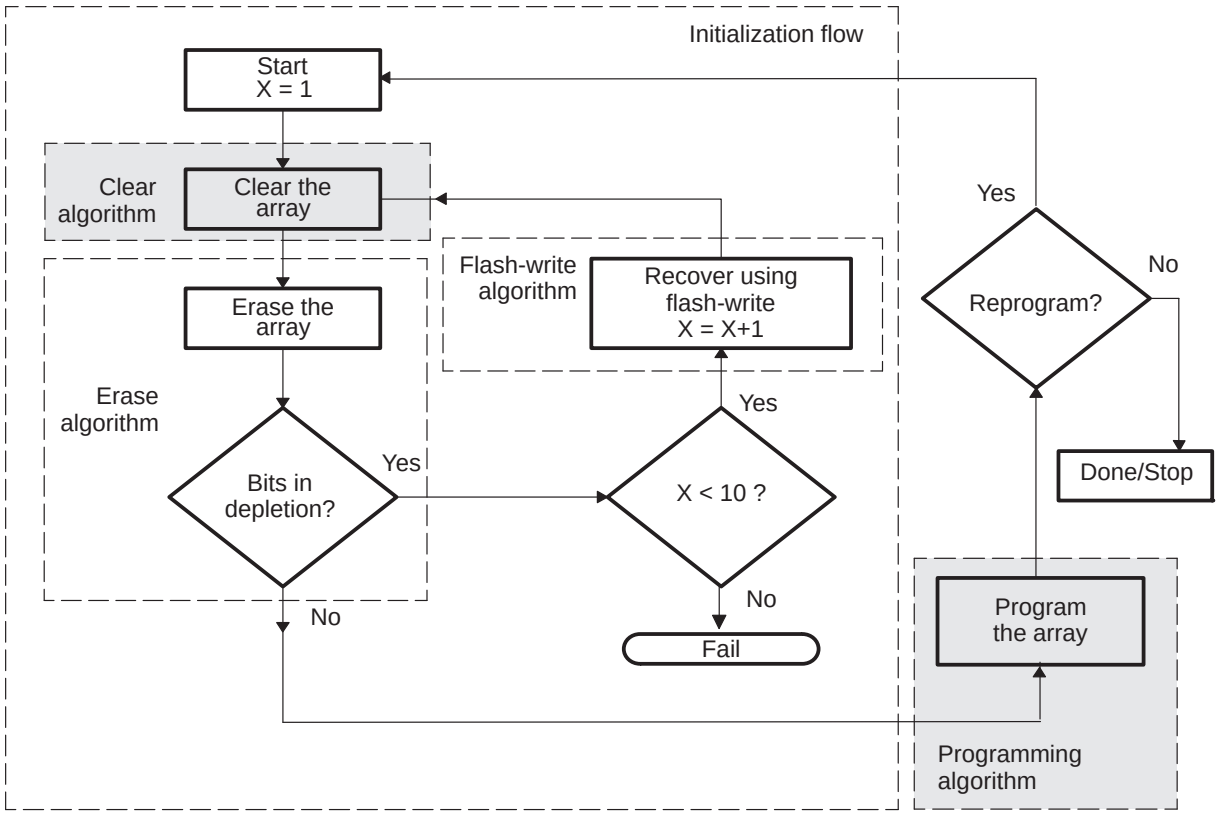
Figure 3–1. Algorithms in the Overall Flow



3.2 Programming (or Clear) Algorithm

The programming algorithm sequentially writes any number of addresses with a specified bit pattern.This algorithm is used to program application code or data into the flash array. With a slight modification, the same algorithm performs the clear portion of the initialization flow (i.e., programs all bits to zero). In this role, the algorithm is called the clear algorithm. For the clear algorithm, the values programmed are always 0000h, while the values for application code can be any combination of 1s and 0s. Figure 3–2 highlights the programming and clear algorithms' place in the overall flow.

Figure 3–2. The Programming Algorithm in the Overall Flow



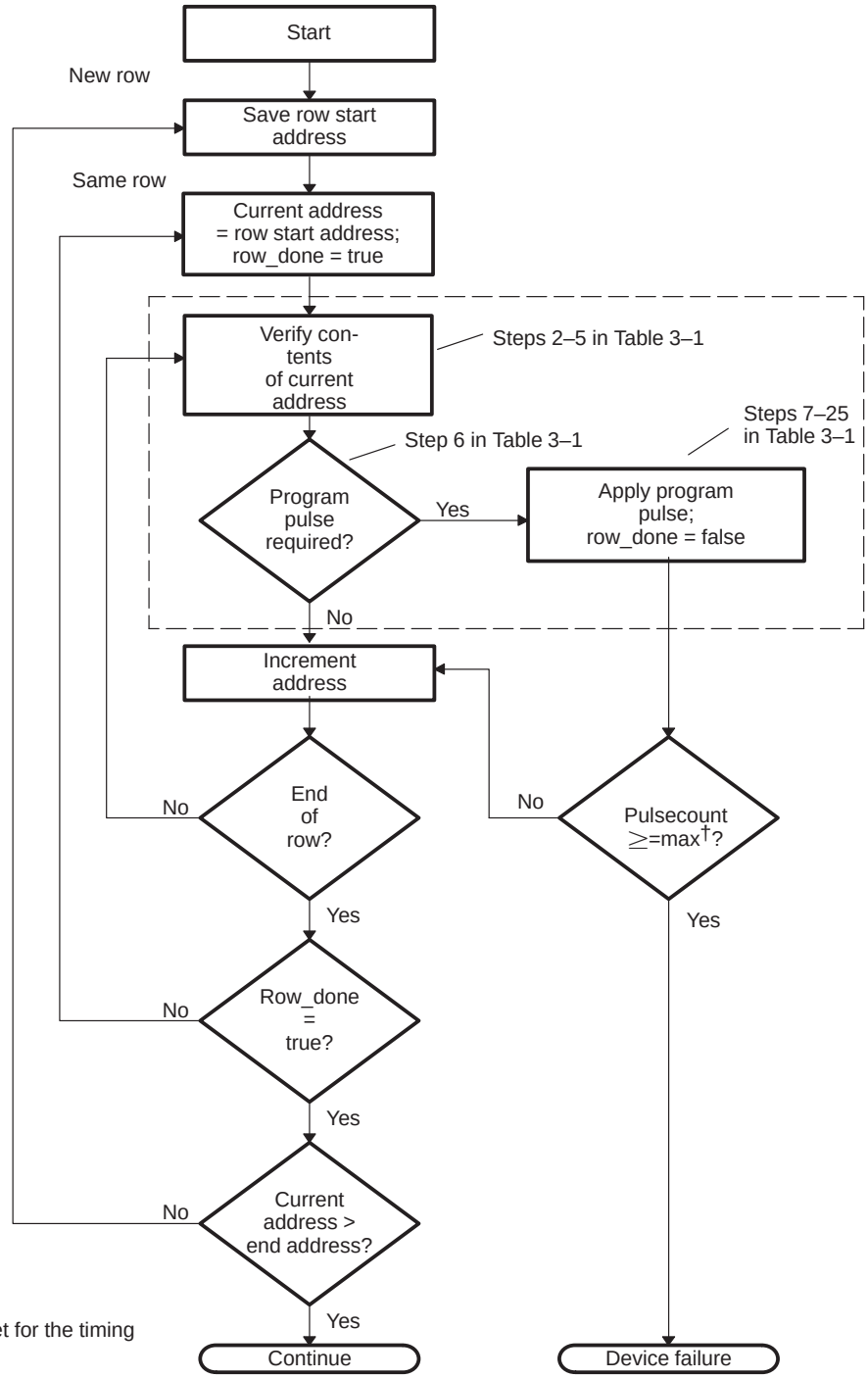
The main feature of the program/clear algorithm is the concept of programming an entire row of bits in a group. The 'F20x/F24x flash array is organized in rows of 32 words. That is, addresses 0000h through 001Fh are physically located on the same row of the flash memory array. The array is designed so that there is a dependence between the charge levels on adjacent (even–odd) addresses during programming. Programming the bits of an odd address reduces the charge margin of the programmed bits (the 0s) in the preceding adjacent (even) address within the row. Similarly, programming the bits of an even address reduces the charge margin of the programmed bits in the next adjacent (odd) address within the row. Because of this dependence, if each address is programmed individually, the charge levels among programmed bits is not uniform. The programming algorithm improves the uniformity of charge levels on programmed bits by programming all of the words of a row in a group. For example, the contents of address 0000h is compared with the data to be programmed and one program pulse is applied if necessary. The same procedure is performed on addresses 0001h through 001Fh. The procedure repeats starting at address 0000h until no more program pulses are required for any address in the row. The number of iterations of this loop equals the maximum number of program pulse*s* required to program the bits in the row.

The flow for the programming algorithm is shown in Figure 3–3, and the assembly code is given in Appendix A.

An important consideration for programming the flash array is the CPU frequency range for the application. Because of the actual implementation of the flash memory circuitry, a 0 bit is most easily read at high frequency; programmed bits have less margin when read at lower frequency. So, if the application requires a variable CPU clock rate, programming should be performed at the lowest frequency in the range. (A similar condition exists for the erase operation, which requires execution of the erase algorithm at the highest frequency in the range. See section 3.3, page 3-10.)

Only the read portion of the program operation must be performed at the lower frequency, because the read is used to determine margin. The read operation can be extended by sequentially executing multiple reads on the same location. Because the same address is selected the entire time and internal control signals are maintained between reads, the final read is equivalent to a slow read. For example, if the DSP core is executing the programming algorithm at a CLKOUT rate of 20 MHz (50 ns), sequentially reading a location three times is equivalent to reading it once at 6.67 MHz (150 ns). This is important, because it facilitates execution of the program and erase algorithms at the same CLKOUT rate.

Figure 3–3. Programming or Clear Algorithm Flow



† See the device data sheet for the timing parameter values.

Another important consideration is the total amount of time required to do the programming. The number of programming pulses required to completely program a flash memory cell increases as ambient temperature increases and/or supply voltage decreases. More programming pulses are required when the minimum supply voltage is used than when the nominal or maximum supply voltage is used. The number of program pulses required also increases throughout the life of the device, as more program-erase cycles are carried out. The device data sheet specifies the maximum number of program pulses under all operating conditions; use this number when you calculate the maximum amount of time required for programming.

The algorithm incorporates the steps for applying a program pulse (outlined in Table 3–1) along with some other techniques to ensure margin. In general, not all flash bits require the same number of program pulses to reach the programmed margin level. For this reason, the programming algorithm applies a series of short program pulses until the memory location is programmed. However, to understand how the series of program pulses works, you must first understand how the algorithm applies a single program pulse. Table 3–1 outlines the steps involved in verifying programmed bits and applying a single pulse to each of the upper and lower bytes of a single location. This process corresponds to the steps enclosed in the dashed box in the flowchart in Figure 3–3.

Table 3–1. Steps for Verifying Programmed Bits and Applying One Program or Clear Pulse

Step	Action	Description
1	Power up the V_{CCP} pin.	Set the V_{CCP} pin to V_{DD}. If the V_{CCP} pin for the flash module to be programmed is not set to V_{DD}, then the array will not be programmed.
2	Activate VER0 mode.	Set the VER0 bit in SEG_CTR (load SEG_CTR with 0010h).
3	Delay for VER0 reference voltage stabilization.	The CPU executes a delay loop for the $t_{d(VERIFY\text{-}SETUP}{}^{\dagger})$ time period.
4	Read flash array contents for verification.	The CPU reads the addressed location. The flash module must be in array-access mode (see section 2.2, *Accessing the Flash Module*, page 2-5).
5	Deactivate VER0 mode.	Clear the VER0 bit in SEG_CTR (load SEG_CTR with 0000h).
6	Compare contents of flash location (16 bits) with desired data.	If the verification passes (i.e., if the data read in step 4 is equal to the desired data value), then no further program pulses are required. The flash word has been programmed with the desired data value. The program or clear function is completed and this algorithm is exited. If the verification fails (i.e., if the data read in step 4 is not equal to the desired data value), then proceed to step 7.

Table 3–1. Steps for Verifying Programmed Bits and Applying One Program or Clear Pulse (Continued)

Step	Action	Description
7	Mask the data to program lower byte.	Mask any bits in the lower byte that do not require programming (are already read as zero), and mask off upper byte. Recall that the algorithm should mask one byte while programming the other because a maximum of eight bits can be programmed simultaneously.
8	Load WADRS and WDATA registers.	If the flash module is in array access mode, write the data to be programmed to its address. If the flash module is in register access mode, load the individual registers directly.
10	Activate the WRITE/ERASE field and enable segments.	Set the WRITE/ERASE field in SEG_CTR to 10 and set the corresponding segment enable bits (SEG0–SEG7) for the segments where the programmed word resides.

† See the device data sheet for the timing parameter values.

11	Wait for internally generated supply voltage stabilization time.	The CPU executes a delay loop for the $t_{d(PGM–MODE}$†$_{)}$ time period.
12	Initiate the program pulse.	Load the EXE, KEY1, and KEY0 bits with 1, 1, and 0, respectively. All three bits must be loaded in the same write cycle. The segment enable bits and the WRITE/ERASE field must also be maintained.
13	Delay for program pulse time.	The CPU executes a delay loop for the $t_{d(PGM}$†$_{)}$ time period.
14	Terminate the program pulse.	Clear the WRITE/ERASE field and EXE bit in SEG_CTR (e.g., load SEG_CTR with 0000h).
15	Delay for array stabilization time.	The CPU executes a delay loop for the $t_{d(BUSY}$†$_{)}$ time period.
16–25	Program upper byte if necessary.	Repeat steps 7–15 for the upper byte. Mask the lower byte to 1s when programming the upper byte.

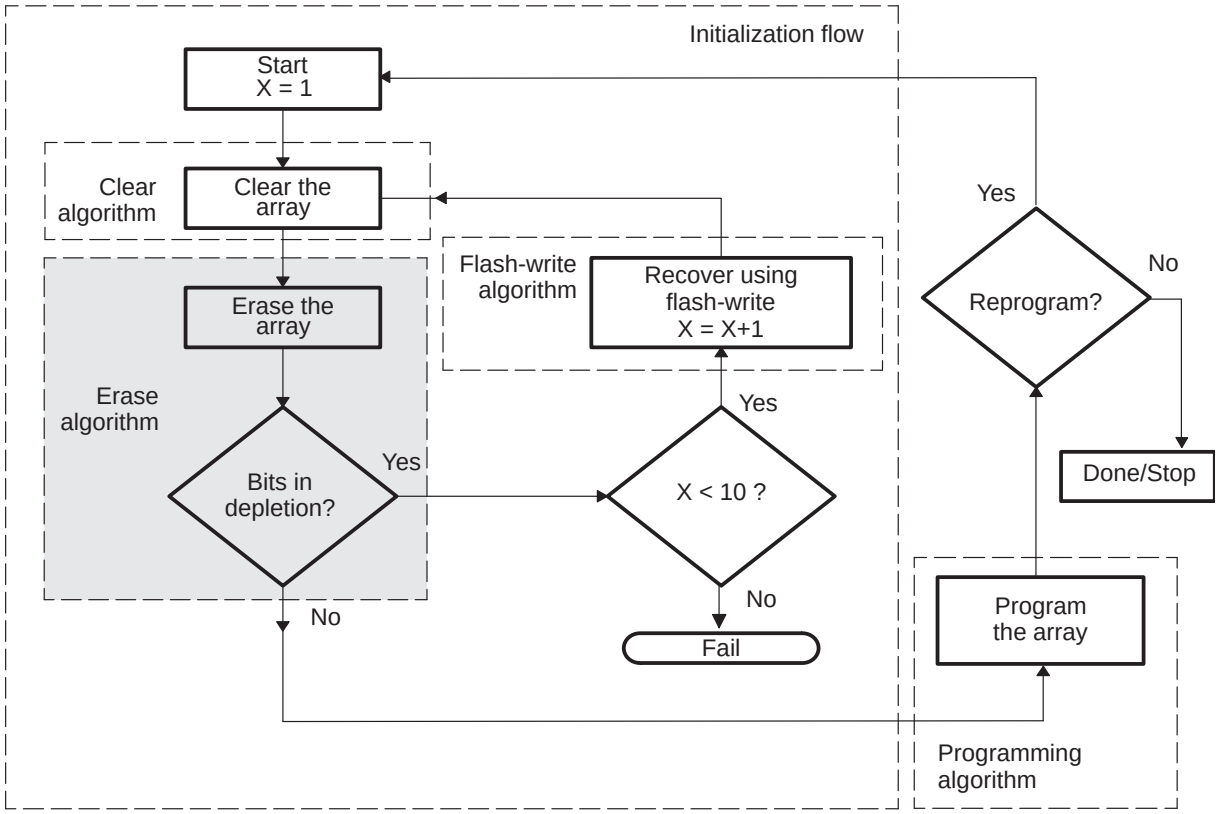
† See the device data sheet for the timing parameter values.

Before each program pulse is applied, a read of the byte is performed to determine which bits have reached the programmed level. Any bits that have reached the programmed level are masked (set to 1 in the WDATA register). This method of programming provides uniform charge levels among programmed bits, whereas using a single, long program pulse could result in some bits having much more charge than others. The uniformity of charge levels among bits has the primary effect of reducing programming time and the secondary effect of reducing the time for a subsequent erase operation. To assure that the bits are programmed with enough margin, the reads associated with programming use the VER0 read mode.

3.3 Erase Algorithm

The erase algorithm follows the clear algorithm in executing the entire initialization flow. Figure 3–4 highlights the erase algorithm's place in the overall flow.

Figure 3–4. Erase Algorithm in the Overall Flow



The erase algorithm consists of multiple iterations of a loop with one erase pulse applied in each iteration. Table 3–2 outlines the steps involved in applying a single erase pulse.

Table 3–2. Steps for Applying One Erase Pulse

Step	Action	Description
1	Power up the V_{CCP} pin.	Set V_{CCP} pin to V_{DD}. If the V_{CCP} pin for the flash module to be erased is not set to V_{DD}, then the array will not be erased properly.
2	Load WDATA register with FFFFh.	This load overrides the erase protection mechanism.
3	Activate the erase mode and enable segments.	Set the WRITE/ERASE field to 01 and set SEG0–SEG7 bits in the SEG_CTR register. The flash module must be in register-access mode (see section 2.2).
4	Wait for internally generated supply voltage stabilization time.	The CPU executes a delay loop for the $t_{d(ERASE-MODE}$†$)$ time period.
5	Initiate the erase pulse.	Load the EXE, KEY1, and KEY0 bits with 1, 1, and 0, respectively. All three bits must be loaded in the same write cycle. The segment enable bits and the WRITE/ERASE field must also be maintained.
6	Delay for erase pulse time.	The CPU executes a delay loop for the $t_{d(ERASE}$†$)$ time period.
7	Terminate the erase pulse.	Clear the EXE bit and WRITE/ERASE field in the SEG_CTR register (load SEG_CTR with 0000h to clear all bits).
8	Delay for mode deselect time.	CPU executes a delay loop for the $t_{d(BUSY}$†$)$ time period.

† See the device data sheet for the timing parameter values.

At the beginning of each iteration, a read operation is performed on all the bits in the array to determine if an erase pulse is required. Erasure is complete when all array locations are read as FFFFh. To assure that the flash array is erased with enough margin, the reads associated with the erase use the VER1 read mode. Additional margin can be gained during the erase operation if the reads are performed using *address complementing.* When the array is read with address complementing, the following sequence is used for each address read:

1) All of the bits of the address to be read are complemented.

2) The contents of the resulting address are read.

3) The value read at the complemented address is discarded.

4) The actual address is restored.

5) The contents of the restored address are read.

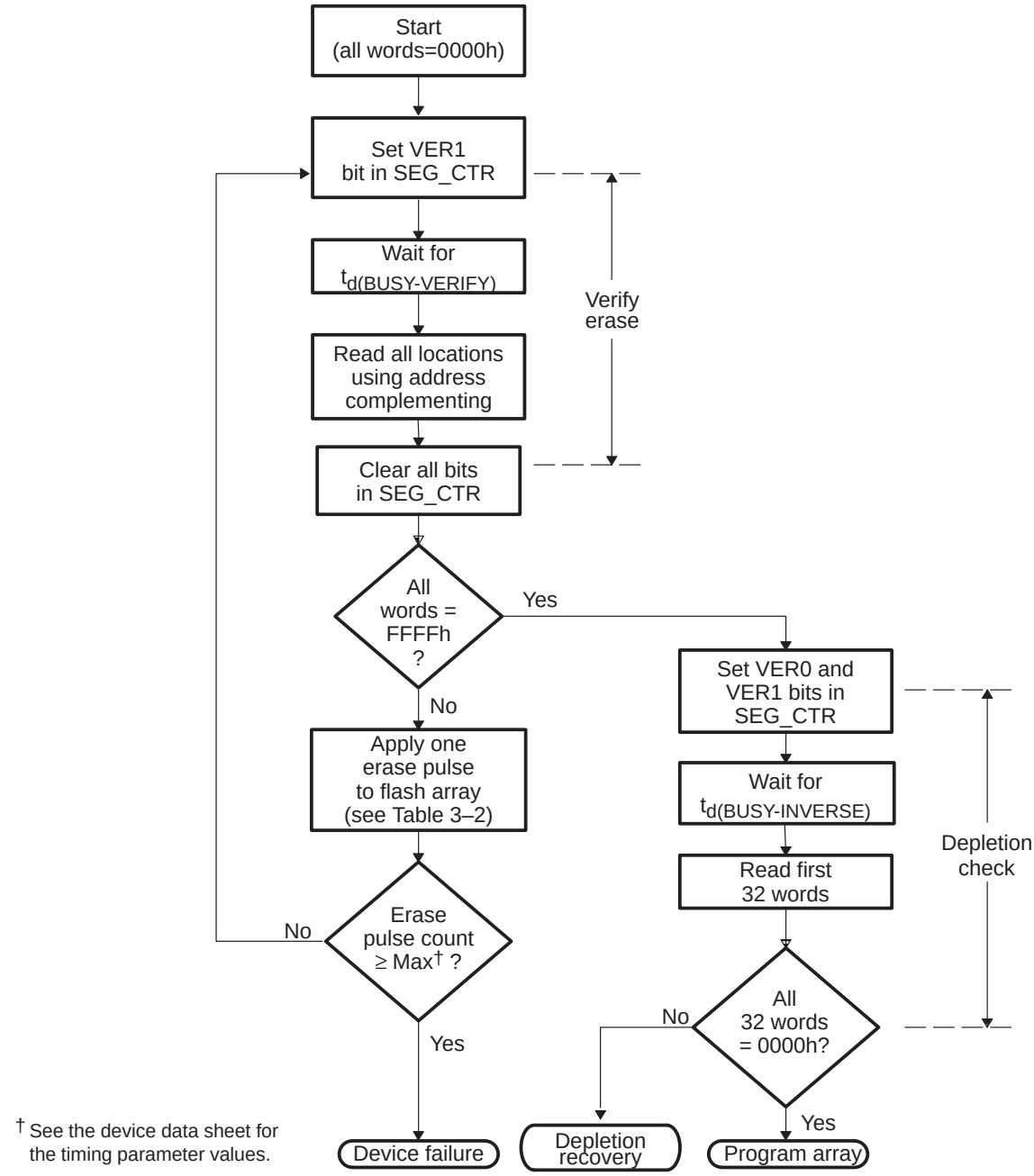
The advantage of this approach is that it forces the worst-case switching condition on the flash addressing logic during the reads, thus improving the margin of the erase. Address complementing on the ’F20x/F24x can be accomplished easily by using the XOR instruction to complement the bits of the address.

An important consideration for erasing the flash array is the CPU frequency range for the application. Because of the actual implementation of the flash memory circuitry, a logic 1 is most easily read at low frequency; erased bits have less margin when read at higher frequency. Accordingly, if the application requires a variable CPU clock rate, the erase should be performed at the highest frequency in the range. (A similar condition exists for the programming operation, which requires execution of the programming algorithm at the lowest frequency in the range. See section 3.2, page 3-4.)

Another important consideration is the total amount of time required to erase the array. The number of erase pulses required to completely erase a flash memory cell increases as ambient temperature increases or decreases relative to the nominal temperature and as supply voltage decreases. More erase pulses are required when the ambient temperature is toward the extremes of the operating range. Also, more erase pulses are required when the minimum supply voltage is used than when the nominal or maximum supply voltage is used. The number of erase pulses required also increases throughout the life of the device, as more program-erase cycles are carried out. The device data sheet specifies the maximum number of erase pulses under all operating conditions; use this number when you calculate the maximum amount of time required for the erase algorithm.

The complete erase algorithm including depletion check is shown in the flowchart in Figure 3–5.

Figure 3–5. Erase Algorithm Flow

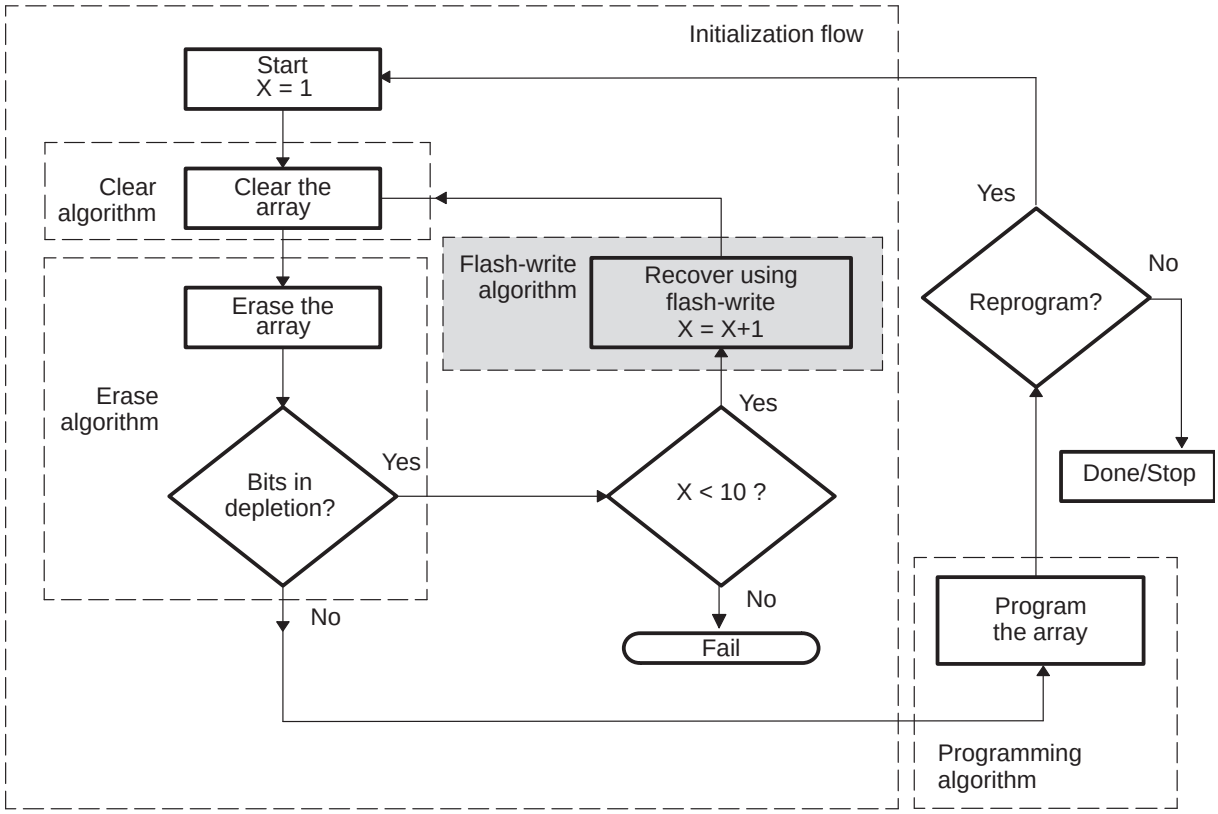


† See the device data sheet for the timing parameter values.

3.4 Flash-Write Algorithm

The flash-write operation recovers bits in depletion mode, which can be caused by over-erasure. The flash-write algorithm's place in the overall flow is highlighted in Figure 3–6.

Figure 3–6. Flash-Write Algorithm in the Overall Flow



A *flash-write pulse* is the time during the flash-write operation between the setting and the clearing of the EXE bit (bit 0 of SEG_CTR). Charge is added to the bits of the flash memory array via the flash-write mechanism. The flash-write algorithm may require multiple flash-write pulses. The steps required to apply one flash-write pulse are outlined in Table 3–3.

Table 3–3. Steps for Applying One Flash-Write Pulse

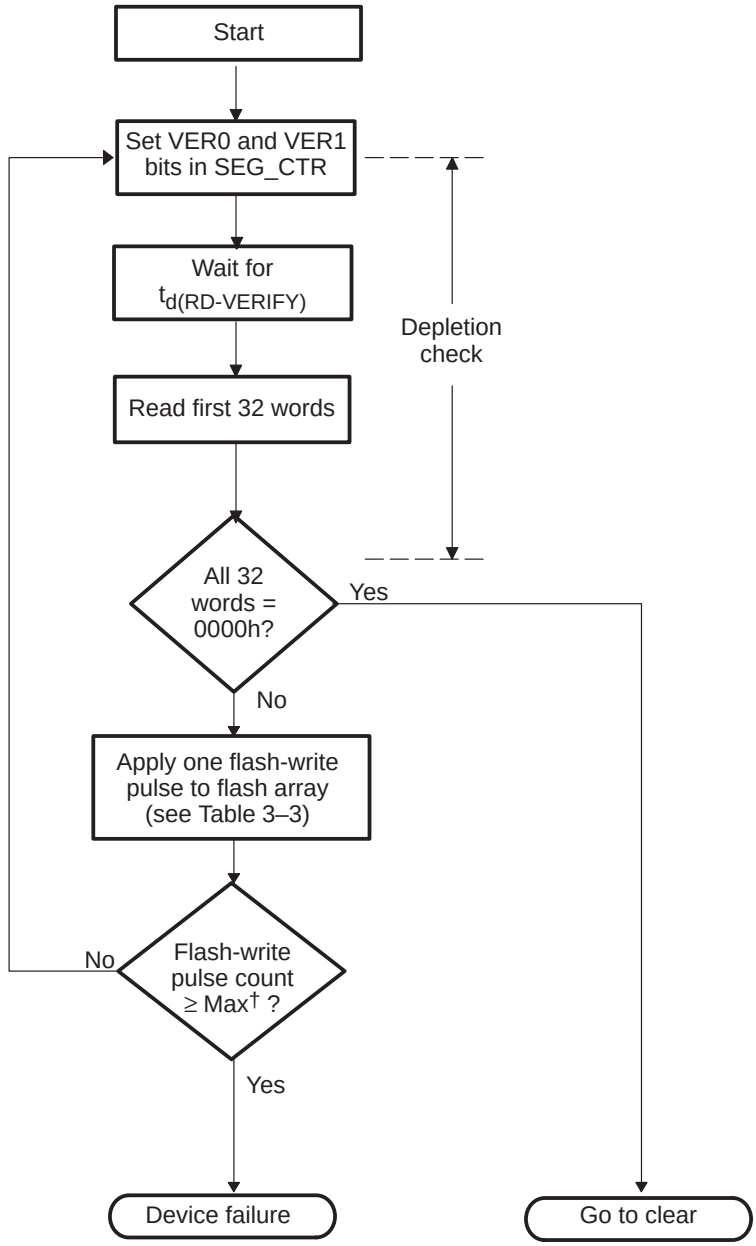
Steps	Action	Description
1	Power up the V_{CCP} pin.	Set the V_{CCP} pin to V_{DD}. If the V_{CCP} pin for the flash module to be recovered is not set to V_{DD}, then the flash-write operation will not be effective.
2	Activate the flash-write mode and enable all segments.	Set the WRITE/ERASE field to 10 and set SEG0–SEG7 in the SEG_CTR register. The flash module must be in register access mode (see section 2.2).
3	Wait for the internally generated supply voltage stabilization time.	The CPU executes a delay loop for the $t_{d(FLW\text{-}MODE}$†$_{)}$ time period.
4	Initiate the flash-write pulse.	Load the EXE, KEY1, and KEY0 bits with 1, 1, and 0, respectively. All three bits must be loaded in the same write cycle. The segment enable bits and WRITE/ERASE field must also be maintained.
5	Delay for the flash-write pulse time.	The CPU executes a delay loop for the $t_{d(FLW}$†$_{)}$ time period.
6	Terminate the flash-write pulse.	Clear all bits in the SEG_CTR register (load SEG_CTR with 0000h).
7	Delay for mode deselect time.	CPU executes a delay loop for the $t_{d(BUSY}$†$_{)}$ time period.

† See the device data sheet for the timing parameter values.

The flash-write algorithm consists of multiple iterations of a loop with one flash-write pulse applied in each iteration. At the beginning of each iteration, a depletion test is performed to determine if a flash-write pulse is required. Figure 3–7 shows the flow of the flash-write algorithm.

The flash-write operation uses the inverse-erase read mode to detect bits that are in depletion mode. For more information about the inverse-erase read mode, see section 2.4, *Read Modes*, on page 2-12.

Figure 3–7. Flash-Write Algorithm Flow



† See the device data sheet for the timing parameter values.

The CPU frequency range for the application is an important consideration for the depletion test, as well as for the program and erase operations. Because of the actual implementation of the flash memory circuitry, a bit in depletion mode is most easily detected at low frequency. Accordingly, if the application requires a variable CPU clock rate, the depletion test should be performed at the lowest frequency in the range. Only the read portion of the depletion test must be performed at the lower frequency, because it is the read that is used to detect depletion. The effective duration of the read operation can be extended by sequentially executing multiple reads on the same location. Because the same address is selected the entire time and internal control signals are maintained between reads, the final read is equivalent to a slow read. For example, if the DSP core is executing the programming algorithm at a CLKOUT rate of 20 MHz (50 ns), sequentially reading a location three times is equivalent to reading it once at 6.67 MHz (150 ns). The erase and flash-write algorithm implementations given in Appendix A use three reads to check for depletion.

Appendix A

Assembly Source Listings and Program Examples

The flash array is erased and programmed by code running on the DSP core. This code can originate from off-chip memory or can be loaded into on-chip RAM. The available flash programming tools for the ’F20x/F24x allow you to program the on-chip flash module without having knowledge or visibility of the algorithms. One scheme uses the scan emulation feature of the ’F20x/F24x to load the algorithms onto the DSP and control execution, and another scheme relies on boot loader code preprogrammed into the flash memory at the factory. You can find more information about these stand-alone flash programming tools on the Texas Instruments web page at http://www.ti.com. This appendix explains how to use the algorithm source files to program the flash module. You need this information to create new flash programming tools or to add such features as remote reprogrammability to a design.

Topic **Page**

A.1 Assembly Source for Algorithms

The algorithm source files implement the flows given in Chapter 3. Each algorithm is written as an assembly language subroutine, beginning with a label at an entry point and ending with a return instruction. The algorithms share a set of 16 relocatable variables for which pointers are defined in the header file, SVAR20.H.

The variables are defined at the beginning of B1 RAM, and an uninitialized section should be declared at link time to reserve this space. Also, the data page pointer (DP) should be initialized to point to this space before a call is made to any of the algorithms.

In addition to these variables, each algorithm references parameters that should be declared globally in the calling code. These parameters are listed in the introduction to each of the algorithm source files below.

The source files given are:

- SVAR20.H: header file that defines variables and constants
- SCLR20.ASM: clear algorithm
- SERA20.ASM: erase algorithm
- SFLW20.ASM: flash-write algorithm
- SPGM20.ASM: programming algorithm
- SUTILS20.ASM: subroutines common to all four algorithms

The same algorithm files can be used for the TMS320F206 and the TMS320F240/1/3 devices. A conditional assembly variable is provided in the header file, SVAR20.H, for assembling the algorithms for the correct device. For more details on this conditional assembly variable, see A.1.1.

A.1.1 Header File for Constants and Variables, SVAR20.H

This header file is included in each of the algorithm files using the .include directive. All of the constants used for flash programming are defined in this file. Also, the conditional assembly constant, F24x, is defined here to allow reuse of the algorithms for multiple device types. This constant should be modified to select the correct device when the algorithms are assembled. The SVAR20.H header file can also be included in the calling code, to allow visibility to the variable names.

```
*****************************************************************
** Variable declaration file                                   **
*                                                              **
*   TMS320F2XX Flash Utilities.                                **
*      Revision: 2.0, 9/10/97                                  **
*      Revision: 2.1, 1/31/98                                  **
*                                                              **
*   Filename: svar20.asm                                       **
*                                                              **
*Note:                                                         **
*DLOOP is a delay loop variable used in flash algorithms.     **
*This is a function of CLKOUT1. If the F206 device runs at    **
*any CLKOUT1 speed other than 20 MHz, DLOOP value should be   **
*redefined per the equation explained below. Use of           **
*current DLOOP for flash programming at speeds other than     **
*20 MHz is not recommended.                                   **
*****************************************************************
       .mmregs
BASE   .set   0300h            ;Base address for variables
                               ;can be changed to relocate
                               ;variable space in RAM.
BASE_0    .set      BASE+0     ;Scratch pad registers.
BASE_1    .set      BASE+1     ;
BASE_2    .set      BASE+2     ;
BASE_3    .set      BASE+3     ;
BASE_4    .set      BASE+4     ;
BASE_5    .set      BASE+5     ;
BASE_6    .set      BASE+6     ;
SPAD1     .set      BASE+7     ;
SPAD2     .set      BASE+8     ;
FL_ADRS   .set      BASE+10    ;Flash load address.
FL_DATA   .set      BASE+11    ;Flash load data.
ERROR     .set      BASE+15    ;Error flag register.
*Variables for ERASE and CLEAR
RPG_CNT   .set      BASE+12    ;Program pulse count.
FL_ST     .set      BASE+13    ;Flash start addr/Seg Cntrl Reg.
FL_END    .set      BASE+14    ;Flash end address.
*
```

```
*CONSTANTS
*
*********************************************************
*Conditional assembly variable for F24X vs F206.        *
*If F24X = 1, then assemble for F24X; otherwise,        *
*assemble for F206.                                     *
*********************************************************
F24X      .set      0            ;Assemble for F206
;F24X     .set      1            ;Assemble for F24X
*********************************************
*  Delay variables for CLEAR,ERASE and PROGRAM*
*********************************************
D5        .set      0            ;5 us delay
D10       .set      1            ;10 us delay
D100      .set      19           ;100 us delay
D5K       .set      999          ;5 ms delay
D7K       .set      1399         ;7 ms delay
************************************************************
*DLOOP constant proportional to CLKOUT1                    *
*Calculate DLOOP in decimal using the following equation:  *
*  DLOOP=FLOOR{(5us/tCLKOUT1)-6};                          *
*Examples                                                  *
*a.@ 15 MHz, DLOOP= 69;                                    *
*b.@ 9.8304 MHz, DLOOP= 43;                                *
*c.@ 16.384 MHz, DLOOP= 75;                                *
************************************************************
;DLOOP    .set   14        ;5-us delay loop @ 4.032 MIPs
;DLOOP    .set   19        ;5-us delay loop @ 5 MIPs
;DLOOP    .set   44        ;5-us delay loop @ 10 MIPs
;DLOOP    .set   75        ;5-us delay loop @ 16.384 MIPs
;DLOOP    .set   94        ;5-us delay loop @ 20 MIPs
*************************
* On-chip I/O registers *
*************************
F_ACCESS0 .set   0FFE0h ;F206 ACCESS CNTRL REGISTER 0.
F_ACCESS1 .set   0FFE1h ;F206 ACCESS CNTRL REGISTER 1.
PMST      .set   0FFE4h ;Defines SARAM in PM/DM and MP/MC bit.
F24X_ACCS .set   0FF0Fh ;F240 ACCESS CNTRL REGISTER.
;-----------------------------------------
;Register Declarations for F240 Peripherals |
;-----------------------------------------
;Watch-Dog(WD)/Real Time Int(RTI)/Phase-Locked Loop (PLL)
;Registers
;~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
RTI_CNTR  .set   07021h    ;RTI Counter reg
WD_CNTR   .set   07023h    ;WD Counter reg
WD_KEY    .set   07025h    ;WD Key reg
RTI_CNTL  .set   07027h    ;RTI Control reg
WD_CNTL   .set   07029h    ;WD Control reg
PLL_CNTL1 .set   0702Bh    ;PLL control reg 1
PLL_CNTL2 .set   0702Dh    ;PLL control reg 2
```

A.1.2 Clear Algorithm, SCLR20.ASM

This code is an implementation of the clear (programming) algorithm described in section 3.2 on page 3-4. Recall that the clear algorithm is identical to the programming algorithm with the data forced to 0000h for all flash addresses.

Memory section: **fl_clr**

Entry point: **GCLR**

Parameters to be declared and initialized by the calling code are:

- ❑ PROTECT defines the values of bits 8–15 of SEG_CTR during the clear algorithm.
- ❑ SEG_ST defines the start address of the flash array to be cleared.
- ❑ SEG_END defines the end address of the flash array to be cleared.

Return value: **ERROR** (@BASE+15); 0 = Pass, 1 = Fail

```
*************************************************************
** CLEAR Subroutine                                        **
*                                                          **
*  TMS320F2XX Flash Utilities.                             **
*     Revision: 2.0, 9/10/97                               **
*     Revision: 2.1, 1/31/98                               **
*                                                          **
*  Filename: sclr20.asm                                    **
*                                                          **
*  Called by: c2xx_bcx.asm or flash application programs.  **
*                                                          **
*  !!CAUTION – INITIALIZE DP BEFORE CALLING THIS ROUTINE!! **
*                                                          **
*  Function: Clears one or more contiguous segments of     **
*            array 0/1 as specified by the following       **
*            variables.                                    **
*            SEG_ST    Segment start address               **
*            SEG_END   Segment end address                 **
*            PROTECT   Sector protect enable               **
*                                                          *
*  The algorithm used is ”row-horizontal”, which means that *
*  an entire flash row (32 words) is programmed in parallel.*
*  This method provides better uniformity of programming   *
*  levels between adjacent bits than if each address were  *
*  programmed independently. The algorithm also uses a     *
*  3-read check for VER0 margin (i.e.,the flash location is *
*  read three times and the first two values are discarded.)*
*  This provides low–frequency read–back margin on         *
```

```
*   programmed bits. For example, if the flash is programmed *
*   using a CLKOUT period of 50 ns, the flash can be read back
*   reliably over the CLKOUT period range of 50 ns to 150 ns *
*   (6.67 MHz–20 MHz). The programming pulse-duration is     *
*   100 us, and a maximum of 150 pulses is applied per row.  *
*                                                            *
*   The following resources are used for temporary storage:  *
*         AR0    Used for comparisons                        *
*         AR1    Used for pgm pulse count                    *
*         AR2    Used for row banz loop.                     *
*         AR6    Parameter passed to Delay                   *
*      FL_ADRS   Used for flash address                      *
*      FL_DATA   Used for flash data.                        *
*      FL_ST     Used for flash start address                *
*      BASE_0    Used for row–done flag                      *
*      BASE_1    Used for row start address                  *
*      SPAD1     Flash commands                              *
*      SPAD2     Flash commands                              *
*                                                            **
*************************************************************
          .include "svar20.h"
*
MAX_PGM   .set      150    ;Only allow 150 pulses per row.
VER0      .set      010h   ;VER0 command.
WR_CMND   .set      4      ;Write command.
WR_EXE    .set      045h   ;Write EXEBIN command.
STOP      .set      0      ;Reset command.
          .def      GCLR
          .ref      PROTECT,SEG_ST,SEG_END
          .ref      DELAY,REGS,ARRAY
       .sect     "fl_clr"
*************************************************************
*   GCLR:  This routine performs a clear operation on the    *
*   flash array defined by the FL_ST variable. The segments  *
*   to be cleared are defined by the SEG_ST, SEG_END, and    *
*   PROTECT variables.                                       *
*   The following resources are used for temp storage:       *
*   AR0       Used for comparisons                           *
*   AR1       Used for pgm pulse count                       *
*   AR2       Used for row banz loop                         *
*   FL_ADRS   Used for flash address                         *
*   FL_DATA   Used for flash data                            *
*   BASE_0    Used for row–done flag                         *
*   BASE_1    Used for row start address                     *
*   BASE_2    Used for byte mask.                            *
*************************************************************
GCLR:
       SETC      INTM          ;Disable all ints.
       CLRC      SXM           ;Disable sign extension.
       SPLK      #0,ERROR      ;Reset error flag
       LACL      SEG_ST        ;Get segment start address.
       SACL      FL_ADRS       ;Save as current address.
       AND       #04000h       ;Get array start address.
```

```
       SACL      FL_ST         ;Save array start address.
       LACL      FL_ADRS       ;Get segment start address.
NEWROW           ;********Begin a new row.*
       SACL      BASE_1        ;Save row start address.
       LAR       AR1,#0        ;Init pulse count to zero.
SAMEROW          ;********Same row, next pulse.*
       SPLK      #1,BASE_0     ;Set row done flag = 1(True).
       LACL      BASE_1        ;Get row start address.
       SACL      FL_ADRS       ;Save as current address.
       LAR       AR2,#31       ;Init row index.
********Repeat the following code 32 times until end of row.*
LOBYTE           ;********First, do low byte.*
       SPLK      #0FFh,BASE_2     ;Get lo–byte mask.
       CALL      PRG_BYTE         ;Check/Program lo–byte
       SPLK      #0FF00h,BASE_2   ;Get hi–byte mask.
       CALL      PRG_BYTE         ;Check/Program hi–byte.
NEXTWORD         ;********Next word in row.
       LACL      FL_ADRS       ;Load address for next word.
       ADD       #1            ;Increment address.
       SACL      FL_ADRS       ;Save as current address.
       MAR       *,AR2         ;Point to row index.
       BANZ      LOBYTE        ;Do next word,and dec AR2.
********Reached end of row. Check if row done.*
       BIT    BASE_0,15        ;Get row_done flag.
       BCND   ROW_DONE,TC      ;If 1, then row is done.
       MAR    *,AR1            ;Else, row is not done, so
       MAR    *+               ;inc row pulse count.
       LAR    AR0,#MAX_PGM     ;Check if passed allowable max.
       CMPR   2                ;If AR1>MAX_PGM, then
       BCND   EXIT,TC          ;fail, don't continue.
       B      SAMEROW          ;else, go to beginning
                               ;of same row.
********If row done, then check if Array done.*
ROW_DONE                       ;Check if end of array.
       SUB    SEG_END          ;Subtract segment end address.
       BCND   DONE,GEQ         ;If >0, then done.
********Else, go to next row.*
       LACL   FL_ADRS          ;Get current address.
       B      NEWROW           ;Start new row.
********If here, then done.
DONE   CALL   ARRAY            ;Access flash in array mode.
       RET
********If here, then unit failed to program.*
EXIT   SPLK   #1,ERROR         ;Update error flag.
       B      DONE             ;Get outa here.
   .page
*************************************************************
*  THIS SECTION PROGRAMS THE VALUE STORED IN FL_DATA INTO   *
*  THE FLASH ADDRESS DEFINED BY FL_ADRS.                    *
*                                                           *
*  The following resources are used for temporary storage:  *
*            AR6    Parameter passed to Delay.             *
*            SPAD1  Flash program and STOP commands.       *
```

```
*             SPAD2  Flash program + EXE command.              *
*************************************************************
EXE_PGM                                                       ;*
   CALL   ARRAY            ;ACCESS ARRAY                      *
*LOAD WADRS AND WDATA                                         **
      LACL      FL_ADRS              ;ACC => PROGRAM ADRS     *
      TBLW      FL_DATA              ;LOAD WADRS AND WDATA    *
   CALL   REGS      ;ACCESS FLASH REGS                        *
*SET UP WRITE COMMAND WORDS                                   **
      LACL      PROTECT          ;GET SEGMENT PROTECT MASK   **
      OR        #WR_CMND         ;OR IN WRITE COMMAND        **
      SACL      SPAD1            ;SPAD1 = WRITE COMMAND      **
      OR        #WR_EXE          ;OR IN EXEBIN COMMAND       **
      SACL      SPAD2            ;SPAD2 = WRITE EXE COMMAND  **
*                                                             *
      LACL      FL_ST            ;ACC => 0 (FLASH0)          *
*   ACTIVATE WRITE BIT                                        **
      TBLW      SPAD1            ;EXECUTE COMMAND            **
      LAR       AR6,#D10         ;SET DELAY                  **
   CALL   DELAY,*,AR6   ;WAIT                                 **
*   SET EXEBIN BIT                                           **
      TBLW      SPAD2            ;EXECUTE COMMAND            **
      LAR       AR6,#D100        ;SET DELAY                  **
   CALL   DELAY,*,AR6   ;WAIT                                 **
*   STOP WRITE OPERATION                                      *
      SPLK      #0,SPAD1         ;SHUTDOWN WRITE OPERATION   *
      TBLW      SPAD1            ;EXECUTE COMMAND            *
      LAR       AR6,#D10         ;SET DELAY                  *
   CALL   DELAY,*,AR6   ;WAIT                                 *
*                                                             *
      RET                        ;RETURN TO CALLING SEQUENCE*
*************************************************************
   .page
*************************************************************
*   ACTIVATE VER0 ON FLASH READS                              *
*   LOADS FLASH WORD AT ADDR FL_ADRS TO FL_DATA.              *
*   Uses SPAD1 for temporary storage of flash commands.       *
*************************************************************
SET_RD_VER0                                                   ;*
      CALL      REGS               ;ACCESS FLASH REGISTERS *
      LACL      FL_ST              ;ACC => FLASH           *
      SPLK      #VER0,SPAD1        ;ACTIVATE VER0          *
      TBLW      SPAD1              ;EXECUTE COMMAND*
      LAR       AR6,#D10           ;SET DELAY              *
      CALL      DELAY,*,AR6      ;WAIT                     *
      CALL      ARRAY            ;ACCESS FLASH ARRAY       *
      LACL      FL_ADRS          ;POINT TO ADRS            *
      TBLR      FL_DATA          ;GET FLASH WORD 1x read   *
      TBLR      FL_DATA          ;2x read                  *
      TBLR      FL_DATA          ;3x read                  *
      CALL      REGS             ;ACCESS FLASH REGISTERS   *
      LACL      FL_ST            ;ACC => FLASH             *
      SPLK      #STOP,SPAD1      ;DEACTIVATE VER0          *
```

```
       TBLW      SPAD1           ;EXECUTE COMMAND          *
       LAR       AR6,#D10        ;SET DELAY                *
       CALL      DELAY,*,AR6     ;WAIT                     *
       CALL      ARRAY           ;ACCESS FLASH ARRAY       *
       RET                       ;RETURN TO CALLING SEQUENCE*
*************************************************************
***********************************************
*  PRG_BYTE: Programs hi or lo byte depending on  *
*            byte mask (BASE_2).                  *
***********************************************
PRG_BYTE:
       CALL      SET_RD_VER0  ;Read word at VER0 level.
       LACL      BASE_2       ;Get lo/hi byte mask.
       AND       FL_DATA      ;Xor with read-back value.
       BCND      PB_DONE,EQ   ;If zero, then done.
       XOR       #0FFFFh      ;else, mask off good bits.
       SACL      FL_DATA      ;New data.
       CALL      EXE_PGM      ;PGM Pulse.
       SPLK      #0,BASE_0    ;Set row done flag = 0(False).
PB_DONE RET
***********************************************
       .end
```

A.1.3 Erase Algorithm, SERA20.ASM

This code is an implementation of the erase algorithm described in section 3.3 on page 3-10.

Memory section: **fl_ers**

Entry point: **GERS**

Parameters to be declared and initialized by the calling code are:

- ❑ PROTECT defines the values of bits 8–15 of SEG_CTR during the erase algorithm.
- ❑ SEG_ST defines the start address of the flash array to be erased.
- ❑ SEG_END defines the end address of the flash array to be erased.

Return value: **ERROR** (@BASE+15); 0 = Pass, 1 = Fail

```
*************************************************************
*  ERASE subroutine                                         **
*                                                           **
*  TMS320F2XX Flash Utilities.                              **
*     Revision: 2.0, 9/10/97                                **
*     Revision: 2.1, 1/31/98                                **
*                                                           **
*  Filename: sera20.asm                                     **
*                                                           **
*  Called by: c2xx_bex.asm or flash application programs.   **
*                                                           **
*  !!CAUTION – INITIALIZE DP BEFORE CALLING THIS ROUTINE!!  **
*                                                           **
*  Function: Erases one or more contiguous segments of      **
*            flash array 0/1 as specified by the            **
*            following variables.                           **
*            SEG_ST    Segment start address                **
*            SEG_END   Segment end address                  **
*            PROTECT   Sector protect enable                **
*                                                           **
*  The algorithm used is XOR–VER1, which means that in      **
*  addition to the VER1 read mode, an XOR readback is used  **
*  to gain more margin. During the read portion of the      **
*  erase, two reads are performed for each address; for the **
*  first read, all address bits are complemented using a    **
*  logical XOR with the array end address. The data read    **
*  during the first read is discarded, and the second read  **
*  is performed on the actual address. This scheme          **
*  simulates the worst–case branching condition for code    **
*  executing from the flash array.                          **
```

```
*   The erase pulse duration is 7ms, and a maximum of        **
*   1000 pulses is applied to the array.                     **
*                                                            **
*   The following resources are used for temporary storage:  **
*      AR0       Used for comparisons                        **
*      AR1       Used for erase pulse count                  **
*      AR2       Used for main banz loop                     **
*      AR6       Parameter passed to DELAY                   **
*      BASE_0    Parameter passed to Set_mode                **
*      BASE_1    Used for flash address.                     **
*      BASE_2    Used for flash data                         **
*      BASE_3    Used for flash checksum                     **
*      BASE_4    Used for segment size                       **
*      BASE_5    Flash Erase command                         **
*      BASE_6    Flash Erase+EXE command                     **
************************************************************
          .include "svar20.h" ;defines variables for flash0
                              ;or for flash1 array
*
MAX_ER    .set   1000      ;Allow only 1000 erase pulses.
VER1      .set   8         ;VER1 command.
ER_CMND   .set   2         ;ERASE COMMAND WORD
ER_EXE    .set   043h      ;ERASE EXEBIN COMMAND WORD
INV_ER    .set   018h      ;INVERSE ERASE COMMAND WORD
FL_WR     .set   6         ;FLASH WRITE COMMAND WORD
FLWR_EX   .set   047h      ;FLASH WRITE EXEBIN COMMAND WORD
STOP      .set   0         ;RESET REGISTER COMMAND WORD
          .def   GERS
          .ref   PROTECT,SEG_ST,SEG_END
          .ref   DELAY,REGS,ARRAY
          .sect  "fl_ers"
************************************************************
*   GERS: This routine performs an erase to                *
*   xorver1 level. The Seg to erase is defined by          *
*   the vars SEG_ST and SEG_END. The following             *
*   resources are used for temporary storage:              *
*      AR0       Used for comparisons                      *
*      AR1       Used for erase pulse count                *
*      AR2       Used for main banz loop                   *
*      BASE_0    Parameter passed to Set_mode              *
*      BASE_1    Used for flash address.                   *
*      BASE_2    Used for flash data                       *
*      BASE_3    Used for flash checksum                   *
*      BASE_4    Used for segment size                     *
************************************************************
GERS:
************************************************************
*   Code initialization section                            *
*   Initialize test loop counters:                         *
*      AR1 is the number of ERASE pulses.                  *
************************************************************
       SETC   INTM         ;Disable all maskable ints.
       SETC   SXM          ;Enable sign extension.
```

```
        CLRC   OVM           ;Disable overflow mode.
        LACL   SEG_ST        ;Get segment start address.
        AND    #04000h       ;Get array start address.
        SACL   FL_ST         ;Save array start address.
        OR     #03FFFh       ;Get array end address.
        SACL   FL_END        ;Save array end address.
        SPLK   #0,ERROR      ;Reset error flag
        LAR    AR1,#0        ;Set erase count to 0.
        SPLK   #STOP, BASE_0 ;Stop command.
        CALL   SET_MODE      ;Disable any flash cmds.
XOR_ERASE
** Compute checksum for flash, using address complementing.**
        LACC   SEG_END
        SUB    SEG_ST
        SAC    BASE_4            ;Segment length–1.
        LAR    AR2,BASE_4        ;load n–1 to loop n times.
        ADD    #1
        SACL   BASE_4            ;Segment length.
        SPLK   #VER1,BASE_0      ;VER1 command.
        CALL   SET_MODE          ;Set VER1 mode.
        MAR    *,AR2
        BLDD   #SEG_ST,BASE_1    ;Segment start address.
        SPLK   #0,BASE_3         ;Clear checksum.
RD1_LOOP                         ;For I = SEG_ST to SEG_END.
        LACC   BASE_1            ;ACC => CURRENT ADDR.
        XOR    FL_END            ;XOR addr with flash end addr.
        TBLR   BASE_2            ;Dummy Read.
        LACC   BASE_1            ;Get actual addr again.
        TBLR   BASE_2            ;True Read.
        ADD    #1                ;Increment flash addr.
        SACL   BASE_1            ;Store for next read.
        LACC   BASE_3            ;Get old check sum.
        ADD    BASE_2            ;ACC=>ACC+FL_DATA.
        SACL   BASE_3            ;Save new check sum.
        BANZ   RD1_LOOP,*–
        ADD    BASE_4            ;Should make ACC = 0 for
                                 ;erased array.
        BCND   XOR_ERFIN,EQ      ;If BASE_3 = 0, finished.

***** If not erased, apply an erase pulse.
        CALL   ERASE_A           ;Else, pulse it again.
        MAR    *,AR1             ;ARP–>AR1 (Erase pulse count)
        MAR    *+                ;Increment Erase count.
        LAR    AR0,#MAX_ER
        CMPR2                    ;If AR1>MAX_ER then
        BCND   EXIT,TC           ;fail, don't continue erasing.
        B      XOR_ERASE         ;Else, check again.
***** If here, then erase passed; now check for depletion.
XOR_ERFIN
        SPLK   #STOP, BASE_0        ;Stop command.
        CALL   SET_MODE             ;Disable any flash cmds.
        CALL   INV_ERASE            ;Check for depletion.
DONE    RET                         ;Return to calling code.
```

```
***** If here, then an error has occurred.
EXIT   SPLK   #1,ERROR            ;Update error flag
       SPLK   #STOP,BASE_0        ;Stop command.
       CALL   SET_MODE            ;Disable any flash cmds.
       B      DONE                ;Get outa here.
*************************************************************
       .page
************************************************
*  SET_MODE: This routine sets the flash in the   *
*  mode specified by the contents of BASE_0. This *
*  can be used for VER0,VER1,INVERASE, or STOP.   *
*         AR6: Parameter passed to DELAY.         *
************************************************
SET_MODE
       CALL   REGS          ;ACCESS FLASH REGS
       LACL   FL_ST         ;ACC => SEG_CTR.
       TBLW   BASE_0        ;Activate MODE.
       LAR    AR6,#D10      ;SET DELAY
       CALL   DELAY,*,AR6   ;WAIT                    *
       CALL   ARRAY         ;ACCESS FLASH ARRAY      *
       RET
************************************************
*  INV_ERASE: This routine is used to check for   *
*  depletion in the flash array.                  *
*      AR2       Used for main banz loop       *
*      BASE_0    Parameter passed to Set_mode     *
*      BASE_1    Used for flash address           *
*      BASE_2    Used for flash data           *
************************************************
INV_ERASE
       SPLK   #INV_ER,BASE_0
       CALL   SET_MODE      ;Set inverse–erase mode.
       BLDD   #FL_ST,BASE_1 ;Array start address.
       LAR    AR2,#31       ;Loop count.
       MAR    *,AR2
NEXT_IVERS
       LACL   BASE_1        ;Get address.
       TBLR   BASE_2        ;Dummy read.
       TBLR   BASE_2        ;Read data.
       ADD    #1            ;Increment address.
       SACL   BASE_1        ;Save address.
       ZAC
       ADD    BASE_2        ;Add data.
       BCND   EXIT,NEQ      ;If ACC<>0, then fail.
*Else continue, until until done with row.
       BANZ   NEXT_IVERS    ;Loop 32 times.
       SPLK   #STOP,BASE_0  ;Stop command.
       CALL   SET_MODE      ;Disable any flash cmds.
       RET                  ;If here then test passed.
       .page
*************************************************************
* ERASE_A: This subroutine applies one erase pulse to the   *
* flash array.                                              *
```

```
*                                                              *
* The following resources are used for temporary storage:     *
*      BASE_0    Flash STOP command, and FFFF for WDATA.      *
*      BASE_5    Flash erase command.                         *
*      BASE_6    Flash erase + EXE command.                   *
****************************************************************
ERASE_A
*   SET UP FLASH ERASE COMMANDS FOR PROTECT MASK.              **
       LACL   PROTECT          ;GET SEGMENT PROTECT MASK       **
       OR     #ER_CMND         ;OR IN ERASE COMMAND            **
       SACL   BASE_5           ;BASE_5 = ERASE COMMAND         **
       OR     #ER_EXE          ;OR IN EXEBIN COMMAND           **
       SACL   BASE_6           ;BASE_6 = ERASE EXE COMMAND     **
*                                                              **
*   MUST LOAD WDATA WITH FFFF.                                 **
       SPLK   #0FFFFh,BASE_0   ;WDATA VALUE FOR ERASE          **
       LACC   FL_ST            ;ACC => FLASH                   **
       TBLW   BASE_0           ;SET WDATA = FFFF               **
*                                                              **
*   THIS SECTION ACTIVATES THE WRITE COMMAND.                  **
       SPLK   #STOP,BASE_0     ;Stop command.                  **
       CALL   SET_MODE         ;Disable any flash cmds.        **
   CALL   REGS             ;ACCESS FLASH REGS                 **
       LACC   FL_ST            ;ACC => FLASH                   **
       TBLW   BASE_5           ;ACTIVATE ERASE                 **
       LAR    AR6,#D10         ;SET DELAY                      **
   CALL   DELAY,*,AR6      ;WAIT                              **
*                                                              **
*   THIS SECTION ACTIVATES THE EXEBIN COMMAND.                 **
       TBLW   BASE_6           ;START ERASURE                  **
       LAR    AR6,#D7K         ;SET DELAY to 7 ms              **
   CALL   DELAY,*,AR6      ;WAIT                              **
       SPLK   #STOP,BASE_0     ;STOP COMMAND                   **
       CALL   SET_MODE         ;STOP ERASE                     **
       RET                     ;RETURN TO CALLING CODE         **
****************************************************************
   .end
```

A.1.4 Flash-Write Algorithm, SFLW20.ASM

This code is an implementation of the flash-write algorithm described in section 3.4 on page 3-14.

Memory section: **fl_wrt**

Entry point: **FLWS**

Parameters to be declared and initialized by the calling code are:

- ❏ PROTECT defines the values of bits 8–15 of SEG_CTR during the flash-write algorithm.
- ❏ SEG_ST defines the start address of the flash array to be recovered.
- ❏ SEG_END defines the end address of the flash array to be recovered.

Return value: **ERROR** (@BASE+15) 0=Pass, 1=Fail

```
*************************************************************
** FLASH–WRITE subroutine                                  **
*                                                          **
*  TMS320F2XX Flash Utilities.                             **
*  Revision: 2.0, 9/10/97                                  **
*  Revision: 2.1, 1/31/98                                  **
*                                                          **
*  Filename: sflw20.asm                                    **
*                                                          **
*  Called by : c2xx_bfx.asm or flash application programs. **
*                                                          **
*  !!CAUTION – INITIALIZE DP BEFORE CALLING THIS ROUTINE!! **
*                                                          **
*  Function: Performs flash writes on flash array 0/1 as   **
*            specified by the following vars:              **
*            SEG_ST    Array segment start address         **
*            PROTECT   Sector protect enable               **
*                                                          *
*  The flash–write pulse duration used is 14 ms, and a     *
*  maximum of 10000 pulses is applied until the device     *
*  passes the depletion test.                              *
*                                                          *
*  The following resources are used for temp storage:      *
*     AR0       Used for comparison                        *
*     AR1       Flash–Write Pulse Count                    *
*     AR2       Used for main BANZ loop                    *
*     AR6       Parameter passed to DELAY                  *
*     BASE_0    Parameter passed to SET_MODE               *
*     BASE_1    Used for flash address                     *
*     BASE_2    Used for flash data                        *
```

```
*      BASE_3    Used for EXE + flw cmd                    *
*****************************************************************
       .include "svar20.h"     ;defines variables for flash0
                               ;or for flash1 array
*
MAX_FLW   .set   10000         ;Allow only 10000 flw pulses.
INV_ER    .set   018h          ;INVERSE ERASE COMMAND WORD
FLWR      .set   6             ;FLASH WRITE COMMAND WORD
FLWR_EX   .set   047h          ;FLASH WRITE EXEBIN COMMAND WORD
STOP      .set   0             ;RESET REGISTER COMMAND WORD
          .def   FLWS
          .ref   PROTECT,SEG_ST,SEG_END
          .ref   DELAY,REGS,ARRAY
       .sect "fl_wrt"
*******************************************************
*  FLWS: This routine is used to check for bits       *
*  in depletion mode. If any are found, flash-        *
*  write is used to recover.                          *
*      AR1    Flash-write pulse count.                *
*      AR2    Used for main banz loop.                *
*      BASE_0 Parameter passed to Set_mode.           *
*      BASE_1 Used for flash address.                 *
*      BASE_2 Used for flash data.                    *
*******************************************************
FLWS:
*******************************************************
*  Code initialization section                        *
*  Initialize test loop counters:                     *
*      AR1 is the number of flash-write pulses.       *
*******************************************************
       SETC   INTM             ;Disable maskable ints.
       LACL   SEG_ST           ;Get segment start address.
       AND    #04000h          ;Get array start address.
       SACL   FL_ST            ;Save array start address.
       SPLK   #0,ERROR         ;Reset error flag.
       LAR    AR1,#0           ;Set FLW count to 0.
       SPLK   #STOP,BASE_0     ;Flash STOP command.
       CALL   SET_MODE         ;Disable any flash commands.
INV_ERASE
       SPLK   #INV_ER,BASE_0
       CALL   SET_MODE         ;Set inverse-erase mode.
       BLDD   #FL_ST,BASE_1    ;Array start address.
       LAR    AR2,#31          ;Loop count.
       MAR    *,AR2
NEXT_IVERS
       LACL   BASE_1       ;Get address.
       TBLR   BASE_2       ;Dummy read.
       TBLR   BASE_2       ;Dummy read.
       TBLR   BASE_2       ;Read data.
       ADD    #1           ;Increment address.
       SACL   BASE_1       ;Save address.
       ZAC
       ADD  BASE_2         ;Add data.
```

```
       BCND   FL_WRITE, NEQ ;If ACC<>0, then flwrite.
*Else, continue until until done with row.
       BANZ   NEXT_IVERS    ;Loop 32 times.
       SPLK   #STOP,BASE_0  ;Flash STOP command.
       CALL   SET_MODE      ;Disable flash commands.
                            ;If here then test passed.
DONE   RET                  ;Return to calling code.
* If here, then an error has occurred.
EXIT   SPLK   #1,ERROR       ;Update error flag
       SPLK   #STOP,BASE_0  ;Flash STOP command.
       CALL   SET_MODE      ;Disable flash commands.
       CALL   ARRAY         ;ACCESS FLASH ARRAY
       B      DONE          ;Get outa here.
       .page
************************************************
*  FL_WRITE: This routine performs a fl_write on  *
*  the flash until a maximum is reached. The      *
*  array is defined by the variable FL_ST         *
*  and the segment(s) is defined by the PROTECT   *
*  mask. The following resources are used for     *
*  temporary storage:                             *
*         AR0    Used for comparison              *
*         AR1    Used for pulse count (Global)    *
*         AR6    Parameter passed to DELAY        *
*      BASE_0    Parameter passed to SET_MODE     *
*      BASE_2    Used for flw cmd                 *
*      BASE_3    Used for EXE + flw cmd           *
************************************************
FL_WRITE
       SPLK   #STOP,BASE_0  ;Flash STOP command.
       CALL   SET_MODE      ;Disable flash commands.
       LACL   PROTECT       ;Get sector_prot mask.
       OR     #FLWR         ;Or in fl_write cmd.
       SACL   BASE_2        ;BASE_2 = fl_write cmd.
       OR     #FLWR_EX      ;Or in EXE + fl_write cmd.
       SACL   BASE_3        ;BASE_3 = EXE + fl_write cmd.
*Set the flash-write command.
       CALL   REGS          ;Access flash regs.
       LACC   FL_ST         ;ACC => SEG_CTL.
       TBLW   BASE_2        ;Initiate fl_write.
       LAR    AR6,#D10      ;Set delay.
       CALL   DELAY,*,AR6   ;Wait,10US flw stabilization time.
*Set the EXE bit (start flash-write pulse).
       TBLW   BASE_3        ;Start flw pulse.
       LAR    AR6,#D7K      ;Set delay to 7 ms.
       CALL   DELAY,*,AR6   ;WAIT,7 ms.
       LAR    AR6,#D7K      ;Set delay to 7 ms.
       CALL   DELAY,*,AR6   ;WAIT 7 ms.
*A 14-mS flash write pulse has been applied.
       SPLK   #STOP,BASE_0  ;Flash STOP command.
       CALL   SET_MODE      ;Disable flash commands.
       MAR    *,AR1
       MAR    *+            ;Increment flw count.
```

```
       LAR    AR0,#MAX_FLW
       CMPR   2             ;If AR1>MAX_FLW then
       BCND   EXIT,TC       ;Fail, don't continue recovery.
       B      INV_ERASE     ;Else, perform iverase again.
************************************************
*  SET_MODE: This routine sets the flash in the   *
*  mode specified by the contents of BASE_0. This *
*  can be used for VER0,VER1,INVERASE,or STOP.    *
************************************************
SET_MODE
       CALL   REGS          ;ACCESS FLASH REGS
       LACL   FL_ST         ;ACC => SEG_CTR.
       TBLW   BASE_0        ;Activate MODE.
       LAR    AR6,#D10      ;SET DELAY
       CALL   DELAY,*,AR6   ;WAIT
       CALL   ARRAY         ;ACCESS FLASH ARRAY
   RET
************************************************
   .end
```

A.1.5 Programming Algorithm, SPGM20.ASM

This code is an implementation of the program algorithm described in section 3.2 on page 3-4.

Memory section: **fl_prg**

Entry point: **GPGMJ**

Parameters to be declared and initialized by the calling code are:

- ❏ PRG_bufaddr defines the destination start address.
- ❏ PRG_length defines the source buffer length.
- ❏ PRG_paddr defines the source buffer start address (data space).
- ❏ PROTECT defines the values of bits 8–15 of SEG_CTR during the programming algorithm.

Return value: **ERROR** (@BASE+15); 0 = Pass, 1 = Fail

```
*************************************************************
** PROGRAM Subroutine                                      **
*                                                          **
*  TMS320F2XX Flash Utilities.                             **
*     Revision: 2.0, 9/10/97                               **
*     Revision: 2.0b, 12/5/97                              **
*     Revision: 2.1, 1/31/98                               **
*                                                          **
*  Filename: spgm20.asm                                    **
*                                                          **
*  Called by: c2xx_bpx.asm or flash application programs.  **
*                                                          **
*  !!CAUTION – INITIALIZE DP BEFORE CALLING THIS ROUTINE!! **
*                                                          **
*  Function: This routine programs all or part of the      **
*            flash as specified by the variables:          **
*            PRG_paddr    Destination start address        *
*            PRG_length   Source buffer length             *
*            PRG_bufaddr  Source buffer start address      *
*                                                          *
*  The algorithm used is ”row–horizontal”, which means that *
*  an entire flash row (32 words) is programmed in parallel.*
*  This method provides better uniformity of programming   *
*  levels between adjacent bits than if each address were  *
*  programmed independently. The algorithm also uses a     *
*  3–read check for VER0 margin (i.e., the flash location is*
*  read three times and the first two values are discarded.)*
*  This provides low–freq read–back margin on programmed   *
```

```
*   bits. For example, if the flash is programmed using a    *
*   CLKOUT period of 50 ns, the flash can be reliably read   *
*   back over the CLKOUT period range of 50 ns to 150 ns     *
*   (6.67MHz–20 MHz). The programming pulse duration is      *
*   100 us, and a maximum of 150 pulses is applied per row.  *
*                                                            *
*   The following variables are used for temp storage:       *
*      AR0       Used for comparisons                        *
*      AR1       Used for pgm pulse count                    *
*      AR2       Used for row banz loop                      *
*      AR3       Used for buffer addr index                  *
*      AR4       Used for flash address.                     *
*      AR6       Parameter passed to Delay                   *
*      SPAD1     Flash program and STOP commands             *
*      SPAD2     Flash program + EXE command                 *
*      FL_ADRS   Used for flash address                      *
*      FL_DATA   Used for flash data                         *
*      BASE_0    Used for row–done flag                      *
*      BASE_1    Used for row start address                  *
*      BASE_2    Used for row length–1                       *
*      BASE_3    Used for buffer/row start addr              *
*      BASE_4    Used for destination end addr               *
*      BASE_5    Used for byte mask                          *
************************************************************
       .include  ”svar20.h”
*
MAX_PGM   .set   150    ;Allow only 150 pulses per row.
VER0      .set   010h   ;VER0 command.
WR_CMND   .set   4      ;Write command.
WR_EXE    .set   045h   ;Write EXEBIN command.
STOP      .set   0      ;Reset command.
          .def   GPGMJ
          .ref   PRG_bufaddr,PRG_length,PRG_paddr
          .ref   PROTECT,DELAY,REGS,ARRAY
          .sect  ”fl_prg”
*********************************************************
*   GPGMJ: This routine programs all or part of          *
*      the flash as specified by the variables:          *
*      PRG_paddr    Destination start address            *
*      PRG_length   Source buffer length              *
*      PRG_bufaddr  Buffer start address              *
*                                                     *
*   The following variables are used for temp         *
*   storage:                                          *
*      AR0       Used for comparisons                 *
*      AR1       Used for pgm pulse count             *
*      AR2       Used for row banz loop               *
*      AR3       Used for buffer addr index           *
*      FL_ADRS   Used for flash address               *
*      FL_DATA   Used for flash data                  *
*      BASE_0    Used for row–done flag               *
*      BASE_1    Used for row start address           *
*      BASE_2    Used for row length–1                *
```

```
*      BASE_3    Used for buffer/row start addr       *
*      BASE_4    Used for destination end addr        *
*      BASE_5    Used for byte mask                   *
*****************************************************
GPGMJ: SPLK   #0,IMR        ;MASK ALL INTERRUPTS
       SETC   INTM          ;GLOBALLY MASK ALL INTERRUPTS
       SPLK   #0,ERROR      ;Initialize error flag (no error).
       LACL   PRG_paddr     ;Get destination start address.
       SACL   FL_ADRS       ;Save as current address.
       ADD    PRG_length    ;Determine destination end addr.
       SUB    #1            ;
       SACL   BASE_4        ;Save destination end addr.
       LACL   PRG_paddr     ;Get destination start addr.
       LAR    AR3,PRG_bufaddr  ;Get buffer start address.
********Begin a new row.*
NEWROW
       SACL   BASE_1           ;Save row start address.
       SAR    AR3,BASE_3       ;Save buffer/row start address.
       LAR    AR1,#0           ;Init pulse count to zero.
       SPLK   #31,BASE_2       ;Init row length–1 to 31.
       AND    #001Fh           ;Is start addr on row boundary?
       CC     ADJ_ROW,NEQ      ;If not then adjust row length.
       LACL   BASE_1           ;Get row start address.
       OR     #001Fh           ;Get row end address.
       SUB    BASE_4           ;Is end address on row boundary?
       CC     ADJ_ROW,GT       ;If not then adjust row length.
********Same row, next pulse.*
SAMEROW   SPLK   #1,BASE_0     ;Set row done flag = 1(True).
          LACL   BASE_1        ;Get row start address.
          SACL   FL_ADRS       ;Save as current address.
          LAR    AR3,BASE_3    ;Get buffer/row start addr.
          LAR    AR2,BASE_2    ;Init row index.
**  Repeat the following code 32 times or until end of row.*
LOBYTE       ;********First, do low byte.*
   CALL   SET_MODULE,AR4      ;Determine which flash module.
      SPLK   #0FFh,BASE_5     ;Set lo–byte mask.
      CALL   PRG_BYTE         ;Check/Program lo–byte.
      SPLK   #0FF00h,BASE_5   ;Set hi–byte mask.
   CALL   PRG_BYTE            ;Check/Program hi–byte.
NEXTWORD     ;********Next word in row.
       LACL   FL_ADRS          ;Load address for next word.
       ADD    #1               ;Increment address.
       SACL   FL_ADRS          ;Save as current address.
       MAR    *,AR3            ;ARP –> buffer addr index.
       MAR    *+,AR2           ;Inc, and ARP –> row index.
       BANZ   LOBYTE           ;Do next word,and dec AR2.
**  Reached end of row. Check if row done. *
       BIT    BASE_0,15        ;Get row_done flag.
       BCND   ROW_DONE,TC      ;If 1 then row is done.
       MAR    *,AR1            ;Else, row is not done, so
       MAR    *+               ;inc row pulse count.
       LAR    AR0,#MAX_PGM     ;Check if passed allowable max.
       CMPR   2                ;If AR1>MAX_PGM then
```

```
       BCND   EXIT,TC          ;fail, don't continue.
       B      SAMEROW          ;else, go to beginning
                                  ;of same row.
** If row done, then check if Array done. *
ROW_DONE
       LACL   FL_ADRS          ;Check if end of array.
       SUB    BASE_4           ;Subtract end addr.
       BCND   DONE, GT         ;If >0 then done.
** Else, go to next row. *
       LACL   FL_ADRS
       B      NEWROW           ;Start new row.
** If here, then done.
DONE
       CALL   ARRAY            ;Access flash in array mode.
       RET                     ;Return to calling program.
** If here, then unit failed to program. *
EXIT   SPLK   #1,ERROR             ;Update error flag (error).
       B      DONE                ;Get outa here.
***********************************************
       .page
***********************************************
* ADJ_ROW: This routine is used to adjust the  *
* row length, if the start or end address of   *
* code being programmed does not fall on a row *
* boundary. The row length is passed in the    *
* BASE_2 variable, and the adjustment value to *
* be subtracted is passed in the accumulator.  *
***********************************************
ADJ_ROW
       NEG              ;Take twos complement.
       ADD    BASE_2    ;Add row length.
       SACL   BASE_2    ;Save new row length.
       RET
***************************************************
*  SET_MODULE: This routine is used to point to   *
*  the appropriate flash array control register   *
*  This is only important for 'F2XX devices with  *
*  multiple flash modules like the 320F206. The   *
*  variable FL_ST is returned with the correct *
*  register address.                              *
*  The following resources are used               *
*  temporarily:                                   *
*      AR0    Used for comparisons                *
*      AR4    Used for flash address              *
***************************************************
SET_MODULE
       LAR    AR4,FL_ADRS   ;AR4 = current address.
       SPLK   #0,FL_ST      ;FL_ST = FLASH0 CTRL REGS
       LAR    AR0,#4000H    ;AR0 = compare value.
       CMPR   1             ;If AR4 < AR0 then
                            ;FL_ADRS < 4000H; SET TC
       BCND   FL0,TC        ;Address is in FL0.
*                           ;Else address is in FL1.
```

```
       SPLK   #04000h,FL_ST;FL_ST = FLASH1 CTRL REGS
FL0    RET
*************************************************************
   .page
*************************************************************
*  THIS SECTION PROGRAMS THE VALUE STORED IN FL_DATA INTO   *
*  THE FLASH ADDRESS DEFINED BY FL_ADRS.                    *
*                                                           *
*  The following resources are used for temporary storage:  *
*     AR6    Parameter passed to Delay                      *
*     SPAD1  Flash program and STOP commands                *
*     SPAD2  Flash program + EXE command.                   *
*************************************************************
EXE_PGM                    ;                                *
*                                                           *
   CALL   ARRAY            ;ACCESS ARRAY                    *
*  LOAD WADRS AND WDATA                                     **
      LACL   FL_ADRS       ;ACC => PROGRAM ADRS             *
      TBLW   FL_DATA       ;LOAD WADRS AND WDATA            *
   CALL   REGS           ;ACCESS FLASH REGS                 *
*  SET UP WRITE COMMAND WORDS                               **
      LACL   PROTECT       ;GET SEGMENT PROTECT MASK        **
      OR     #WR_CMND      ;OR IN WRITE COMMAND             **
      SACL   SPAD1         ;SPAD1 = WRITE COMMAND           **
      OR     #WR_EXE       ;OR IN EXEBIN COMMAND            **
      SACL   SPAD2         ;SPAD2 = WRITE EXE COMMAND       **
      LACL   FL_ST         ;ACC => (FLASH)                  *
*                                                           *
*  ACTIVATE WRITE BIT                                       **
      TBLW   SPAD1         ;EXECUTE COMMAND                 **
      LAR    AR6,#D10      ;SET DELAY                       **
   CALL   DELAY,*,AR6   ;WAIT                               **
*  SET EXEBIN BIT                                           **
      TBLW   SPAD2         ;EXECUTE COMMAND                 **
      LAR    AR6,#D100     ;SET DELAY                       **
   CALL   DELAY,*,AR6   ;WAIT                               **
*  STOP WRITE OPERATION                                     *
      SPLK   #0,SPAD1      ;SHUT DOWN WRITE OPERATION       *
      TBLW   SPAD1         ;EXECUTE COMMAND                 *
      TBLW   SPAD1            ;EXECUTE COMMAND              *
      LAR    AR6,#D10         ;SET DELAY                    *
   CALL   DELAY,*,AR6   ;WAIT                               *
*                                                           *
   RET    ;RETURN TO CALLING SEQUENCE                       *
*************************************************************
   .page
*************************************************************
*  ACTIVATE VER0 ON FLASH READS                             *
*  LOADS FLASH WORD AT ADDR FL_ADRS TO FL_DATA.             *
*  Uses SPAD1 for temporary storage of flash commands.      *
*************************************************************
SET_RD_VER0                ;                                *
   CALL   REGS          ;ACCESS FLASH REGISTERS             *
```

```
        LACL   FL_ST         ;ACC => FLASH                    *
        SPLK   #VER0,SPAD1   ;ACTIVATE VER0                   *
        TBLW   SPAD1         ;EXECUTE COMMAND                 *
        LAR    AR6,#D10      ;SET DELAY                       *
    CALL   DELAY,*,AR6   ;WAIT                                *
    CALL   ARRAY         ;ACCESS FLASH ARRAY                  *
        LACL   FL_ADRS       ;POINT TO ADRS                   *
        TBLR   FL_DATA       ;GET FLASH WORD 1x read          *
        TBLR   FL_DATA       ;   2x read                      *
        TBLR   FL_DATA       ;   3x read                      *
    CALL   REGS          ;ACCESS FLASH REGISTERS              *
        LACL   FL_ST         ;ACC => FLASH                    *
        SPLK   #STOP,SPAD1   ;DEACTIVATE VER0                 *
        TBLW   SPAD1         ;EXECUTE COMMAND                 *
        LAR    AR6,#D10      ;SET DELAY                       *
    CALL   DELAY,*,AR6   ;WAIT                                *
    CALL   ARRAY         ;ACCESS FLASH ARRAY                  *
        RET           ;RETURN TO CALLING SEQUENCE             *
************************************************************
        .page
************************************************
*   PRG_BYTE: Programs hi or lo byte depending on*
*         byte mask (BASE_5).              *
************************************************
PRG_BYTE:
        CALL   SET_RD_VER0      ;Read word at VER0 level.
        MAR    *,AR3            ;ARP –> buffer addr index.
        LACL   *                ;Get word to program.
        XOR    FL_DATA          ;Xor with read–back value.
        AND    BASE_5           ;Mask off hi/lo byte.
        BCND   PB_END,EQ        ;If zero then done.
        XOR    #0FFFFh          ;else, mask off good bits.
        SACL   FL_DATA          ;New data.
        CALL   EXE_PGM          ;PGM Pulse.
        SPLK   #0,BASE_0        ;Set row done flag = 0(False).
PB_END RET
************************************************************
        .end
```

A.1.6 Subroutines Used By All Four Algorithms, SUTILS20.ASM

This assembly file includes two subroutines that change the flash module access mode and one subroutine that performs software delays. More details on the individual functions are given in the comments.

```
*************************************************************
** Delay And Access Mode Subroutines                       **
*                                                           **
*   TMS320F2XX Flash Utilities.                             **
*      Revision: 2.0, 9/10/97                               **
*      Revision: 2.1, 1/31/98                               **
*                                                           **
*   Filename: sutils20.asm                                  **
*                                                           **
*   Called by:    These utilities are used by CLEAR,ERASE,  **
*                 PROGRAM algorithms written for F2xx       **
*                 devices.                                  **
* Function:   DELAY  Delay loop specified by AR6.           **
*             REGS   Clears MODE bit of F_ACCESS0/1 to      **
*                    access flash module control registers. **
*             ARRAY  Sets MODE bit of F_ACCESS0/1 to access **
*                    the flash array.                       **
*************************************************************
       .include  "svar20.h"
       .def   DELAY,REGS,ARRAY
   .sect  "DLY"
************************************
*Delays as follows:                *
*   LAR    AR6,#N       2  Cycles  *
*   CALL   DELAY        4  Cycles  *
*   RPT    #DLOOP  2*(N+1)  Cycles  *
*   NOP       DLOOP*(N+1)  Cycles  *
*   BANZ   DLY_LP    4*N+2  Cycles  *
*   RET              4  Cycles  *
*          ________________________  *
*      =   DLOOP(N+1)+6*N+14 Cycles  *
*   Set N and DLOOP appropriately to *
*   get desired delay.             *
************************************
DELAY                          ;AR6 = OUTER LOOP COUNT
DLY_LP    RPT    #DLOOP        ;APPROX 5US DELAY
          NOP
       BANZ   DLY_LP,*- ;LOOP UNTIL DONE
       RET              ;RETURN TO CALLING SEQUENCE
       .page
************************************************
*   REGS      Clears MODE bit of F_ACCESS0/1 to       **
*             access flash module control registers.  **
************************************************
       .sect  "REG"
REGS
```

```
       SPLK   #0000h,SPAD2
***********The next instruction is for F240 only************
       .if    F24X = 1            ;Assemble for F24X only.
       OUT    SPAD2,F24X_ACCS     ;Enable F240 flash reg mode.
                                  ;SPAD1 is dummy value.
       .endif
*************************************************************
       .if    F24X = 0         ;Assemble for F206 only.
       LACC   FL_ST
       SUB    #4000h
       BCND   reg1,geq         ;if address>= 4000h,set
                               ;set reg mode for flash1 array
       OUT    SPAD2,F_ACCESS0  ;Change mode of flash0.
       RET
reg1   OUT    SPAD2,F_ACCESS1  ;Change mode of flash1.
       .endif
       RET                     ;RETURN TO CALLING SEQUENCE

       .page
*************************************************
*  ARRAY  Sets MODE bit of F_ACCESS0/1 to access  **
*         the flash array.                        **
*************************************************
   .sect  "ARY"
ARRAY
       SPLK   #0001h,SPAD2
***********The next instruction is for F240 only************
       .if    F24X = 1         ;Assemble for F240 only.
       IN     SPAD1,F24X_ACCS  ;Enable F240 flash array mode.
                               ;SPAD1 is dummy value.
       .endif
*************************************************************
       .if    F24X = 0         ;Assemble for F206 only.
       LACC   FL_ST
       SUB    #4000h
       BCND   ary1,geq         ;if address>= 4000h,set
                               ;set reg mode for flash1 array
       OUT    SPAD2,F_ACCESS0  ;Change mode of flash0.
       RET
ary1   OUT    SPAD2,F_ACCESS1  ;Change mode of flash1.
       .endif
       RET                     ;RETURN TO CALLING SEQUENCE
          .end
```

A.2 C-Callable Interface to Flash Algorithms

The two functions erase() and program() are intended for in-application programming of the ’F20x/F24x flash module. These functions were written to be C callable, but they can also be called from assembly as long as the C stack calling convention is used.

```
*****************************************************
*  This file contains two C–callable functions:      *
*     program(), and erase()                         *
*  These functions are used for programming and      *
*  erasing the on–chip flash EEPROM of the ’F2XX     *
*  product family.                                   *
*****************************************************
*  The functions provide a C–callable, interface to  *
*  the standard ’F2XX flash algorithms. They can     *
*  also be used from assembly code, as long as the   *
*  C stack calling convention is used. Since the     *
*  standard flash algorithms are actually used to    *
*  perform the various flash operations, they must   *
*  must be combined with this code at link time.     *
*                                                    *
*  The erase function includes all the operations    *
*  (clear+erase+flw) required to prepare the flash   *
*  for programming. In addition to providing the     *
*  C–callable interface, this function is very       *
*  useful since it provides a single call to erase   *
*  the flash memory.                                 *
*  Since programming the device requires a single    *
*  algorithm, the only purpose for the program()     *
*  function is to provide a C–callable interface.    *
*  The program() function transfers a specified      *
*  block of data memory into a specified, erased     *
*  flash array.                                      *
*                                                    *
*  The parameters for each function are described    *
*  in detail below. Note these functions cannot      *
*  reside in the same flash module that they are     *
*  meant to modify.                                  *
*                                                    *
*     10/29/97  Ruben D. Perez                       *
*               DSP Applications Team                *
*               Texas Instruments, Inc.              *
*     03/20/98  Updated for inclusion in flash       *
*               technical reference.                 *
*****************************************************


   .title ”C–callable Interface to ’F2XX Flash Algorithms**”
;**C–callable functions defined in this file.
   .global   _erase,   _program
```

```
;**Variables included from flash algorithms.
       .include  "svar20.h"    ;Variable declarations
       .ref   GCLR             ;References clear algo.
       .ref   GPGMJ            ;References program algo.
       .ref   GERS             ;References erase algo.
       .ref   FLWS             ;References flash-write algo.
;**Parameters used by flash algorithms.
       .def   PRG_bufaddr, PRG_paddr
       .def   PRG_length, PARMS
       .def   SEG_ST,SEG_END,PROTECT
*************************************************************
VARS:  .usect "PRG_data",16;This is an uninitialized data    *
                           ;section required by the standard*
                           ;flash algos for temporary       *
                           ;variables. Pointers to this     *
                           ;space are hardcoded in SVAR20.H,*
                           ;and variables are init'd at     *
                           ;run time.                       *
*************************************************************
PARMS: .usect "PRG_parm",10;This is an uninitialized data    *
                           ;section used for temporary      *
                           ;variables and for passing       *
                           ;parameters to the flash         *
                           ;algorithms.                     *
*************************************************************
PROTECT   .set   PARMS  ;Segment enable bits.                *
*************************************************************
*****  Parameters needed for Programming algorithm.   ********
*************************************************************
PRG_bufaddr  .set   PARMS+1   ;Addr of buffer for pgm data   *
PRG_paddr    .set   PARMS+2   ;First flash addr to program   *
PRG_length   .set   PARMS+3   ;Length of block to program    *
*************************************************************
**  Parameters needed for CLEAR, ERASE, and FLW algorithms.  *
*************************************************************
SEG_ST    .set   PARMS+4      ;Segment start address.        *
SEG_END   .set   PARMS+5      ;Segment end address.          *
*************************************************************
****             Other misc variables.                  ****
*************************************************************
ERS_COUNT .set   PARMS+6      ;Used for erase fail count.    *
SV_AR1    .set   PARMS+7      ;Used to save AR1.             *
*************************************************************
       .sect  "PRG_text"
*********************************************************
*  function erase(PROTECT,SEG_ST,SEG_END)                *
*  Status is returned in the accumulator.                *
*     0 = Fail,1 = Pass                                  *
*********************************************************
*  This function performs the clear and erase operation  *
*  on the 'F2XX flash. If the erase operation fails, the *
*  flash-write operation is used to try to recover from  *
*  depletion. If the array recovers, the entire process  *
*  (clr+ers+flw) is repeated a maximum of 10 times. The  *
*  return value indicates the status. If this function   *
```

```
*  passes, the flash is ready to be reprogrammed. The   *
*  operations are performed on the segments of the flash*
*  module described by the parameter list:              *
*     1)PROTECT–defines which flash segments to protect.*
*     2)SEG_ST –start address of segment to be erased.  *
*     3)SEG_END–end address of segment to be erased.    *
*  To erase flash0 use erase(0xff00,0x0000,0x3fff).     *
*  To erase flash1 use erase(0xff00,0x4000,0x7fff).     *
*******************************************************
*  CAUTION: Erasing individual segments is not allowed. *
*     The PROTECT parameter should always be set to     *
*     enable all segments, and SEG_ST and SEG_END       *
*     should be set to the end and start address of     *
*     the array to be erased.                           *
*******************************************************
_erase:
ERS_PARAMS   .set 3
AR_STACK     .set ar1
AR_PROTECT   .set ar2
AR_SEG_ST    .set ar3
AR_SEG_END   .set ar4

;Begin C Preprocessing
   POPD   *+     ;pop return address, push on software stack
   sar    ar0,*+ ;save FP
   sar    ar6,*  ;save ar6
   sbrk   #3
      ;get arguments and place them properly – take them from
      ;the software stack and place them into their correct
      ;positions
   lar    AR_PROTECT,*–
   lar    AR_SEG_ST,*–
   lar    AR_SEG_END,*–
   adrk #ERS_PARAMS+4  ;ar1 = next empty point on stack (SP)
;End C Preprocessing
   LDP    #PARMS
   SAR    AR1,SV_AR1    ;Save AR1.
   SPLK   #0,ERS_COUNT  ;Set erase fail count to 0.
   SPLK   #0,ERROR      ;Set algo error flag to 0 (no errors).
**********Put parameters where they belong.**********
   SAR    AR_PROTECT,PROTECT
   SAR    AR_SEG_ST,SEG_ST
   SAR    AR_SEG_END,SEG_END
***********Next Setup to clear flash ************
ers_loop:
      CALL   GCLR          ;Clear flash.
      LACL   ERROR         ;Check for CLEAR/ERASE error
      BCND   ers_error,neq ;If error, then hard fail.
***********Next Setup to erase flash ************
      CALL   GERS          ;Erase flash.
      LACL   ERROR         ;Check for CLEAR/ERASE error
      BCND   depletion,neq ;If error, try Flash–write.
      LACL   #1            ;Else, no errors erasing.
      B      ers_done      ;Restore registers and return.
depletion:
   LACL   ERS_COUNT    ;Get erase fail count.
```

```
   ADD    #1           ;Increment fail count.
   SACL   ERS_COUNT    ;Save new count.
   SUB    #10          ;CHECK for max of 10.
   BCND   ers_error,GT ;If ers_cout>10 then hard fail.
   CALL   FLWS         ;Else, try to recover from depletion.
   LACL   ERROR        ;Check for FLASH-WRITE error.
   BCND   ers_error,neq;If couldn't recover, then hard fail.
   B      ers_loop     ;Else, try erase again.
ers_error:
   LACL   #0           ;Error while erasing.
ers_done:
   LAR    AR1,SV_AR1   ;Restore AR1.
   CLRC   OVM          ;Disable overflow.
*****************************************
;Begin C Post Processing
   mar *,ar1
   sbrk #1
   lar    ar6,*-    ;save FP
   lar    ar0,*-    ;save ar6
   pshd   *         ;pop return address, push on s/w stack
;End C Post Processing
   ret
*****************END of _erase*****************************

*********************************************************
*  function  program(PROTECT,PRG_bufaddr,PRG_paddr,     *
*               PRG_length)                             *
*  Status will be returned in the accumulator.          *
*     0 = Fail, 1 = Pass                                *
*********************************************************
*  This function performs the program operation on the  *
*  'F2XX flash. The values to be programmed will be read*
*  from a buffer in data memory. The function can program*
*  one to n words of flash in a single call; restricted *
*  only by the data buffer size. If the function passes,*
*  the flash was programmed correctly. The function is  *
*  controlled by the following parameter list:          *
*     1)PROTECT    -flash segments to protect.          *
*     2)PRG_bufaddr-Start address of program buffer in  *
*                   data memory.                        *
*     3)PRG_paddr  -Start address of flash locations to *
*                   be programmed.                      *
*     4)PRG_length -Number of words to be programmed.   *
*                                                       *
*  To program 20 words of flash1 starting at address    *
*  0x4020, from a buffer at 0x0800@data use this:       *
*     program(0xff00,0x0800,0x4020,20).                 *
*********************************************************
_program:
PRG_PARAMS    .set 4
AR_STACK      .set ar1
;**Parameters to be popped from s/w stack.
AR_PROTECT    .set ar2
AR_bufaddr    .set ar3
AR_paddr      .set ar4
AR_length     .set ar5
```

```
;Begin C Preprocessing
   POPD   *+        ; pop return address, push on s/w stack
   sar ar0,*+       ; save FP
   sar ar6,*        ; save ar6
   sbrk #3
   ; Local variables (and parameters) are set up as follows:
   ;
   ;get arguments and place them properly – take them from
   ;the software stack and place them into their correct
   ;positions
   lar AR_PROTECT,*–
   lar AR_bufaddr,*–
   lar AR_paddr,*–
   lar AR_length,*–
   adrk #PRG_PARAMS+4  ; ar1 = next empty point on stack (SP)
; End C Preprocessing
       LDP    #PARMS
       SAR    AR1,SV_AR1    ;Save AR1.
       SPLK   #0,ERROR      ;Set algo error flag to 0
                            ;(no errors).
**********Put parameters where they belong.**********
       SAR AR_PROTECT,PROTECT
       SAR AR_bufaddr,PRG_bufaddr
       SAR AR_paddr,PRG_paddr
       SAR AR_length,PRG_length
***********Next, program flash ************
       CALL   GPGMJ         ;Program flash from buffer.
       LACL   ERROR         ;Check for program error.
       BCND   prg_error,neq ;If error then clear ACC.
       LACL   #1            ;Else, No errors programming.
       B      prg_done

prg_error:
       LACL   #0            ;Error while programming.
prg_done:
       LAR    AR1,SV_AR1       ;Restore AR1.
       CLRC   OVM           ;Disable overflow.
*****************************************
;Begin C Post Processing
   mar *,ar1
   sbrk #1
   lar    ar6,*–        ;save FP
   lar    ar0,*–        ;save ar6
   pshd   *             ;pop return address, push on s/w stack
;End C Post Processing
   ret
*****************END of _program***********************
```

A.3 Sample Assembly Code to Erase and Reprogram the TMS320F206

The algorithm files can be used from assembly in a straightforward manner. In general, the algorithms can reside anywhere in program space. However, the algorithms cannot be executed from the flash module that is being modified, and the algorithms must execute with zero wait states. The assembly code and linker command file in this section provide a working example for the 'F206. In this example, the algorithms reside in SARAM, and flash1 is erased and reprogrammed.

A.3.1 Assembly Code for TMS320F206

```
*********************************************************
*  Filename: ASMEXAMP.ASM                               *
*  Description:                                         *
*  This file contains an example of how to erase        *
*  and program the TMS320F206 flash from assembly       *
*  code using the standard flash algorithm modules.     *
*  The example erases one of the 'F206 flash            *
*  modules, then programs the first three words.        *
*  Since the standard flash algorithms are actually     *
*  used to perform the various flash operations,        *
*  they must must be combined with this code at         *
*  link time.                                           *
*                                                       *
*  03/20/98  Updated for inclusion in flash             *
*            technical reference.                       *
*********************************************************
      .title"**Example of Using 'F2XX Flash Algorithms**"
;**Variables included from flash algorithms.
      .include "svar20.h"   ;Variable declarations
      .ref  GCLR            ;References clear algo.
      .ref  GPGMJ           ;References program algo.
      .ref  GERS            ;References erase algo.
      .ref  FLWS            ;References Flash-write algo.
;**Parameters used by flash algorithms.
      .def  PRG_bufaddr, PRG_paddr
      .def  PRG_length, PARMS
      .def  SEG_ST,SEG_END,PROTECT
*********************************************************
VARS: .usect"PRG_data",16   ;This is an uninitialized   *
                            ;data section required by   *
                            ;the standard flash algos   *
                            ;for temporary variables.   *
                            ;Pointers to this space     *
                            ;are hardcoded in SVAR20.H, *
                            ;and variables are          *
                            ;init'd at run time.        *
*********************************************************
PARMS:.usect"PRG_parm",10   ;This is an uninitialized   *
```

```
                              ;data section used for      *
                              ;temporary variables, and   *
                              ;for passing parameters     *
                              ;to the flash algorithms.   *
*************************************************************
PROTECT   .set   PARMS        ;Segment enable bits.       *
*************************************************************
***Parameters needed for Programming algorithm.      ***
*************************************************************
PRG_bufaddr  .set   PARMS+1   ;Address of buffer for      *
                              ;program data.              *
PRG_paddr    .set   PARMS+2   ;First flash address to     *
                              ;program.                   *
PRG_length   .set   PARMS+3   ;Length of block to program.*
*************************************************************
*  Parameters needed for CLEAR, ERASE, and FLW algorithms.
*
*************************************************************
SEG_ST    .set   PARMS+4   ;Segment start address.        *
SEG_END   .set   PARMS+5   ;Segment end address.          *
*************************************************************
****            Other misc variables.                 ****
*************************************************************
ERS_COUNT .set   PARMS+6   ;Used for erase fail count.    *
*************************************************************
      .text
*************************************************************
** First, erase flash1 by invoking the clear and erase  *
** algorithms.                                          *
** Note: three parameters must be initialized before    *
** calling the algorithms.                              *
*************************************************************
      LDP    #PARMS
      SPLK   #0,ERS_COUNT ;Set erase fail count to 0.
**********Put parameters where they belong.**********
      SPLK   #0ff00h,PROTECT
      SPLK   #04000h,SEG_ST
      SPLK   #07FFFh,SEG_END
***********First clear flash ************
ers_loop:
      CALL   GCLR             ;Clear flash.
      LACL   ERROR            ;Check for CLEAR error
      BCND   ers_error,neq    ;If error, then hard fail.
***********Next erase flash ************
      CALL   GERS             ;Erase flash.
      LACL   ERROR            ;Check for CLEAR error
      BCND   depletion,neq    ;If error, then try
                              ;flash–write.
      B      ers_done         ;Else, no errors erasing.
depletion:
      LACL   ERS_COUNT        ;Get erase fail count.
      ADD #1                  ;Increment fail count.
      SACL   ERS_COUNT        ;Save new count.
```

```
      SUB #10               ;CHECK for max of 10.
      BCND  ers_error,GT    ;If ers_cout>10 then hard
                            ;fail.
      CALL  FLWS            ;Else, try to recover from
                        ;depletion.
      LACL  ERROR         ;Check for FLASH–WRITE error.
      BCND  ers_error,neq   ;If couldn't recover, then
                            ;hard fail.
      B     ers_loop      ;Else, try erase again.
ers_error:
*********************************************************
** If here, then an unrecoverable error has occurred **
** during erase. In an actual application, the system**
** takes some action to indicate that service is     **
** required.                                         **
*********************************************************
      B     ers_error     ;Error while erasing.
ers_done:
*********************************************************
** If here, then flash is erased and ready to be     **
** reprogrammed. This is a good place in the example **
** to set a breakpoint so that erasure can be        **
** verified (i.e., all flash bits should be 1).      **
*********************************************************

***********************************************************
** At this point, an actual application fills a buffer   **
** with the data to be programmed. To simulate this in   **
** the example, three SARAM locations are initialized.   **
***********************************************************
      LAR   AR1,   #0c00h;Using last 3K of SARAM as
                         ;buffer.
      MAR   *,AR1
      SPLK  #0AAAAh,*+   ;Use dummy data for buffer.
      SPLK  #05555h,*+
      SPLK  #0AAAAh,*
***********************************************************
** Now that the data to be programmed is ready, the      **
** programming algorithm is invoked. Note that four      **
** parameters must be initialized before calling the     **
** algorithm.                                            **
***********************************************************
      LDP #PARMS
**********Put parameters where they belong.**********
      splk  #0ff00h,PROTECT
      splk  #0c00h,PRG_bufaddr
      splk  #04000h,PRG_paddr
      splk  #3,PRG_length
***********Next program flash ************
      CALL  GPGMJ           ;Program flash from buffer.
      LACL  ERROR           ;Check for program error.
      BCND  prg_error,neq   ;If error then clear ACC.
      B     prg_done        ;Else, No errors programming.
```

```
prg_error:
*********************************************************
** If here, then an error has occurred during         **
** programming. In an actual application, the system **
** takes some action to indicate that service is      **
** required.                                          **
*********************************************************
      B     prg_error    ;Error while programming.

prg_done:
*********************************************************
*********
** If here, then flash has been successfully programmed.**
*********************************************************
      B     prg_done     ;Done programming.
```

A.3.2 Linker Command File for TMS320F206 Sample Assembly Code

```
/*****************************************************************************/
/* Filename: ASMEXAMP.CMD                                                    */
/* Description: Linker command file for 'F206 example of on-chip             */
/* flash programming from assembly. This command file links the example to addr */
/* 0x8000 of the on-chip SARAM so that the debugger can be used to set       */
/* breakpoints. Another benefit of linking the example to SARAM is that the  */
/* code can be modified to operate on either flash module0, or module1, or   */
/* both.                                                                     */
/* Notes:                                                                    */
/*     1. This example expects the 'F206 SARAM to be mapped in both data space  */
/*        and program space (DON=PON=1).                                     */
/*     2. The object modules for the standard flash algos are expected to be in */
/*     a subdirectory (ALGOS) of the path of this file.                      */
/*****************************************************************************/
/* Rev1.0                                                          3/98 RDP  */
/*****************************************************************************/

/*****************************Command Line Options*****************************/
-e .text
-o asmexamp.out
-m asmexamp.map

/*************************Input Files*************************************/
asmexamp.obj        /*User assembly code that calls flash algos.          */
algos\spgm20.obj    /*Standard Programming algorithm.                     */
algos\sclr20.obj    /*Standard Clear algorithm.                           */
algos\sera20.obj    /*Standard Erase algorithm.                           */
algos\sflw20.obj    /*Standard Flash-write algorithm.                     */
algos\sutils20.obj  /*Subroutines used by standard algos.                 */

/*****************************Memory Map**************************************/
MEMORY
{
PAGE 0:   /*  PM - Program memory */

   FLASH0: origin = 0x0000, length = 0x3fff
   FLASH1: origin = 0x4000, length = 0x3fff
   PSARAM: origin = 0x8000, length = 0x400 /*Use 1K of SARAM for PROGRAM       */
   B0:  origin = 0xff00, length = 0x1ff
```

```
PAGE 1:   /* DM – Data memory */

   BLK_B2: origin = 0x60,length = 0x20        /*BLOCK B2                          */
   DSARAM: origin = 0xc00, length = 0xC00     /*Use 3K of SARAM for data DON=1  */
   EX1_DM: origin = 0x4000, length = 0x4000   /*External data RAM                 */
   B1: origin = 0x300, length = 0x1ff         /*B1 Ram (Used for algo vars )    */
}

/************************Section Allocation*********************************/
SECTIONS
{
   .text  :  {} >  PSARAM PAGE 0/* asmexamp.asm */

      /*All these sections are for flash programming.*/
   fl_prg    :  {} >  PSARAM PAGE 0    /**Programming Algorithm*****/
   fl_clr    :  {} >  PSARAM PAGE 0    /******Clear Algorithm*******/
   fl_ers    :  {} >  PSARAM PAGE 0    /******Erase Algorithm*******/
   fl_wrt    :  {} >  PSARAM PAGE 0    /****Flash–write Algorithm***/
   DLY       :  {} >  PSARAM PAGE 0    /******Delay Subroutine******/
   REG       :  {} >  PSARAM PAGE 0    /*******Regs Subroutine******/
   ARY       :  {} >  PSARAM PAGE 0    /******Array Subroutine******/
   PRG_data  :  {} >  B1     PAGE 1    /*Reserved in asmexamp.asm **/
                                       /*for flash algo variables.**/
   PRG_parm  :  {} >  B1     PAGE 1    /*Reserved in asmexamp.asm **/
                                       /*for param passing to algos*/
   /*End of sections for flash programming.      */

}
```

A.4 Sample C Code to Erase and Reprogram the TMS320F206

Because the algorithm implementations do not follow the C-calling convention of the 'C2000 C environment, they cannot be used directly from C. The assembly code of section A.2, *C-Callable Interface to Flash Algorithms*, is provided as a C-callable interface to the programming algorithms. The following C source file and linker command file provide a working example for the 'F206. In this example, the algorithms reside in the on-chip SARAM, and either flash0 or flash1 can be reprogrammed. The code can be relocated anywhere in program space, with the exceptions described in section A.3, *Using the Algorithms With Assembly Code*.

A.4.1 C Code That Calls the Interface to Flash Algorithms for TMS320F206

```
/*************************************************/
/* Filename: sample.c                            */
/* Description: This is an example of how to     */
/* program the 'F2XX flash from C code.          */
/* The C–callable interface for the standard     */
/* flash algorithms is used. This interface is   */
/* defined in the file <flash.asm>, as two       */
/* C–callable functions: erase(), and program()  */
/* At link time, this example must be combined   */
/* with the code in <flash.asm> as well as with  */
/* the object modules for the standard algos.    */
/*************************************************/
/* This example is set up for the TMS320F206,    */
/* and uses the SARAM as a buffer for programming */
/* data. The code first erases module1,          */
/* then programs the first three locations.      */
/*************************************************/
/* Rev1.0                              10/97 RDP */
/*************************************************/
extern    int    erase();     /* Declare external func for flash erase.   */
extern    int    program();   /* Declare external func for flash programming.   */
main()
{
   int *a;
   if (erase(0xff00,0x4000,0x7fff))
   { /*Flash is erased, now let's program it.*/
      /* Init program buffer. */
      a=(int *)0xC00;  /*Use last 3K of SARAM for data buffer*/
      a[0]=0x7A80;
      a[1]=0x0FDF;
      a[2]=0x7A80;

      /*Program the flash from the buffer*/
      if (program(0xff00,0xc00,0x4000,0x3))
      { /*Flash programmed ok.*/
         while(1){} /*Spin here forever*/
      }
      else
```

```
      { /*Flash fails programming, EXIT*/
         while(1){} /*Spin here forever*/
      }
   }
   else
   { /*Flash fails erase, EXIT*/
      while(1){} /*Spin here forever*/
   }
}
```

A.4.2 Linker Command File for TMS320F206 Sample C Code

```
/*************************************************************************/
/* Filename: F206_SA.CMD                                                 */
/* Description: Linker command file for 'F206 example of on-chip flash   */
/*           programming from C code. This command file links the        */
/*           example to addr 0x8000 of the on-chip SARAM so that         */
/*           the debugger can be used to set breakpoints. Another        */
/*           benefit of linking the example to SARAM is that the         */
/*           C code can be modified to operate on either flash           */
/*           module0, or module1, or both.                               */
/*           Notes:                                                      */
/*           1. This example expects the 'F206 SARAM to be               */
/*              mapped in both data space and program space              */
/*              (DON=PON=1).                                             */
/*           2. The object modules for the standard flash algos          */
/*              are expected to be in a subdirectory (ALGOS) of          */
/*              the path of this file.                                   */
/*************************************************************************/
/* Rev1.0                                                  10/97 RDP */
/*************************************************************************/

/************************Command Line Options*************************/
-cr             /*Use Ram init model.                                  */
-heap 0x0       /*No heap needed for this example.                     */
-stack 0x96     /*150-word stack is enough for this example.           */
-x              /*Force rereading of libraries.                        */
-l c:\dsptools\rts2xx.lib
-o sample_S.out
-m sample_S.map

/*****************************Input Files*****************************/
sample.obj          /*User C code with calls to erase() and program() */
flash.obj           /*C-callable interface to standard algorithms.    */
algos\spgm20.obj    /*Standard Programming algorithms.                */
algos\sclr20.obj    /*Standard Clear algorithm.                       */
algos\sera20.obj    /*Standard Erase algorithm.                       */
algos\sflw20.obj    /*Standard Flash-write algorithm.                 */
algos\sutils20.obj  /*Subroutines used by standard algorithms.        */

/*****************************Memory Map******************************/
MEMORY
{
PAGE 0: /*    PM - Program memory */
```

```
   FLASH0:   origin =  0x0000,   length =  0x3fff
   FLASH1:   origin =  0x4000,   length =  0x3fff
   PSARAM:   origin =  0x8000,   length =  0x400  /*Use 1K of SARAM for PROGRAM*/
   B0:       origin =  0xff00,   length =  0x1ff

PAGE 1: /* DM – Data memory   */

   BLK_B2:   origin =  0x60,     length =  0x20   /*BLOCK B2    */
   DSARAM:   origin =  0xc00,    length =  0xC00  /*Use 3K of SARAM for data */
                                                  /*DON=1*/
   EX1_DM:   origin =  0x4000,   length =  0x4000 /*External data RAM */
   B1:       origin =  0x300,    length =  0x1ff  /*B1 RAM (Used for algo vars )*/
}

/*************************Section Allocation*************************/
SECTIONS
{
   .text     :  {} >  PSARAM PAGE 0    /* sample.c  */

   /*All these sections are for flash programming.*/
   PRG_text  :  {} >  PSARAM PAGE 0    /**erase() and program()*****/
                                       /*****from flash.asm file****/
   fl_prg    :  {} > PSARAM PAGE 0  /**Programming Algorithm*****/
   fl_clr    :  {} > PSARAM PAGE 0  /******Clear Algorithm*******/
   fl_ers    :  {} > PSARAM PAGE 0  /******Erase Algorithm*******/
   fl_wrt    :  {} > PSARAM PAGE 0  /****Flash–write Algorithm***/
   DLY       :  {} > PSARAM PAGE 0  /******Delay Subroutine******/
   REG       :  {} > PSARAM PAGE 0  /*******Regs Subroutine******/
   ARY       :  {} > PSARAM PAGE 0  /******Array Subroutine******/
   PRG_data  :  {} >  B1  PAGE 1    /*Reserved in flash.asm for**/
                                    /***flash algo variables.****/
   PRG_parm  :  {} >  B1  PAGE 1    /*Reserved in flash.asm for**/
                                    /*parameter passing to algos*/
   /*End of sections for flash programming.      */

   .bss      :  {} >  B1 PAGE 1
   .cinit    :  {} >  B1 PAGE 1
   .const    :  {} >  B1 PAGE 1
   .data     :  {} >  B1 PAGE 1
   .stack    :  {} >  B1 PAGE 1    /*C stack.            */

}
```

A.5 Sample Assembly Code to Erase and Reprogram the TMS320F240

The algorithm files can be used from assembly in a straightforward manner. In general, the algorithms can reside anywhere in program space. However, the algorithms cannot be executed from the flash module that is being modified, and the algorithms must execute with zero wait states. The assembly code and linker command file in this section provide a working example for the ’F240.

Note:

This is not an actual application example since a boot mechanism is required to load the external SRAM on powerup. This example uses the ’C2xx C-source Debugger to download the code to the external SRAM. In addition, no reset or interrupt vectors are initialized.

The system requirements are F240 EVM or target board with external program space SRAM located at 0x8000 and a minumum of 1K words.

A.5.1 Assembly Code for TMS320F240

```
*************************************************************
*   Filename: ASMEXA24.ASM                                  *
*   Description:                                            *
*   This file contains an example of how to erase           *
*   and program the TMS320F240 flash from assembly          *
*   code using the standard flash algorithm modules.        *
*   The example erases the ’F240 flash                      *
*   modules, then programs the first three words.           *
*   Since the standard flash algorithms are actually        *
*   used to perform the various flash operations,           *
*   they must must be combined with this code at            *
*   link time.                                              *
*                                                           *
*   03/25/98   Updated for inclusion in flash               *
*              technical reference.                         *
*************************************************************
       .title ”**Example of Using ’F2XX Flash Algorithms**”
```

```
;**Variables included from flash algorithms.
       .include  "svar20.h"    ;Variable declarations
       .ref   GCLR             ;References clear algo.
       .ref   GPGMJ            ;References program algo.
       .ref   GERS             ;References erase algo.
       .ref   FLWS             ;References flash-write algo.
;**Parameters used by flash algorithms.
       .def   PRG_bufaddr, PRG_paddr
       .def   PRG_length, PARMS
       .def   SEG_ST,SEG_END,PROTECT
;**F240 Register definitions
RTICR     .set   07027h    ;RTI Control Register
WDCR      .set   07029h    ;WD Control Register
CKCR0     .set   0702Bh    ;Clock Control Register 0
CKCR1     .set   0702Dh    ;Clock Control Register 1
SYSSR     .set   0701Ah    ;System Module Status Register
DP_PF1    .set   224       ;page 1 of peripheral file
                           ;(7000h/80h)

*************************************************************
VARS:  .usect "PRG_data",16;This is an uninitialized data    *
                           ;section required by the standard*
                           ;flash algos for temporary       *
                           ;variables. Pointers to this     *
                           ;space are hardcoded in SVAR20.H,*
                           ;and variables are init'd at     *
                           ;run time.                       *
*************************************************************
PARMS: .usect "PRG_parm",10;This is an uninitialized data    *
                           ;section that is used for        *
                           ;temporary variables and for     *
                           ;passing parameters to the flash *
                           ;algorithms.                     *
*************************************************************
PROTECT   .set   PARMS     ;Segment enable bits.            *
*************************************************************
*****     Parameters needed for Programming algorithm.   ***
*************************************************************
PRG_bufaddr   .set   PARMS+1   ;Addr of buffer for pgm data. *
PRG_paddr     .set   PARMS+2   ;1st flash addr to program.   *
PRG_length    .set   PARMS+3   ;Length of block to program.  *
*************************************************************
***Parameters needed for CLEAR, ERASE, and FLW algorithms.  **
*************************************************************
SEG_ST        .set   PARMS+4   ;Segment start address.       *
SEG_END       .set   PARMS+5   ;Segment end address.         *
*************************************************************
****             Other misc variables.                   ****
*************************************************************
ERS_COUNT     .set   PARMS+6   ;Used for erase fail count.   *
*************************************************************
       .text
*************************************************************
** First, initialize the key F240 registers for use with    *
** the EVM.                                                 *
*************************************************************
```

```
F240INIT: ;Set Data Page pointer to page 1 of the
          ;peripheral frame
       LDP#DP_PF1          ;Page DP_PF1 includes WET through*
                           ;EINT frames
       ;initialize WDT registers
       SPLK   #06Fh, WDCR  ;clear WDFLAG, Disable WDT
                           ;(if Vpp=5V), set WDT
                           ;for 1 second overflow (max)
       SPLK   #07h,  RTICR ;clear RTI Flag,
                           ;set RTI for 1 second overflow
                           ;(max)
       ;EVM 10-MHz oscillator settings.
       ;(XTAL2 open, OSCBYP_=GND)
       SPLK   #00B1h,CKCR1 ;CLKIN(OSC)=10MHz,
                           ;Mult by 2, Div by 1.
       SPLK   #00C3h,CKCR0 ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2

       ;Clear reset flag bits in SYSSR
       ;(PORRST, PLLRST, ILLRST, SWRST, WDRST)
       LACL   SYSSR        ;ACCL <= SYSSR
       AND#00FFh           ;Clear upper 8 bits of SYSSR
       SACL   SYSSR        ;Load new value into SYSSR


*********************************************************
** First, erase flash1 by invoking the clear and erase  **
** algorithms.                                          **
** Note: Three parameters must be initialized before    **
** calling the algorithms.                              **
*********************************************************
       LDP    #PARMS
       SPLK   #0,ERS_COUNT ;Set erase fail count to 0.
**********Put parameters where they belong.**********
       SPLK   #0ff00h,PROTECT
       SPLK   #00000h,SEG_ST
       SPLK   #03FFFh,SEG_END
***********First, clear flash ************


ers_loop:
          CALL   GCLR            ;Clear flash.
          LACL   ERROR           ;Check for CLEAR/ERASE error
clrerr:   BCND   ers_error,neq   ;If error, then hard fail.
***********Next erase flash ************
       CALL   GERS            ;Erase flash.
       LACL   ERROR           ;Check for CLEAR/ERASE error
       BCND   depletion,neq   ;If error, then try Flash-write.
       B      ers_done        ;Else, no errors erasing.
```

```
depletion:
      LACL   ERS_COUNT       ;Get erase fail count.
      ADD    #1              ;Increment fail count.
      SACL   ERS_COUNT       ;Save new count.
      SUB    #10             ;CHECK for max of 10.
      BCND   ers_error,GT    ;If ers_cout>10 then hard fail.
      CALL   FLWS            ;Else, try to recover from
                             ;depletion.
      LACL   ERROR           ;Check for FLASH–WRITE error.
      BCND   ers_error,neq   ;If couldn't recover, then hard
                             ;fail.
      B      ers_loop        ;Else, Try erase again.

ers_error:
*************************************************************
** If here, then an unrecoverable error occurred during     **
** erase.                                                   **
** In an actual application, the system takes some action   **
** to indicate that service is required.                    **
*************************************************************
      B      ers_error       ;Error while erasing.

ers_done:
*********************************************************
** If here, then flash is erased and ready to be       **
** reprogrammed.                                       **
** This is a good place in the example to set a        **
** breakpoint so that erasure can be verified (i.e.,   **
** all flash bits should be 1).                        **
*********************************************************

*************************************************************
** At this point, an actual application fills a buffer with **
** the data to be programmed. To simulate this in the       **
** example, three DARAM locations are initialized.          **
*************************************************************
      LAR    AR1, #0380h   ;Using last 128 words of B1 DARAM
                           ;as buffer.
      MAR    *,AR1
      SPLK   #0AAAAh,*+    ;Use dummy data for buffer.
      SPLK   #05555h,*+
      SPLK   #0AAAAh,*
```

```
*************************************************************
** Now that the data to be programmed is ready, the         **
** programming algorithm is invoked. Note: Four parameters  **
** must be initialized before calling the algorithm.        **
*************************************************************
      LDP    #PARMS
**********Put parameters where they belong.**********
      splk   #0ff00h,PROTECT
      splk   #0380h,PRG_bufaddr
      splk   #00000h,PRG_paddr
      splk   #3,PRG_length
***********Next, program flash************
      CALL   GPGMJ        ;Program flash from buffer.
      LACL   ERROR        ;Check for program error.
      BCND   prg_error,neq;If error then clear ACC.
      B      prg_done     ;Else, No errors programming.

prg_error:
*********************************************************
** If here, then an error occurred during programming.  **
** In an actual application, the system takes some      **
** action to indicate that service is required.         **
*********************************************************
      B      prg_error    ;Error while programming.

prg_done:
*********************************************************
** If here, then flash has been successfully programmed.*
*********************************************************
      B      prg_done     ;Done programming.
```

A.5.2 Linker Command File for TMS320F240 Sample Assembly Code

```
/*********************************************************/
/* Filename: ASMEXA24.CMD                                */
/* Description: Linker command file for 'F240 example of */
/* on-chip flash programming from assembly. This command */
/* file links the example to addr 0x8000 of the off-chip */
/* pgm RAM, so that the debugger can be used to set      */
/* breakpoints.                                          */
/*     Notes:                                            */
/*     1. The object modules for the standard flash      */
/*        algos are expected to be in a subdirectory     */
/*        (ALGOS) of the path of this file.              */
/*********************************************************/
/* Rev1.0                                      3/98 JGC  */
/*********************************************************/

/*******************Command Line Options********************/
-e .text
-o asmexa24.out
-m asmexa24.map

/************************Input Files************************/
asmexa24.obj        /*User assembly code that calls flash algos.   */
algos\spgm20.obj    /*Standard Programming algorithm.              */
algos\sclr20.obj    /*Standard Clear algorithm.                    */
algos\sera20.obj    /*Standard Erase algorithm.                    */
algos\sflw20.obj    /*Standard Flash-write algorithm.              */
algos\sutils20.obj  /*Subroutines used by standard algorithms.     */

/************************Memory Map*************************/
MEMORY
{
PAGE 0:   /*  PM - Program memory    */

   FLASH0:   origin = 0x0000, length = 0x4000
   EXTRAM:   origin = 0x8000, length = 0x400  /*Use 1K of Ext. RAM for PROGRAM*/
   B0PGM:    origin = 0xfe00, length = 0x100

PAGE 1:                                       /* DM - Data memory */

   BLK_B2:   origin = 0x60, length = 0x20     /* BLOCK B2*/
   EX1_DM:   origin = 0x8000, length = 0x4000 /* External data RAM */
   B0:       origin = 0x200, length = 0x100   /* B0 Ram (Used for temp data )*/
   B1:       origin = 0x300, length = 0x100   /* B1 Ram (Used for algo vars )*/
}

/************************Section Allocation*************************/
SECTIONS
{
   .text  :{} > EXTRAM PAGE 0       /* asmexa24.asm */
```

```
   /*All these sections are for flash programming.*/
   fl_prg    : {} > EXTRAM    PAGE 0 /**Programming Algorithm*****/
   fl_clr    : {} > EXTRAM    PAGE 0 /******Clear Algorithm*******/
   fl_ers    : {} > EXTRAM    PAGE 0 /******Erase Algorithm*******/
   fl_wrt    : {} > EXTRAM    PAGE 0 /****Flash–write Algorithm***/
   DLY       : {} > EXTRAM    PAGE 0 /******Delay Subroutine******/
   REG       : {} > EXTRAM    PAGE 0 /*******Regs Subroutine******/
   ARY       : {} > EXTRAM    PAGE 0 /******Array Subroutine******/
   PRG_data  : {} > B1        PAGE 1 /*Reserved in asmexamp.asm **/
                                     /*for flash algo variables.**/
   PRG_parm  : {} > B1        PAGE 1 /*Reserved in asmexamp.asm **/
                                     /*for param passing to algos*/
   /*End of sections for flash programming.                     */
}
```

A.6 Using the Algorithms With C Code to Erase and Reprogram the ’F240

Because the algorithm implementations do not follow the C-calling convention of the ’C2000 C environment, they cannot be used directly from C. The assembly code of section A.2, *C-Callable Interface to Flash Algorithms*, is provided as a C-callable interface to the programming algorithms. The C source file and linker command file provide a working example for the ’F240.

In this example, the algorithms reside in external SRAM. The code can be relocated anywhere in program space, with the exceptions described in section A.3, *Using the Algorithms With Assembly Code*.

Note:

This is not an actual application example since a boot mechanism is required to load the external SRAM on powerup. This example uses the ’C2xx C-source Debugger to download the code to the external SRAM. In addition, no reset or interrupt vectors are initialized.

The system requirements are F240 EVM or target board with external program space SRAM located at 0x8000 and a minimum of 1K words.

A.6.1 C Code That Calls the Interface to Flash Algorithms for TMS320F240

```
/*********************************************/
/* Filname: sample24.c                       */
/* Description: This is an example of how to */
/* program the ’F2XX flash from C code.      */
/* The C–callable interface for the standard */
/* flash algorithms is used. This interface is */
/* defined in the file <flash.asm>, as two   */
/* C–callable functions: erase(), and program() */
/* At link time, this example must be combined */
/* with the code in <flash.asm> as well as with */
/* the object modules for the standard algos. */
/*************************************************/
/* This example is setup for the TMS320F240,  */
/* and uses the B1 DARAM as a buffer for program– */
/* –ming data. The code first claers, erases, */
/* then programs the first three locations.   */
/*************************************************/
/* Rev1.0                        03/98 JGC    */
/*************************************************/
```

```
extern int    erase();      /* Declare external func for flash erase. */
extern int    program();    /* Declare external func for flash programming */
extern c240init();      /* Declare external func for C240 register init'l'n */
extern wdtoff();        /* Declare external func for wdt disable */
main()
{
   int *a;
   asm("   CLRC CNF ");          /* map B0 to data space */
   c240init();                   /* initialize key '240 registers */
   wdtoff();                     /* disable WD timer (works when VCCP=5v) */
   if (erase(0xff00,0x0000,0x3fff))
   { /*Flash is erased, now let's program it.*/


      /* Init program buffer. */
      a=(int *)0x200;     /*Use last 128 words of B1 DARAM for data buffer*/
      a[0]=0x7A80;
      a[1]=0x0FDF;
      a[2]=0x7A80;

      /*Program the flash from the buffer*/
      if (program(0xff00,0x200,0x0000,0x3))
      { /*Flash programmed ok.*/
         while(1){} /*Spin here forever*/
      }
      else
      { /*Flash fails programming, EXIT*/
         while(1){} /*Spin here forever*/
      }
   }
   else
   { /*Flash fails erase, EXIT*/
      while(1){} /*Spin here forever*/
   }
}
```

A.6.2 Linker Command File for TMS320F240 Sample C Code

```
/*************************************************************************/
/* Filename: F240_EXT.CMD                                                */
/* Description: Linker command file for 'F240 example of on-chip flash   */
/*        programming from C-code. This command file links the           */
/*        example to addr 0x8000 of the offchip SRAM, so that            */
/*        the debugger can be used to set breakpoints.                   */
/*        Notes:                                                         */
/*        1. The object modules for the standard flash algos             */
/*           are expected to be in a subdirectory (ALGOS) of             */
/*           the path of this file.                                      */
/*************************************************************************/
/* Rev1.0                                                 03/98 JGC      */
/*************************************************************************/
```

```
/************************Command Line Options*************************/
-cr                 /*Use Ram init model.                            */
-heap 0x0           /*No heap needed for this example.               */
-stack 0x96         /*150 word stack is enough for this example.     */
-x                  /*Force re-reading of libraries.                 */
-l c:\dsptools\fix\rts2xx.lib
-o sample24.out
-m sample24.map

/*****************************Input Files*****************************/
sample24.obj        /*User C-code with calls to erase() and program() */
c240init.obj        /*C-callabe asm function to init '240 regs        */
wdtoff.obj          /*C-callable asm function to disable the wdt      */
flash.obj           /*C-callable interface to standard algorithms.    */
algos\spgm20.obj    /*Standard Programming algorithms.                */
algos\sclr20.obj    /*Standard Clear algorithm.                       */
algos\sera20.obj    /*Standard Erase algorithm.                       */
algos\sflw20.obj    /*Standard Flash-write algorithm.                 */
algos\sutils20.obj  /*Subroutines used by standard algorithms.        */

/*****************************Memory Map******************************/
MEMORY
{
PAGE 0:   /* PM - Program memory  */

   FLASH0: origin = 0x0000, length = 0x4000
   EXTRAM: origin = 0x8000, length = 0x400    /*Use 1K of EXT SRAM for PROGRAM*/
   B0:  origin = 0xfe00, length = 0x100

PAGE 1:   /* DM - Data memory */

   BLK_B2:   origin = 0x60, length = 0x20     /*BLOCK B2          */
   DSRAM:    origin = 0x8000, length = 0x4000 /*External data RAM */
   B0DAT:    origin = 0x200, length = 0x100   /*B0 RAM            */
                                              /*(Used for pgm data buffer)*/
   B1:       origin = 0x300, length = 0x100   /* B1 RAM (Used for algo vars )*/
}

/************************Section Allocation***************************/
SECTIONS
{
   .text   :{}  > EXTRAM PAGE 0      /* sample.c */
   /*All these sections are for flash programming.*/
   PRG_text  : {} > EXTRAM PAGE 0    /**erase() and program()*****/
                                     /*****from flash.asm file****/
   fl_prg : {} > EXTRAM    PAGE 0    /**Programming Algorithm*****/
   fl_clr : {} > EXTRAM    PAGE 0    /******Clear Algorithm*******/
   fl_ers : {} > EXTRAM    PAGE 0    /******Erase Algorithm*******/
   fl_wrt : {} > EXTRAM    PAGE 0    /****Flash-write Algorithm***/
   DLY    : {} > EXTRAM    PAGE 0    /******Delay Subroutine******/
   REG    : {} > EXTRAM    PAGE 0    /*******Regs Subroutine******/
   ARY    : {} > EXTRAM    PAGE 0    /******Array Subroutine******/
   PRG_data  : {} > B1    PAGE 1     /*Reserved in flash.asm for**/
                                     /***flash algo variables.****/
   PRG_parm  : {} > B1    PAGE 1     /*Reserved in flash.asm for**/
                                     /*parameter passing to algos*/
      /*End of sections for flash programming.      */
```

```
    .bss      :{}  > B1 PAGE 1
    .cinit    :{}  > B1 PAGE 1
    .const    : load = EXTRAM PAGE 0, run = DSRAM PAGE 1
          {
             /* GET RUN ADDRESS  */
             __const_run = .;
             /* MARK LOAD ADDRESS */
             *(.c_mark)
             /* ALLOCATE .const  */
             *(.const)
             /* COMPUTE LENGTH   */
             __const_length = .- __const_run;
          }
    .data   :{}  > B1 PAGE 1
    .stack :{}  > B1 PAGE 1    /*C stack.                */

}
```

A.6.3 C Function for Disabling TMS320F240 Watchdog Timer

```
*************************************************************************
*  Watchdog Timer Disable function                                      *
*  Arguments passed from Caller: None                                   *
*  Local Variables:           None                                      *
*************************************************************************
SYSSR  .set   0701Ah    ; System Module Status Register
WDCR   .set   07029h    ; WDT Control reg
DP_PF1 .set 224         ; 7000h/80h = 100h or 224
          .globl _wdtoff
          .text
          .def _wdtoff
_wdtoff:                ; presume ARP = AR1 (SP)
*************************************************************************
*  On entry, presume ARP = AR1 (SP)                                     *
*                                                                       *
*  Step 1. Pop the return address off the h/w stack and push to s/w stack   *
*************************************************************************
          POPD *+               ; pop return address, push on software stack
                                ; ARP=AR1, SP=SP+1
*************************************************************************
*  Step 2. Push the frame pointer onto s/w stack                        *
*************************************************************************
          SAR    AR0,*+    ; push AR0 (FP) onto SP
                           ; ARP=AR1, SP=SP+2
*************************************************************************
*  Step 3. Allocate the local frame                                     *
*************************************************************************
          SAR    AR1,*         ; *SP = FP
          LAR    AR0,#1        ; FP = size of local frame, 1
          LAR    AR0,*0+       ; FP = SP, SP += size ==> allocate frame
*************************************************************************
*  Step 5. Begin code that will disable the WDT                         *
*************************************************************************
       LDP#DP_PF1               ; Page DP_PF1 includes WET through EINT frames
          LACL   WDCR           ; ACC = WDTCR, watchdog timer control register
```

```
          OR     #06fh         ; set WDDIS bit and WDCHK2:0 bits, WDCLK to max.
          SACL   WDCR          ; write ACC out to WDTCR
*************************************************************************
*  Step 9. Deallocate the local frame                                   *
*************************************************************************
          SBRK 1+1         ; deallocate frame, point to saved FP
*************************************************************************
*  Step 10. restore the frame pointer                                   *
*************************************************************************
          LAR  AR0,*–       ; pop FP
*************************************************************************
*  Step 11. copy the return address from the s/w stack and push onto h/w *
*         stack                                                         *
*************************************************************************
          PSHD *           ; push return address on h/w stack
          RET              ; return
          .en
```

A.6.4 C Functions for Initializing the TMS320F240

```
*************************************************************************
*  TMS320x240 Initialization Function                                   *
*  Arguments passed from Caller: None                                   *
*  Local Variables:           None                                      *
*************************************************************************
SYSSR     .set   0701Ah
SYSCR     .set   07018h
WDTCR     .set   07029h    ;WD Control reg
CKCR0     .set   0702ah    ;PLL Clock Control Register 0
CKCR1     .set   0702ch    ;PLL Clock Control Register 1
DP_PF1    .set   224
          .globl _c240init
          .text
          .def   _c240init
_c240init:                 ; presume ARP = AR1 (SP)**
*************************************************************************
*  On entry, presume ARP = AR1 (SP)                                     *
*                                                                       *
*  Step 1. pop the return address off the h/w stack and push to s/w stack   *
*************************************************************************
          POPD *+          ; pop return address, push on software stack
                           ; ARP=AR1, SP=SP+1
*************************************************************************
*  Step 2. push the frame pointer onto s/w stack                        *
*************************************************************************
          SAR  AR0,*+      ; push AR0 (FP) onto SP
                           ; ARP=AR1, SP=SP+2
*************************************************************************
*  Step 3. Allocate the local frame                                     *
*************************************************************************
          SAR  AR1,*           ; *SP = FP
          LAR  AR0,#1          ; FP = size of local frame, 1
          LAR  AR0,*0+         ; FP = SP, SP += size ==> allocate frame
*************************************************************************
```

```
*   Step 5. begin code that will initialize the '240 registers                *
*****************************************************************************
       CLRC   SXM       ; Clear Sign Extension Mode
       CLRC   OVM       ; Reset Overflow Mode
*   Set Data Page pointer to  page 1 of the peripheral frame
       LDP #DP_PF1      ; Page DP_PF1 includes WET through EINT frames
*   Clear system status register reset bits (PORRST, ILLADR, SWRST, & WDRST)
          LACL #020h       ; load mask pattern to clear rst flags
          SACL SYSSR       ; write ACC to SYSSR
*   Set Watchdog timer period to 1 second
          LACL #02Fh       ; set WDCHK2 & 0 bits, WDCLK divider to max (1s)
          SACL WDTCR       ; write ACC out to WDTCR
*   Configure PLL for 4–MHz xtal, 10–MHz SYSCLK, and 20–MHz CPUCLK
*      SPLK   #00E4h,CKCR1  ;CLKIN(XTAL)=4 MHz,CPUCLK=20 MHz
*      SPLK   #00C3h,CKCR0  ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2
*   Configure PLL for 10–MHz osc, 10–MHz SYSCLK, and 20–MHz CPUCLK
       SPLK   #00B1h,CKCR1  ;CLKIN(OSC)=10 MHz,CPUCLK=20 MHz
       SPLK   #00C3h,CKCR0   ;CLKMD=PLL Enable,SYSCLK=CPUCLK/2
*   Set VCCAON bit and CLKSRC1:0; leave other bits at their reset values.
       SPLK   #40C8h,SYSCR ; SYSCR <= 40C8h
*****************************************************************************
*   Step 9. Deallocate the local frame                                      *
*****************************************************************************
       SBRK 1+1         ; deallocate frame, point to saved FP
*****************************************************************************
*   Step 10. restore the frame pointer                                      *
*****************************************************************************
       LAR  AR0,*–       ; pop FP
*****************************************************************************
*   Step 11. copy the return address from the s/w stack and push onto h/w *
*   stack                                                                   *
*****************************************************************************
       PSHD *        ; push return address on h/w stack
       RET           ; return
       .en
```

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

R

S

T

U

V

W

TMS320x281x, 280x DSP Peripheral Reference Guide

Literature Number: SPRU566B
June 2003 − Revised November 2004

Texas Instruments

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third−party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products & application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

Contents

Tables

TMS320x281x, 280x DSP Peripherals

This overview guide describes all the peripherals available for TMS320x28x devices. Table 1 shows the peripherals used by each device. You can download the peripheral guide by clicking on the literature number, which is linked to the portable document format (pdf) file.

Note:

The 281x EV, SCI, and SPI peripherals are almost identical to those available on the 240x™ devices; however, these modules have enhanced features. See the respective documents for details.

Brief descriptions of the peripherals are included in the following sections.

Table 1. Peripheral Selection Guide

Peripheral	Lit. No.	2812 Device	2811 Device	2810 Device	2808 Device	2806 Device	2801 Device
TMS320x281x System Control and Interrupts	SPRU078	✓	✓	✓			
TMS320x280x System Control and Interrupts	SPRU712				✓	✓	✓
TMS320x281x External Interface (XINTF)	SPRU067	✓					
TMS320x281x, 280x Enhanced Controller Area Network (eCAN)	SPRU074	✓	✓	✓	✓	✓	✓
TMS320x281x Event Manager (EV)	SPRU065	✓	✓	✓			
TMS320x281x Analog-to-Digital Converter (ADC)	SPRU060	✓	✓	✓			
TMS320x280x Analog-to-Digital Converter (ADC)	SPRU716				✓	✓	✓
TMS320x281x Multichannel Buffered Serial Port (McBSP)	SPRU061	✓	✓	✓			
TMS320x281x, 280x Serial Communications Interface (SCI)	SPRU051	✓	✓	✓	✓	✓	✓
TMS320x281x, 280x Serial Peripheral Interface (SPI)	SPRU059	✓	✓	✓	✓	✓	✓
TMS320x281x Boot ROM	SPRU095	✓	✓	✓			
TMS320x280x Boot ROM	SPRU722				✓	✓	✓
TMS320x280x Enhanced Quadrature Encoder Pulse (eQEP)	SPRU790				✓	✓	✓
TMS320x280x Enhanced PWM Module	SPRU791				✓	✓	✓
TMS320x280x Enhanced Capture (eCAP) Module	SPRU807				✓	✓	✓
TMS320x280x Inter-Integrated Circuit (I^2C)	SPRU721				✓	✓	✓

1.1 System Control and Interrupts (different 281x and 280x versions)

The *TMS320x281x System Control and Interrupts Peripheral Guide* (SPRU078) and the *TMS320x280x System Control and Interrupts Peripheral Guide* (SPRU712) include information on the following modules:

- ❑ Memory, including Flash and OTP configuration
- ❑ Code security module (CSM)
- ❑ Clocking, low power modes, watchdog, and CPU-timers
- ❑ General-purpose inputs/outputs (GPIO)
- ❑ Peripheral frames
- ❑ Peripheral interrupt expansion (PIE)

1.1.1 Code Security Module (CSM)

Security is defined with respect to the access of the on-chip program memory and prevents unauthorized copying of proprietary code. The code security module (CSM) blocks access to several on-chip program memory blocks.

1.1.2 Peripheral Clocking

The clocks to each individual peripheral can be enabled/disabled so as to reduce power consumption when a peripheral is not in use. Additionally, the system clock to the serial ports and the event managers, CAP and QEP blocks can be scaled relative to the CPU clock. This enables the timing of peripherals to be decoupled from increasing CPU clock speeds.

1.1.2.1 CPU-Timers

CPU-Timers 0, 1, and 2 are identical 32-bit timers with presettable periods and with 16-bit clock prescaling. The timers have a 32-bit count down register, which generates an interrupt when the counter reaches zero. The counter is decremented at the CPU clock speed divided by the prescale value setting. When the counter reaches zero, it is automatically reloaded with a 32-bit period value. CPU-Timers 1 and 2 are reserved for Real-Time OS (RTOS) applications. CPU-Timer 2 is connected to INT14 of the CPU. CPU-Timer 1 can be connected to INT13 of the CPU. CPU-Timer 0 is for general use and is connected to the PIE block.

1.1.2.2 Watchdog Timer

The 28x devices support a watchdog timer. The user software must regularly reset the watchdog counter within a certain time frame; otherwise, the

watchdog generates a reset to the processor. The watchdog can be disabled if necessary.

1.1.3 General-Purpose Input/Output (GPIO) Multiplexer

Most of the peripheral signals are multiplexed with general-purpose I/O (GPIO) signals. This enables you to use a pin as GPIO if the peripheral signal or function is not used. On reset, all GPIO pins are configured as inputs. You can then individually program each pin for GPIO mode or Peripheral Signal mode. For specific inputs, you can also select the number of input qualification cycles to filter unwanted noise glitches.

1.1.4 Peripheral Frames

The 281x and 280x devices contain three peripheral register spaces. Some registers within these frames can be protected from CPU writes by the EALLOW protection mechanism.

1.1.5 Peripheral Interrupt Expansion (PIE) Block

The PIE block multiplexes numerous interrupt sources into a smaller set of interrupt inputs. The interrupts are grouped into blocks of eight and each group is fed into one of 12 CPU interrupt lines ($\overline{\text{INT1}}$ to $\overline{\text{INT12}}$). Each of the 96 interrupts is supported by its own vector stored in a dedicated RAM block that can be overwritten by the user. The vector is automatically fetched by the CPU on servicing the interrupt. It takes nine CPU clock cycles to fetch the vector and save critical CPU registers. Therefore, the CPU can respond quickly to interrupt events. Prioritization of interrupts is controlled in hardware and software. Each individual interrupt can be enabled/disabled within the PIE block.

1.2 TMS320x281x External Interface (XINTF) (SPRU067)

The external interface (XINTF) on the 2812 device is a nonmultiplexed asynchronous bus that is used to interface to external devices and memory.

1.3 TMS320x281x, 280x Enhanced Controller Area Network (eCAN) (SPRU074)

This is the enhanced version of the CAN peripheral. It supports 32 mailboxes, time stamping of messages, and is CAN 2.0B-compliant.

1.4 TMS320x281x Event Manager (EV) (SPRU065)

The event manager module includes general-purpose timers, full-compare/pulse-width modulation (PWM) units, capture inputs (CAP) and

quadrature-encoder pulse (QEP) circuits. Two such event managers are provided, which enable two three-phase motors to be driven or four two-phase motors. The event managers on the F2810 and F2812 are compatible to the event managers on the 240x devices (with some minor enhancements).

1.5 TMS320x281x, 280x Analog-to-Digital Converter (ADC) (different 281x and 280x versions)

The ADC block is a 12-bit converter, single ended, 16-channels. It contains two sample-and-hold units for simultaneous sampling. See the *TMS320x281x Analog-to-Digital Converter (ADC) Reference Guide* (literature number SPRU060) or the *TMS320x280x Analog-to-Digital Converter (ADC) Reference Guide* (literature number SPRU712) for details on how to use the ADC for your device.

1.6 TMS320x281x Multichannel Buffered Serial Port (McBSP) (SPRU061)

The McBSP is used to connect to E1/T1 lines, phone-quality codecs for modem applications or high-quality stereo-quality Audio DAC devices. The McBSP receive and transmit registers are supported by a 16-level FIFO. This significantly reduces the overhead for servicing this peripheral.

1.7 TMS320x281x, 280x Serial Communications Interface (SCI) (SPRU051)

The SCI is a two-wire asynchronous serial port, commonly known as UART. On the F2810 and the F2812, the port supports a 16-level, receive and transmit FIFO for reducing servicing overhead.

1.8 TMS320x281x, 280x Serial Peripheral Interface (SPI) (SPRU059)

The SPI is a high-speed, synchronous serial I/O port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmable bit-transfer rate. Normally, the SPI is used for communications between the DSP controller and external peripherals or another processor. Typical applications include external I/O or peripheral expansion through devices such as shift registers, display drivers, and ADCs. Multi-device communications are supported by the master/slave operation of the SPI. The port supports a 16-level, receive and transmit FIFO for reducing servicing overhead.

1.9 Boot ROM (different 281x and 280x versions)

The boot ROM is factory-programmable with boot-loading software. Boot-mode signals (general-purpose I/Os) are used to tell the bootloader software

which mode to use. The Boot ROM also contains standard math tables such as SIN/COS for use in IQ math related algorithms. See the *TMS320x281x Boot ROM Reference Guide* (literature number SPRU095) or the *TMS320x280x Boot ROM Reference Guide* (SPRU722) for details.

1.10 TMS320x280x Inter-Integrated Circuit (I2C) (SPRU721)

This guide describes the features and operation of the inter-integrated circuit (I^2C) module that is available on the TMS320x280x digital signal processor (DSP). The I^2C module provides an interface between one of these DSPs and devices compliant with Philips Semiconductors Inter-IC bus (I^2C-bus) specification version 2.1 and connected by way of an I^2C-bus. External components attached to this 2-wire serial bus can transmit/receive 1- to 8-bit data to/from the 280x DSP through the I^2C module. This chapter assumes the reader is familiar with the I^2C-bus specification.

1.11 TMS320x280x Enhanced Quadrature Encoder Pulse (eQEP) Module (SPRU790)

The enhanced quadrature encoder pulse (eQEP) module is used for direct interface with a linear or rotary incremental encoder to get position, direction, and speed information from a rotating machine for use in a high-performance motion and position-control system

1.12 TMS320x280x Enhanced Capture (eCAP) Module (SPRU807)

The enhanced Capture (eCAP) Module is essential in systems where accurate timing of external events is important.

Uses for eCAP include:

- ❏ Speed measurements of rotating machinery (e.g., toothed sprockets sensed via Hall sensors)
- ❏ Elapsed time measurements between position sensor triggers
- ❏ Period and duty cycle measurements of pulse train signals
- ❏ Decoding current or voltage amplitude derived from duty cycle encoded current/voltage sensors

1.13 TMS320x280x Enhanced PWM (ePWM) Module (SPRU791)

The enhanced pulse width modulator (ePWM) peripheral controls many of the power-related systems found in both commercial and industrial equipments.

The main systems include digital motor control, switch mode power supply control, uninterruptible power supplies (UPS), and other forms of power conversion. The PWM peripheral performs a DAC function, where the duty cycle is equivalent to a DAC analog value; it is sometimes referred to as a Power DAC.

TMS320F2810, TMS320F2811, TMS320F2812

TMS320C2810, TMS320C2811, TMS320C2812

DSP Silicon Errata

SPRZ193H
January 2003
Revised December 2004

REVISION HISTORY

This revision history highlights the technical change made to the SPRZ193G errata to make it an SPRZ193H revision.

Scope: Added ROM devices (TMS320C2810, TMS320C2811, and TMS320C2812)

PAGE(s) NO.	ADDITIONS/CHANGES/DELETIONS

Contents

1 Introduction

This document describes the silicon updates to the functional specifications for the TMS320F2810, TMS320F2811, and TMS320F2812 digital signal processors. The updates are applicable to:

- TMS320C2812 and TMS320F2812 (179-ball MicroStar BGA™, GHH and ZHH suffix)
- TMS320C2812 and TMS320F2812 (176-pin low-profile quad flatpack [LQFP], PGF suffix)
- TMS320C2810, TMS320C2811, TMS320C2811, and TMS320F2811 (128-pin LQFP, PBK suffix))

1.1 Device and Development Tool Support Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all TMS320™ DSP devices and support tools. Each TMS320™ DSP commercial family member has one of three prefixes: TMX, TMP, or TMS. Texas Instruments recommends two of three possible prefix designators for support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMDX) through fully qualified production devices/tools TMDS).

Device development evolutionary flow:

TMX Experimental device that is not necessarily representative of the final device's electrical specifications

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification

TMS Fully qualified production device

Support tool development evolutionary flow:

TMDX Development-support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully qualified development-support product

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

"Developmental product is intended for internal evaluation purposes."

TMS devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (TMX or TMP) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

1.2 Device Markings

Figure 1 provides an example of the TMS320F281x and TMS320C281x device markings and defines each of the markings. The device revision can be determined by the symbols marked on the top of the package as shown in Figure 1. Some prototype devices may have markings different from those illustrated.

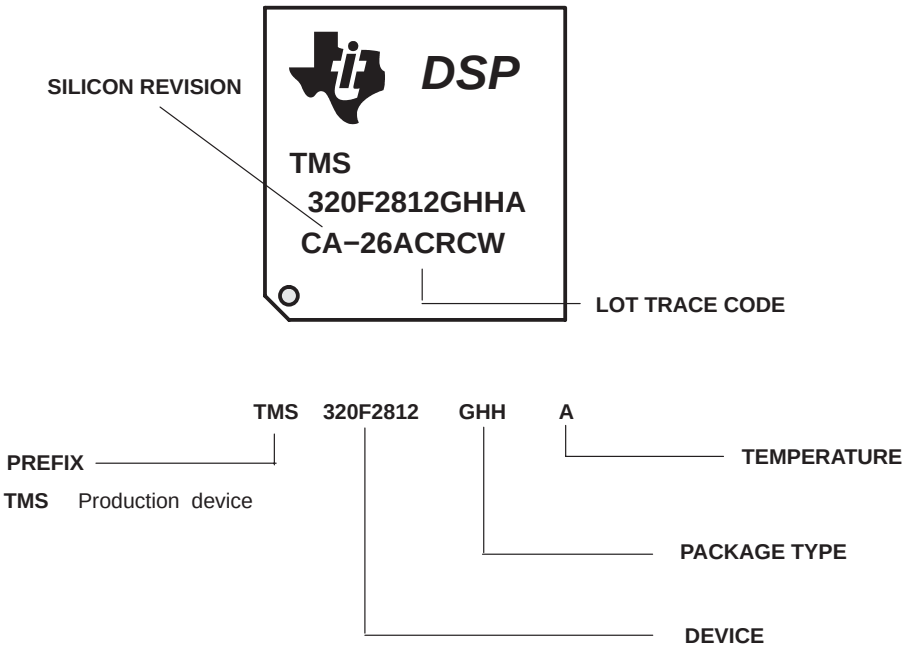


Figure 1. Example Markings for 281x Packages

Table 1. Determining Silicon Revision From Lot Trace Code (F281x)

Second Letter in Prefix of Lot Trace Code	Silicon Revision	Revision ID (0x0883)	Comments
Blank (no second letter in prefix)	Indicates Revision 0	0x0000	This silicon revision is available as TMX only.
A	Indicates Revision A	0x0001	This silicon revision is available as TMX only.
B	Indicates Revision B	0x0002	Internal
C	Indicates Revision C	0x0003	TMP/TMX/TMS
D	Indicates Revision D	0x0003	Internal
E	Indicates Revision E	0x0005	Production device (TMS)

2 Known Design Marginality/Exceptions to Functional Specifications

Advisory	*WD − WDFLAG Bit Does Not Work as Intended*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: The WDFLAG bit in F281x devices cannot be used to reliably distinguish a watchdog-initiated reset from a power-on (or warm) reset. This is because the device expects the $\overline{\text{XRS}}$ pin to be pulled high (by the external reset circuit) within 4 SYSCLKOUT cycles (8 OSCLK cycles at power up) after the end of the watchdog-initiated reset pulse, which is 512 OSCCLK cycles. Most of the external $\overline{\text{XRS}}$ circuits cannot provide the fast rise-time requirement (due to the capacitance); therefore, the WDFLAG bit should not be used in applications .

Workaround: None. This bit should not be used.

Advisory	*ADC − EOS BUF1/2 Bits in ADCST Corrupted at the End of Conversion of Sequencer 1/2 When INT MOD SEQ1/2 is Enabled*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: Setting the INT MOD SEQx bit in ADCTRL2 as per the user guide should result in the ADC wrapper generating INT SEQx at the end of every other conversion rather than at the end of every conversion. Also EOS BUFx will be set at the end of every conversion to track the status of the SEQx in use. However, a conversion on SEQ1 will cause EOS_BUF2 to be set if INT_MOD_SEQ2 is enabled for that sequencer, even if INT_MOD_SEQ1 is not enabled.

For example, if INT_MOD_SEQ2 is set, a conversion on SEQ1 will cause the EOS_BUF2 bit to be set incorrectly. This will cause INT SEQ2 to be set incorrectly after the next SEQ2 completion. If EOS_BUF2 is already set (from previous SEQ conversion), a conversion on SEQ1 will cause EOS_BUF2 to be cleared causing the interrupt to be missed. The above relationship is also true for SEQ2 affecting SEQ1. In all cases, the sequencers do work correctly, with the exception that EOS BUFx gets corrupted.

Workaround: Do not use the INT_MOD_SEQx feature if both SEQ1 and SEQ2 will be used before two completions of sequence have completed on the INT_MOD_SEQ selected sequencer.

Advisory

CAN − CPU Access to the eCAN Registers May Fail If It Is in Conflict With an eCAN Access to the eCAN Registers

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: If contention exists between the CPU and the eCAN controller for access to certain eCAN register areas, a CPU read may erroneously read all zeros (0x00000000), and a CPU write may erroneously fail to execute. Specifically:

1) If the CPU reads the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH registers) at the same time that the eCAN controller is accessing (reading or writing) the LAM/MOTO/MOTS register area, the CPU may erroneously read all zeros (0x00000000).

Workaround: For all CPU reads from the eCAN mailbox RAM area, check to see if the read returns all zeros. If so, the CPU should perform a second read. If the second read returns zero as well, then the data is correctly zero. If the second read returns a non-zero value, then the second data is the correct value. Note that interrupts must be disabled during the consecutive CPU reads. See Note 5.

Details: 2) If the CPU writes to the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH register) at the same time that the eCAN controller is accessing (reading or writing) the LAM/MOTO/MOTS register area, the CPU write may fail to execute.

Workaround: For all CPU writes to the eCAN mailbox RAM area, the CPU should write the data twice. Note that interrupts must be disabled during the consecutive CPU writes. See Note 5.

Details: 3) If the CPU reads the LAM/MOTO/MOTS register area at the same that the eCAN controller is accessing (reading or writing) the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH registers), the CPU may erroneously read all zeros (0x00000000).

Workaround: For all CPU reads from the LAM/MOTO/MOTS register area, check to see if the read returns all zeros. If so, the CPU should perform a second read. If the second read returns zero as well, then the data is correctly zero. If the second read returns a non-zero value, then the second data is the correct value. Note that interrupts must be disabled during the consecutive CPU reads. See Note 5.

Details: 4) If the CPU writes to the LAM/MOTO/MOTS register area at the same that the eCAN controller is accessing (reading or writing) the eCAN mailbox RAM area (MSGID, MSGCTRL, MDL, or MDH registers), the CPU write may fail to execute.

Workaround: For all CPU writes to the LAM/MOTO/MOTS register area, the CPU should write the data twice with a minimum of 4 CPU cycles in between the writes. Note that interrupts must be disabled during the consecutive CPU writes. See Note 5.

NOTES: 1. An example of the eCAN controller reading the LAM/MOTO/MOTS register area is a read of the LAMn register to check if a received message passes the acceptance mask filtering criterion. This happens during reception of a frame.
2. An example of the eCAN controller writing to the LAM/MOTO/MOTS register area is a write to the MOTSn register to update the time-stamp upon successful transmission of a frame.
3. An example for the eCAN controller attempting to read the mailbox RAM area (MSGID, MSGCTRL, MDL & MDH registers) is right before transmission.

CAN − CPU Access to the eCAN Registers May Fail If It Is in Conflict With an eCAN Access to the eCAN Registers (Continued)

4. An example for the eCAN controller attempting to write to the mailbox RAM area (MSGID, MSGCTRL, MDL & MDH registers) is right after reception.
5. A “C callable assembly” implementation of the workaround can be downloaded from the TI Website (literature number SPRC180).

Advisory *SPI − Slave-Mode Operation*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: When in slave mode, the SPI does not resynchronize received words based on SPISTE. A spurious SPICLK pulse could therefore throw the data stream out of sync.

Workaround: If the circuit board is not noisy enough to generate spurious SPICLK pulses, then this is not an issue. If noise is an issue, then the McBSP in SPI-slave mode may be used, since the McBSP resynchronizes on each new word.

Advisory *McBSP − Receive FIFO Read Conflict*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: The McBSP peripheral operates with or without FIFOs. The receive FIFO has interrupt generation logic that initiates interrupts based on the 5-bit FIFO status bits (12−8) and interrupt level bits (4−0) in the MFFRX register.

If the CPU reads the receive FIFO while the McBSP module writes new data into the FIFO, there is a potential conflict. The CPU read will not be stalled and read data will not be valid. The FIFO write gets the priority. The receive FIFO will be updated after every word is received in DRR2/1 registers. The DRR2/1 register update time will primarily depend on the word size and CLKR rate. For example for 8-bit word, it should be typically 8 times the CLKR cycle time. This conflict will be more pronounced if data transferred on the receive channel is back-to-back with no delays between words.

Workaround: The receive FIFOs should be read based on receive interrupts and within the next word receive time. To avoid the read conflict, additional checks could be used before initiating receive FIFO read. In most McBSP configurations, the FSR is a receiving sync pulse either active high or low (based on the FSR polarity bit) and will go inactive during word transfer time. These active and inactive phases can be detected by checking the FSR flag bit MCFFST (bit 3) register or checking the status of the FSR pin. See the FSR flag bit description for details.

Advisory *McBSP − Read Operations Decrement the McBSP FIFO*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: A read operation from any of the following locations will cause the McBSP receive FIFO contents to decrement by 1, as if the McBSP DRR1 register had been read:

- 0x7001 Reserved
- 0x7401 EV−A T1CNT
- 0x7C01 Reserved

The actual value read from the location is correct and is not affected by this issue.

Workaround(s):

1. Ensure that the McBSP receive FIFO is empty before performing any read operation from any of these addresses.
2. If McBSP traffic is common in the application and a timer count needs to be monitored, consider using a timer other then EV Timer1.

Advisory *SCI Bootloader Does Not Clear the ABD Bit Before Auto-Baud Lock*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: The SCI ROM bootloader code does not correctly clear the Auto-Baud Detect (ABD) bit in the SCIFFCT register before the auto-baud process begins. The bootloader code fragment is shown below:

```
// Prepare for autobaud detection
// Set the CDC bit to enable autobaud detection
// and clear the ABD bit
SCIARegs.SCIFFCT.all = 0x2000;
```

The comments incorrectly state that the ABD bit is cleared. The ABD bit is cleared by writing a 1 to the ABD_CLR bit (bit 14) of the SCIFFCT register. This situation does not hinder operation from power up or reset because the ABD bit is cleared by default after reset. If, however, the bootloader is invoked a second time from software, then the ABD bit will not be cleared and autobaud lock will not occur properly.

Workaround: If the bootloader is going to be re-invoked by software, the user's code must first clear the ABD bit before calling the bootloader. To do this, write a 1 to the ABD CLR bit (bit 14) in the SCIFFCT register.

Advisory *EV − QEP Circuit*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: After a DSP reset, the QEP module fails to detect the first transition that occurs on QEP input pins. (This problem also manifests itself when an external clock is used for the EV timers.) Therefore, if the first transition occurs after a GP timer has been initialized and enabled as the QEP counter (i.e., to use QEP as source of clock), the first transition will not be counted by the GP timer. The result is an error of one count in the GP timer out of a total of 1024 counts for a 256-line encoder, or 4096 counts for a 1024-line encoder. However, the issue is not a concern under any of the following conditions:

1. *The first transition happens **before** the GP timer is initialized and enabled as QEP counter.* This ensures that all transitions are counted after initialization.
2. *After the first index pulse is received and if the index pulse is used to recalibrate the GP Timer (through capture interrupt).* The recalibration corrects the error in the GP timer; therefore, from the time the first index pulse is received, the QEP counter becomes accurate.

Workaround(s):

1. Make the first transition happen before the GP timer is initialized and enabled as QEP counter. This is usually the case because typically the rotor shaft is locked to a known position before the GP timer is initialized. Locking the rotor shaft will generate transitions on QEP input pins, unless the rotor shaft is exactly aligned to the known position (which is a rare case). Disturbing the rotor shaft on purpose takes care of the rare case.
2. Use the index pulse of the encoder to recalibrate the GP timer used as QEP counter.
3. The counter has to be forced to count before the application actually uses the QEP. During initialization, configure the internal clock (HSPCLK) to be the counter source. After the first count is done, the counter should be reconfigured for external signals (QEP/TCLKIN) and reset to 0. Now the counter will also count the first edge of the QEP.

Advisory *Logic-High Level for XCLKIN Pin*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: This advisory is applicable only when an external oscillator is used to clock the device. The X1/XCLKIN pin is referenced to the core power supply (V_{DD}), rather than the 3.3-V I/O supply (V_{DDIO}). Therefore, the logic-high level for the input clock should not exceed V_{DD}. This requirement remains the same for future silicon revisions as well.

Workaround: A clamping diode may be used to clamp a buffered clock signal to ensure that the logic-high level does not exceed V_{DD} (1.8 V or 1.9 V). Otherwise, 1.8-V oscillators may be used.

Advisory — *ADC − Reserved Bits in Autosequence Status Register (ADCASEQSR)*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: SEQ2 STATE2−0 and SEQ1 STATE3−0 bit fields (bits 6 through 0) are the pointers of SEQ2 and SEQ1, respectively. These bits are reserved for TI testing and should not be used in customer applications.

Workaround: None

Advisory — *ADC Result Register Update Delay*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: The ADC result status flags INT_SEQ1 and INT_SEQ2 bit fields (bits 0 and 1 respectively) in the ADC_ST_FLG register indicate the availability of new ADC results after conversions and initiation of the ADC interrupts.

The update of the ADC result register requires one extra ADC cycle to complete after the status flags INT_SEQ1 and INT_SEQ2 bit(s) are set. The result of reading the result register prior to this extra cycle will result in old data being read (reset value/previous conversion result).

If auto-sequencers are enabled with a non-zero value in the MAXCONV register, the last result register update takes an additional ADC cycle from the time the INT_SEQ1 or INT_SEQ2 flag is set.

Workaround: Delay the read of the ADC result register(s) by at least one ADC clock period. This delay can be implemented by using software delay loops.

If the ADC result register(s) are read using the ADC interrupt, rather than polling, the wait period introduced by the ISR (interrupt service routine) could minimize the delay needed in software. This ISR branching delay is generally greater than 8 SYSCLKOUT cycles.

The ratio of the ADC clock (ADCCLK) to the CPU clock (SYSCLKOUT) determines the size of the software delay. For example, if ADCCLK = 10 MHz the software delay should be at least 100 ns.

Timing example to estimate the software delay:

1) Get the HSPCLK prescaler value − HISPCP
2) Get the ADCCLK prescaler value − ADCCLKPS
3) Get the CPS (ADCCTRL1[7]) value − CPS
4) Software wait-period in CPU cycles (SYSCLKOUT) before the ADC result register read is defined as:

```
Software wait  = (HISPCP *2) * (ADCCLKPS * 2) * (CPS +1)  cycles

If HISPCP  or ADCCLKPS is 0, then the respective terms should be (HISPCP +1)
or (ADCCLKPS+1)
```

Advisory	*ADC Sequencer Reset While Dual Sequencers Are Running*

Revision(s) Affected: TMS320F281x: 0, A, B, C, D, and E
TMS320C281x: 0, A

Details: In the TMS320F2812/TMS320F2811/TMS320F2810 on-chip ADC, there are two sequencers for performing ADC conversions; SEQ1 and SEQ2. If one of the sequencers is reset while the other sequencer is running, it will result in the running sequencer never completing its current sequence. The sequencer busy bit (bit 3/bit2 in ADCST register) for the sequencer will remain active and an “End-of-sequence (EOS)” interrupt for the running sequencer will never be generated. For example, if SEQ1 is reset while SEQ2 is performing a sequence, then SEQ2 will never complete.

Workaround: If dual sequencers are enabled, then the software handling the ADC module should make sure that SEQ1 BSY/ SEQ2 BSY bits are not set before performing a reset of either sequencer.

2.1 Advisory for TMS320F281x Revision D Only

Advisory	*DEVICE-ID Register of the Silicon*

Revision(s) Affected: TMS320F281x: D

Details: The DEVICE-ID register of the rev D silicon contains the same value (0x0003) as that of the rev C silicon.

Workaround: The next revision (Rev E) of the silicon has a DEVICE-ID value of 0x0005.

2.2 Advisories for TMS320F281x Revisions 0 and A Only

These revisions are not to be used in development or production.

Advisory	*ADC − Device Has Higher Gain Error Than the Design Goal of 1% FSR on All of the B0−B7 Channels*

Revision(s) Affected: TMS320F281x: 0 and A

Details: The device has a higher gain error than the design goal of 1% FSR on all of the B0−B7 channels. The gain error varies across channels A0−A7 and B0−B7.

Based on the current data obtained on B group channels, all B group channels show a uniform gain error as high as 2 to 3%.

Workaround: The channel-to-channel gain error data across channels are listed in Table 2. This should help in calibrating in software or hardware until the next revision of the silicon.

ADC − Device Has Higher Gain Error Than the Design Goal of 1% FSR on All of the B0−B7 Channels (Continued)

Table 2. Channel-to-Channel Offset Error Data Across Channels (176-Pin PGF)

ADC Channels	A0	A1	A2	A3	A4	A5	A6	A7	B0	B1	B2	B3	B4	B5	B6	B7
Gain Error in %	0.20	0.18	0.52	0.53	0.53	0.55	0.54	0.54	2.92	2.92	2.92	2.93	2.93	2.93	2.93	2.97
Offset in LSB Counts	14.80	15.64	4.86	9.82	5.82	−14.57	−13.98	−31.78	20.94	21.98	22.48	23.39	22.39	23.14	23.64	24.91

Note:

The data provided are typical values only. These values are obtained from bench characterization at room temperature on a few devices.

TMX samples are not fully screened for all ADC parameters. If there are devices that have worse performance than suggested issues/values, it is recommended that the part be replaced.

Advisory

ADC − Device Has Higher Offset Error Than the Design Goal (0.5 to 1%) on Some Channels

Revision(s) Affected: TMS320F281x: 0 and A

Details: Based on the current data obtained on all channels, some channels show an offset error as high as 1%.

Workaround: The channel-to-channel offset error data across channels are listed in Table 2. This should help in calibrating in software or hardware until the next revision of the silicon.

Note:

The data provided are typical values only. These values are obtained from bench characterization at room temperature on a few devices.

TMX samples are not fully screened for all ADC parameters. If there are devices that have worse performance than suggested issues/values, it is recommended that the part be replaced.

Advisory *ADC − Device Has Higher Non-Linearity Than the Design Goal of 2 LSBs*

Revision(s) Affected: TMS320F281x: 0 and A

Details: Based on the current data obtained on all channels, some channels show non-linearity as high as 12 LSBs in the mid-scale range. That is, the mid-range conversions will be off by about 12 LSB counts.

Workaround: This issue is corrected in the next revision of the silicon. The following option could be used to correct for INL errors only for the TMX Revision A silicon.

The INL issue is across all channels. Use the ADC results only for 9-bit data accuracy on this revision of the silicon and ignore the rest of the bits. This will mitigate the INL effect in the application provided the algorithm can tolerate 9-bit accuracy.

Note:

The data provided are typical values only. These values are obtained from bench characterization at room temperature on a few devices.

TMX samples are not fully screened for all ADC parameters. If there are devices that have worse performance than suggested issues/values, it is recommended that the part be replaced.

Advisory *XINTF − XREADY Signal is not Sampled Properly When Using Asynchronous Sampling Mode*

Revision(s) Affected: TMS320F281x: 0 and A

Details: In case of asynchronous ready mode, if the XREADY signal is high within the Lead period, then access will complete in the number of cycles programmed in LEAD + ACTIVE + TRAIL counters even if XREADY goes low before the start of the ACTIVE period. In this case, XREADY is not being used properly to extend the access.

Workaround: Try one of the following possible workarounds:

- Ensure that the XREADY signal is not low at the start of an access when using asynchronous sampling mode. If the XINTF sees the XREADY signal low from the start of an access, then the ACTIVE period will be extended as desired.
- Use the XTIMING register wait-state values to extend the access such that timings are met without using XREADY.
- Use the synchronous XREADY sampling mode. This problem is not observed in synchronous mode.

This issue is fixed in the next revision of the silicon.

Advisory — *Set Device Emulation Register Bits for On-Chip RAM Performance*

Revision(s) Affected: TMS320F281x: 0 and A

Details: To get the best performance of on-chip RAM blocks M0/M1/L0/L1/H0, the internal control register bits have to be enabled. The bits are in the Device Emulation Registers.

Workaround: All device initialization code should include the following register updates. These are EALLOW-protected registers.

Register Address	Value
0x950	0x0300
0x951	0x0300
0x952	0x0300
0x953	0x0300
0x954	0x0300

Code Example:

```
EALLOW
MOVL    XAR1,#0x0950
MOVL    XAR2,#0x0300
MOV     *XAR1++,AR2
MOV     *XAR1++,AR2
MOV     *XAR1++,AR2
MOV     *XAR1++,AR2
MOV     *XAR1++,AR2
EDIS
```

The Code Composer GEL init files will initialize these for emulation and debug environment. From the next silicon revision onward, this initialization is automatically done upon reset.

Advisory — *OTP Memory*

Revision(s) Affected: TMS320F281x: 0 and A

Details: The 1K-word OTP memory is not available.

Workaround: This is fixed in the next revision of the silicon.

Advisory — *A Low Output on GPIOF14 Can Disable the PLL and Watchdog if the Watchdog Fires a Reset*

Revision(s) Affected: TMS320F281x: 0 and A

Details: If, during program execution, the XF_$\overline{\text{XPLLDIS}}$/GPIOF14 signal is changed to either of the following:

- A general-purpose output and driven low
- The XF functionality and driven low

and a watchdog reset occurs, then the low output state of the XF_$\overline{\text{XPLLDIS}}$/GPIOF14 pin will be latched into the $\overline{\text{XPLLDIS}}$ signal. The result of this is that the PLL and the reset function of the watchdog will be disabled. The watchdog itself is not disabled.

A Low Output on GPIOF14 Can Disable the PLL and Watchdog if the Watchdog Fires a Reset (Continued)

Workaround: One of the following workarounds can be used:

- Do not toggle XF/GPIOF14 in user code. Instead, use another GPIO signal for status.
- Set the watchdog to fire an interrupt instead of reset.

This is fixed in the next revision of the silicon.

Advisory *PLL − PLL x4 and x8 Multiplier Ratios*

Revision(s) Affected: TMS320F281x: 0 and A

Details: When the PLL multiplier is set to x4 or x8 (by writing 0004 or 0008, respectively, in the PLLCR register), the watchdog is re-enabled and resets the device upon a WD overflow. With noisy board conditions, this problem may be observed with other PLL multipliers as well.

Workaround: Do not use these multiplier values for these revisions.
This is fixed in the next revision of the silicon.

Advisory *Low-Power Modes − STANDBY Mode*

Revision(s) Affected: TMS320F281x: 0 and A

Details: When the device is put into STANDBY mode, the watchdog is re-enabled and resets the device upon a WD overflow.

Workaround: Do not use the STANDBY mode for these revisions.
This is fixed in the next revision of the silicon.

3 Documentation Support

For device-specific data sheets and related documentation, visit the TI web site at: **http://www.ti.com**

To access documentation on the web site:

1. Go to **http://www.ti.com**
2. Click on **DSP Product Tree**
3. Click on the **C2000** platform
4. Click on **C28x DSPs**
5. Click on a device name and then click on the documentation type you prefer.

For further information regarding the TMS320F2810 and TMS320F2812, please see the following publication:

- *TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, TMS320C2812 Digital Signal Processors* data manual (literature number SPRS174)

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

TMS320F2810, TMS320F2811, TMS320F2812
TMS320C2810, TMS320C2811, TMS320C2812
Digital Signal Processors

Data Manual

Literature Number: SPRS174L
April 2001 − Revised December 2004

TEXAS INSTRUMENTS

IMPORTANT NOTICE

Texas Instruments Incorporated and its subsidiaries (TI) reserve the right to make corrections, modifications, enhancements, improvements, and other changes to its products and services at any time and to discontinue any product or service without notice. Customers should obtain the latest relevant information before placing orders and should verify that such information is current and complete. All products are sold subject to TI's terms and conditions of sale supplied at the time of order acknowledgment.

TI warrants performance of its hardware products to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are used to the extent TI deems necessary to support this warranty. Except where mandated by government requirements, testing of all parameters of each product is not necessarily performed.

TI assumes no liability for applications assistance or customer product design. Customers are responsible for their products and applications using TI components. To minimize the risks associated with customer products and applications, customers should provide adequate design and operating safeguards.

TI does not warrant or represent that any license, either express or implied, is granted under any TI patent right, copyright, mask work right, or other TI intellectual property right relating to any combination, machine, or process in which TI products or services are used. Information published by TI regarding third-party products or services does not constitute a license from TI to use such products or services or a warranty or endorsement thereof. Use of such information may require a license from a third party under the patents or other intellectual property of the third party, or a license from TI under the patents or other intellectual property of TI.

Reproduction of information in TI data books or data sheets is permissible only if reproduction is without alteration and is accompanied by all associated warranties, conditions, limitations, and notices. Reproduction of this information with alteration is an unfair and deceptive business practice. TI is not responsible or liable for such altered documentation.

Resale of TI products or services with statements different from or beyond the parameters stated by TI for that product or service voids all express and any implied warranties for the associated TI product or service and is an unfair and deceptive business practice. TI is not responsible or liable for any such statements.

Following are URLs where you can obtain information on other Texas Instruments products and application solutions:

Products		**Applications**	
Amplifiers	amplifier.ti.com	Audio	www.ti.com/audio
Data Converters	dataconverter.ti.com	Automotive	www.ti.com/automotive
DSP	dsp.ti.com	Broadband	www.ti.com/broadband
Interface	interface.ti.com	Digital Control	www.ti.com/digitalcontrol
Logic	logic.ti.com	Military	www.ti.com/military
Power Mgmt	power.ti.com	Optical Networking	www.ti.com/opticalnetwork
Microcontrollers	microcontroller.ti.com	Security	www.ti.com/security
		Telephony	www.ti.com/telephony
		Video & Imaging	www.ti.com/video
		Wireless	www.ti.com/wireless

Mailing Address: Texas Instruments
Post Office Box 655303 Dallas, Texas 75265

REVISION HISTORY

This data sheet revision history highlights the technical changes made to the SPRS174K device-specific data sheet to make it an SPRS174L revision.

Global change: Added the ZHH package

PAGE NO.	ADDITIONS/CHANGES/DELETIONS
15	Changed notes on Table 2−1
15	Added the ZHH package to Table 2−1 and changed the Product Status from TMX to TMS
16	Added the ZHH package to Section 2.3
19	Added a line to the description of XCLKOUT in Table 2−2
37	Added new Table 3−4, Boot Mode Selection
47	Modified the note on interrupts in Table 3−11
64	Modified the equation in section 4.3
75	Changed PCR1 to PCR in Table 4−7
85	Modified Figure 4−12 and changed title from “Modes of Operation” to “GPIO/Peripheral Pin Multiplexing”
87	Added ZHH package to Figure 5−1
87	Updated section 5.2, Documentation Support
92	Added a new row and two footnotes to the Electrical Characterisitcs Over Recommended Operating Conditions table (section 6.3)
92	Added VDD1 to the first footnote of the Current Consumption by Power-Supply Pins Over Recommended Operating Conditions During Low-Power Modes at 150-MHz SYSCLKOUT (TMS320F281x) table (section 6.4)
93	Added new values to the Current Consumption by Power-Suply Pins Over Recommended Operating Cnditions During Low-Power Modes at 150-MHz SYSCLKOUT (TMS320C281x) (section 6.5)
94−95	Added Figure 6−1−through Figure 6−4
107	Added Table 6−10 as part of splitting IDLE mode timing requirements and switiching characteristics into two separate tables
107	Added Table 6−12 as part of splitting STANDBY mode timing requirements and switiching characteristics into two separate tables
108	Added Table 6−14 as part of splitting HALT mode timing requirements and switiching characteristics into two separate tables
153	Added new Table 6−60 ROM Access Timing
153	Added new Table 6−61 Minimum Required Wait-States at Different Frequencies (C281x devices)
154	Added section 6.33 Migrating From F281x to C281x Devices

This page is intentionally left blank.

Contents

List of Figures

List of Tables

Table *Page*

This page is intentionally left blank.

1 Features

- High-Performance Static CMOS Technology
  - 150 MHz (6.67-ns Cycle Time)
  - Low-Power (1.8-V Core @135 MHz, 1.9-V Core @150 MHz, 3.3-V I/O) Design
- JTAG Boundary Scan Support†
- High-Performance 32-Bit CPU (TMS320C28x)
  - 16 x 16 and 32 x 32 MAC Operations
  - 16 x 16 Dual MAC
  - Harvard Bus Architecture
  - Atomic Operations
  - Fast Interrupt Response and Processing
  - Unified Memory Programming Model
  - 4M Linear Program/Data Address Reach
  - Code-Efficient (in C/C++ and Assembly)
  - TMS320F24x/LF240x Processor Source Code Compatible
- On-Chip Memory
  - Flash Devices: Up to 128K x 16 Flash (Four 8K x 16 and Six 16K x 16 Sectors)
  - ROM Devices: Up to 128K x 16 ROM
  - 1K x 16 OTP ROM
  - L0 and L1: 2 Blocks of 4K x 16 Each Single-Access RAM (SARAM)
  - H0: 1 Block of 8K x 16 SARAM
  - M0 and M1: 2 Blocks of 1K x 16 Each SARAM
- Boot ROM (4K x 16)
  - With Software Boot Modes
  - Standard Math Tables
- External Interface (2812)
  - Up to 1M Total Memory
  - Programmable Wait States
  - Programmable Read/Write Strobe Timing
  - Three Individual Chip Selects
- Clock and System Control
  - Dynamic PLL Ratio Changes Supported
  - On-Chip Oscillator
  - Watchdog Timer Module
- Three External Interrupts
- Peripheral Interrupt Expansion (PIE) Block That Supports 45 Peripheral Interrupts
- Three 32-Bit CPU-Timers
- 128-Bit Security Key/Lock
  - Protects Flash/ROM/OTP and L0/L1 SARAM
  - Prevents Firmware Reverse Engineering
- Motor Control Peripherals
  - Two Event Managers (EVA, EVB)
  - Compatible to 240xA Devices
- Serial Port Peripherals
  - Serial Peripheral Interface (SPI)
  - Two Serial Communications Interfaces (SCIs), Standard UART
  - Enhanced Controller Area Network (eCAN)
  - Multichannel Buffered Serial Port (McBSP)
- 12-Bit ADC, 16 Channels
  - 2 x 8 Channel Input Multiplexer
  - Two Sample-and-Hold
  - Single/Simultaneous Conversions
  - Fast Conversion Rate: 80 ns/12.5 MSPS
- Up to 56 General Purpose I/O (GPIO) Pins
- Advanced Emulation Features
  - Analysis and Breakpoint Functions
  - Real-Time Debug via Hardware
- Development Tools Include
  - ANSI C/C++ Compiler/Assembler/Linker
  - Code Composer Studio™ IDE
  - DSP/BIOS™
  - JTAG Scan Controllers†
- Low-Power Modes and Power Savings
  - IDLE, STANDBY, HALT Modes Supported
  - Disable Individual Peripheral Clocks
- Package Options
  - 179-Ball MicroStar BGA™ With External Memory Interface (GHH), (ZHH) (2812)
  - 176-Pin Low-Profile Quad Flatpack (LQFP) With External Memory Interface (PGF) (2812)
  - 128-Pin LQFP Without External Memory Interface (PBK) (2810, 2811)
- Temperature Options:
  - A: −40°C to 85°C (GHH, ZHH, PGF, PBK)
  - S/Q: −40°C to 125°C (GHH, ZHH, PGF, PBK)

TMS320C24x, Code Composer Studio, DSP/BIOS, and MicroStar BGA are trademarks of Texas Instruments.
† IEEE Standard 1149.1−1990, IEEE Standard Test-Access Port

2 Introduction

This section provides a summary of each device's features, lists the pin assignments, and describes the function of each pin. This document also provides detailed descriptions of peripherals, electrical specifications, parameter measurement information, and mechanical data about the available packaging.

2.1 Description

The TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, and TMS320C2812 devices, members of the TMS320C28x™ DSP generation, are highly integrated, high-performance solutions for demanding control applications. The functional blocks and the memory maps are described in Section 3, Functional Overview.

Throughout this document, TMS320F2810, TMS320F2811, and TMS320F2812 are abbreviated as F2810, F2811, and F2812, respectively. F281x denotes all three Flash devices. TMS320C2810, TMS320C2811, and TMS320C2812 are abbreviated as C2810, C2811, and C2812, respectively. C281x denotes all three ROM devices. 2810 denotes both F2810 and C2810 devices; 2811 denotes both F2811 and C2811 devices; and 2812 denotes both F2812 and C2812 devices.

2.2 Device Summary

Table 2−1 provides a summary of each device's features.

Table 2−1. Hardware Features†

FEATURE		F2810	F2811	F2812	C2810	C2811	C2812
Instruction Cycle (at 150 MHz)		6.67 ns	6.67 ns	6.67 ns	6.67 ns	6.67 ns	6.67 ns
Single-Access RAM (SARAM) (16-bit word)		18K	18K	18K	18K	18K	18K
3.3-V On-Chip Flash (16-bit word)		64K	128K	128K	—	—	—
On-Chip ROM (16-bit word)		—	—	—	64K	128K	128K
Code Security for On-Chip Flash/SARAM/OTP/ROM		Yes	Yes	Yes	Yes	Yes	Yes
Boot ROM		Yes	Yes	Yes	Yes	Yes	Yes
OTP ROM (1K X 16)		Yes	Yes	Yes	Yes‡	Yes‡	Yes‡
External Memory Interface		—	—	Yes	—	—	Yes
Event Managers A and B (EVA and EVB)		EVA, EVB	EVA, EVB	EVA, EVB	EVA, EVB	EVA, EVB	EVA, EVB
• General-Purpose (GP) Timers		4	4	4	4	4	4
• Compare (CMP)/PWM		16	16	16	16	16	16
• Capture (CAP)/QEP Channels		6/2	6/2	6/2	6/2	6/2	6/2
Watchdog Timer		Yes	Yes	Yes	Yes	Yes	Yes
12-Bit ADC		Yes	Yes	Yes	Yes	Yes	Yes
• Channels		16	16	16	16	16	16
32-Bit CPU Timers		3	3	3	3	3	3
SPI		Yes	Yes	Yes	Yes	Yes	Yes
SCIA, SCIB		SCIA, SCIB	SCIA, SCIB	SCIA, SCIB	SCIA, SCIB	SCIA, SCIB	SCIA, SCIB
CAN		Yes	Yes	Yes	Yes	Yes	Yes
McBSP		Yes	Yes	Yes	Yes	Yes	Yes
Digital I/O Pins (Shared)		56	56	56	56	56	56
External Interrupts		3	3	3	3	3	3
Supply Voltage		1.8-V Core, (135 MHz) 1.9-V Core (150 MHz), 3.3-V I/O					
Packaging		128-pin PBK	128-pin PBK	179-ball GHH and ZHH 176-pin PGF	128-pin PBK	128-pin PBK	179-ball GHH and ZHH 176-pin PGF
Temperature Options§	A: −40°C to 85°C	Yes	Yes	Yes	Yes	Yes	Yes
	S/Q: −40°C to 125°C	Yes	Yes	Yes	Yes	Yes	Yes
Product Status¶		TMS	TMS	TMS	TMS	TMS	TMS

† The *TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, TMS320C2812 Digital Signal Processors Silicon Errata* (literature number SPRZ193) has been posted on the Texas Instruments (TI) website. It will be updated as needed.
‡ On C281x devices, OTP is replaced by a 1K X 16 block of ROM.
§ The S temperature option has been replaced by the Q temperature option (40°C to 125°C) from silicon revision E onwards. Q stands for −40°C to 125°C Q100 automotive fault grading.
¶ See Section 5.1, Device and Development Support Nomenclature for descriptions of device stages.

2.3 Pin Assignments

Figure 2−1 illustrates the ball locations for the 179-ball GHH and ZHH ball grid array (BGA) package. Figure 2−2 shows the pin assignments for the 176-pin PGF low-profile quad flatpack (LQFP) and Figure 2−3 shows the pin assignments for the 128-pin PBK LQFP. Table 2−2 describes the function(s) of each pin.

2.3.1 Terminal Assignments for the GHH Package

See Table 2−2 for a description of each terminal's function(s).

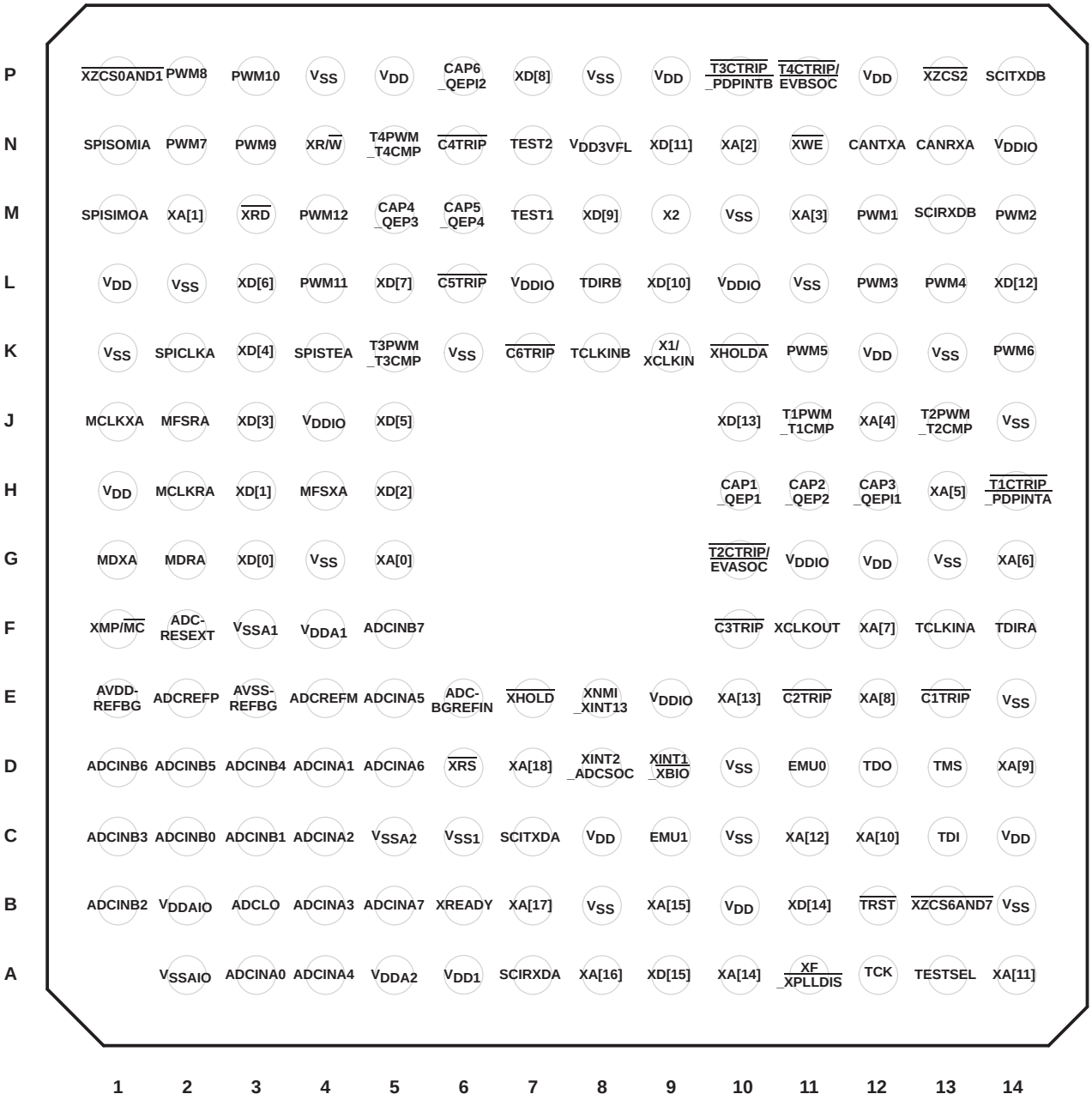


Figure 2−1. TMS320F2812 and TMS320C2812 179-Ball GHH MicroStar BGA™ (Bottom View)

2.3.2 Pin Assignments for the PGF Package

The TMS320F2812 and TMS320C2812 176-pin PGF low-profile quad flatpack (LQFP) pin assignments are shown in Figure 2−2. See Table 2−2 for a description of each pin's function(s).

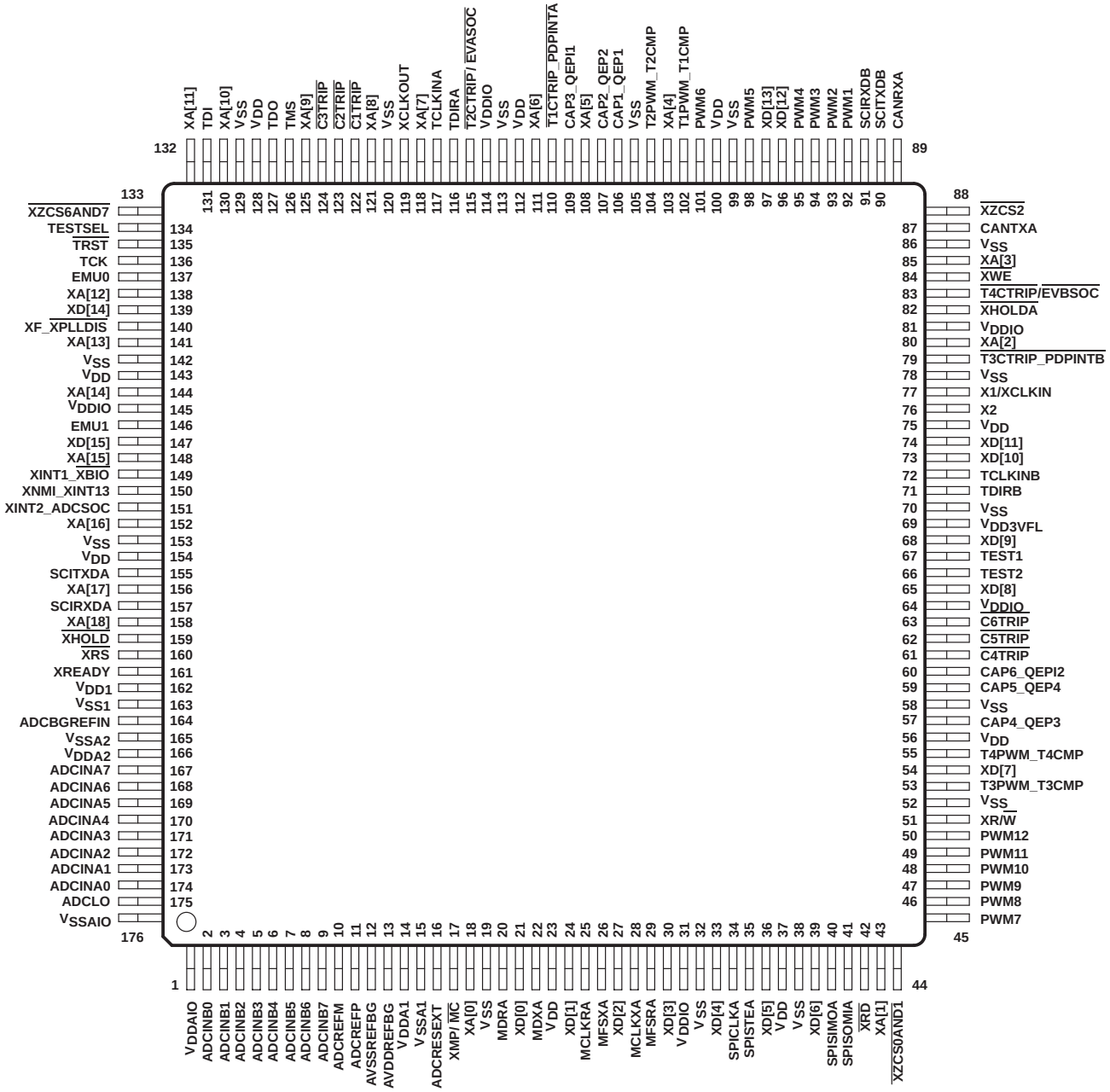


Figure 2−2. TMS320F2812 and TMS320C2812 176-Pin PGF LQFP (Top View)

2.3.3 Pin Assignments for the PBK Package

The TMS320F2810, TMS320F2811, TMS320C2810, and TMS320C2811 128-pin PBK low-profile quad flatpack (LQFP) pin assignments are shown in Figure 2−3. See Table 2−2 for a description of each pin's function(s).

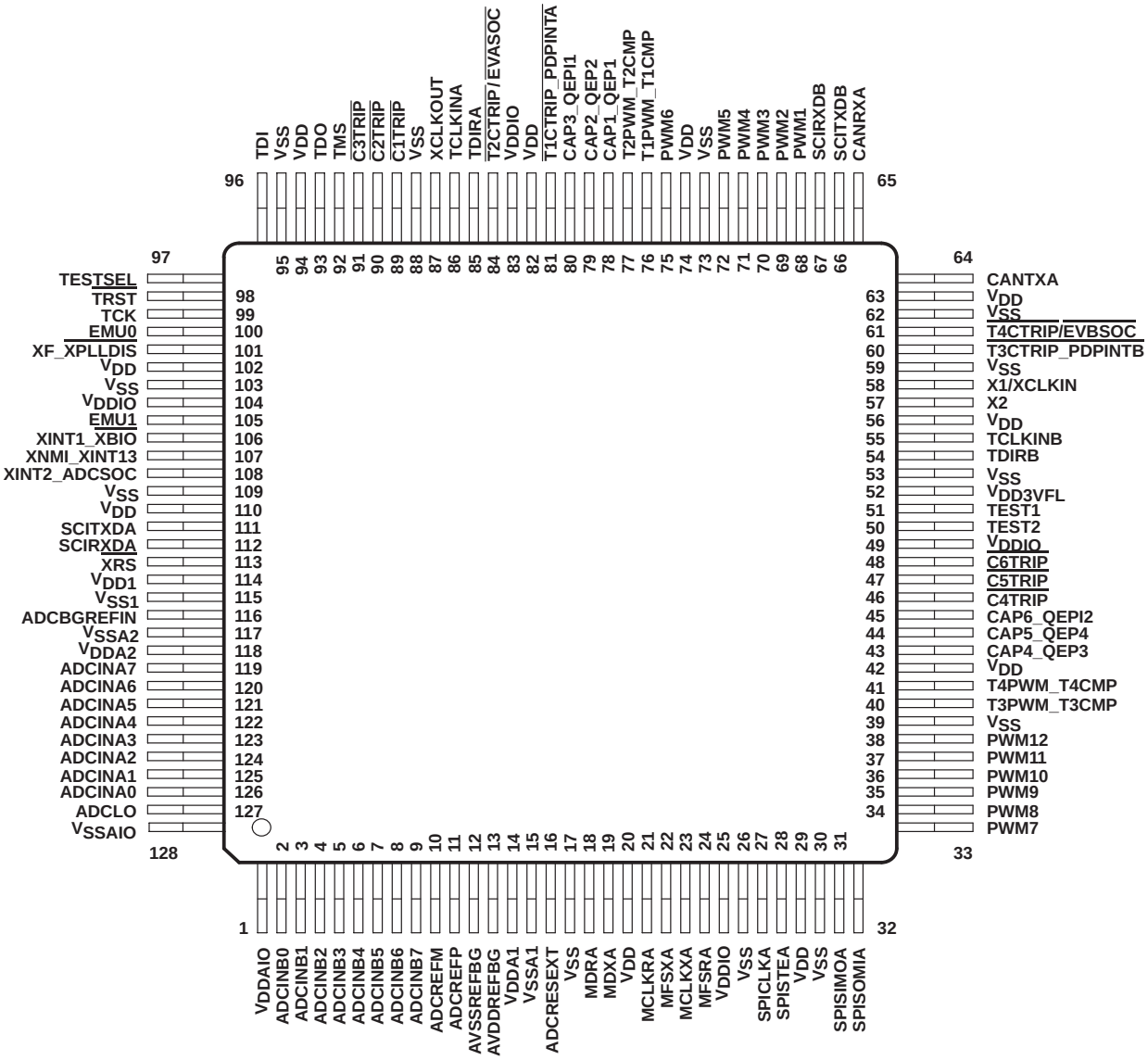


Figure 2−3. TMS320F2810, TMS320F2811, TMS320C2810, and TMS320C2811 128-Pin PBK LQFP (Top View)

2.4 Signal Descriptions

Table 2−2 specifies the signals on the F281x and C281x devices. All digital inputs are TTL-compatible. All outputs are 3.3 V with CMOS levels. Inputs are not 5-V tolerant. A 100-µA (or 20-µA) pullup/pulldown is used.

Table 2−2. Signal Descriptions†

NAME	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
	179-PIN GHH	176-PIN PGF	128-PIN PBK			
XINTF SIGNALS (2812 ONLY)						
XA[18]	D7	158	−	O/Z	−	19-bit XINTF Address Bus
XA[17]	B7	156	−	O/Z	−	
XA[16]	A8	152	−	O/Z	−	
XA[15]	B9	148	−	O/Z	−	
XA[14]	A10	144	−	O/Z	−	
XA[13]	E10	141	−	O/Z	−	
XA[12]	C11	138	−	O/Z	−	
XA[11]	A14	132	−	O/Z		
XA[10]	C12	130	−	O/Z	−	
XA[9]	D14	125	−	O/Z	−	
XA[8]	E12	121	−	O/Z	−	
XA[7]	F12	118	−	O/Z	−	
XA[6]	G14	111	−	O/Z	−	
XA[5]	H13	108	−	O/Z	−	
XA[4]	J12	103	−	O/Z	−	
XA[3]	M11	85	−	O/Z	−	
XA[2]	N10	80	−	O/Z	−	
XA[1]	M2	43	−	O/Z	−	
XA[0]	G5	18	−	O/Z		
XD[15]	A9	147	−	I/O/Z	PU	16-bit XINTF Data Bus
XD[14]	B11	139	−	I/O/Z	PU	
XD[13]	J10	97	−	I/O/Z	PU	
XD[12]	L14	96	−	I/O/Z	PU	
XD[11]	N9	74	−	I/O/Z	PU	
XD[10]	L9	73	−	I/O/Z	PU	
XD[9]	M8	68	−	I/O/Z	PU	
XD[8]	P7	65	−	I/O/Z	PU	
XD[7]	L5	54	−	I/O/Z	PU	
XD[6]	L3	39	−	I/O/Z	PU	
XD[5]	J5	36	−	I/O/Z	PU	
XD[4]	K3	33	−	I/O/Z	PU	
XD[3]	J3	30	−	I/O/Z	PU	
XD[2]	H5	27	−	I/O/Z	PU	
XD[1]	H3	24	−	I/O/Z	PU	
XD[0]	G3	21	−	I/O/Z	PU	

† Typical drive strength of the output buffer for all pins is 4 mA except for TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins, which are 8 mA.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2−2. Signal Descriptions† (Continued)

NAME	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
	179-PIN GHH	176-PIN PGF	128-PIN PBK			
XINTF SIGNALS (2812 ONLY) (CONTINUED)						
XMP/$\overline{MC}$	F1	17	−	I	PD	Microprocessor/Microcomputer Mode Select. Switches between microprocessor and microcomputer mode. When high, Zone 7 is enabled on the external interface. When low, Zone 7 is disabled from the external interface, and on-chip boot ROM may be accessed instead. This signal is latched into the XINTCNF2 register on a reset and the user can modify this bit in software. The state of the XMP/$\overline{MC}$ pin is ignored after reset.
$\overline{XHOLD}$	E7	159	−	I	PU	External Hold Request. $\overline{XHOLD}$, when active (low), requests the XINTF to release the external bus and place all buses and strobes into a high-impedance state. The XINTF will release the bus when any current access is complete and there are no pending accesses on the XINTF.
$\overline{XHOLDA}$	K10	82	−	O/Z	−	External Hold Acknowledge. $\overline{XHOLDA}$ is driven active (low) when the XINTF has granted a $\overline{XHOLD}$ request. All XINTF buses and strobe signals will be in a high-impedance state. $\overline{XHOLDA}$ is released when the $\overline{XHOLD}$ signal is released. External devices should only drive the external bus when $\overline{XHOLDA}$ is active (low).
$\overline{XZCS0AND1}$	P1	44	−	O/Z	−	XINTF Zone 0 and Zone 1 Chip Select. $\overline{XZCS0AND1}$ is active (low) when an access to the XINTF Zone 0 or Zone 1 is performed.
$\overline{XZCS2}$	P13	88	−	O/Z	−	XINTF Zone 2 Chip Select. $\overline{XZCS2}$ is active (low) when an access to the XINTF Zone 2 is performed.
$\overline{XZCS6AND7}$	B13	133	−	O/Z	−	XINTF Zone 6 and Zone 7 Chip Select. $\overline{XZCS6AND7}$ is active (low) when an access to the XINTF Zone 6 or Zone 7 is performed.
$\overline{XWE}$	N11	84	−	O/Z	−	Write Enable. Active-low write strobe. The write strobe waveform is specified, per zone basis, by the Lead, Active, and Trail periods in the XTIMINGx registers.
$\overline{XRD}$	M3	42	−	O/Z	−	Read Enable. Active-low read strobe. The read strobe waveform is specified, per zone basis, by the Lead, Active, and Trail periods in the XTIMINGx registers. NOTE: The $\overline{XRD}$ and $\overline{XWE}$ signals are mutually exclusive.
XR/$\overline{W}$	N4	51	−	O/Z	−	Read Not Write Strobe. Normally held high. When low, XR/$\overline{W}$ indicates write cycle is active; when high, XR/$\overline{W}$ indicates read cycle is active.

† Typical drive strength of the output buffer for all pins is 4 mA except for TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins, which are 8 mA.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2−2. Signal Descriptions† (Continued)

NAME	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
	179-PIN GHH	176-PIN PGF	128-PIN PBK			
XINTF SIGNALS (2812 ONLY) (CONTINUED)						
XREADY	B6	161	−	I	PU	Ready Signal. Indicates peripheral is ready to complete the access when asserted to 1. XREADY can be configured to be a synchronous or an asynchronous input. See the timing diagrams for more details.
JTAG AND MISCELLANEOUS SIGNALS						
X1/XCLKIN	K9	77	58	I		Oscillator Input − input to the internal oscillator. This pin is also used to feed an external clock. The 28x can be operated with an external clock source, provided that the proper voltage levels be driven on the X1/XCLKIN pin. It should be noted that the X1/XCLKIN pin is referenced to the 1.8-V (or 1.9-V) core digital power supply (V_{DD}), rather than the 3.3-V I/O supply (V_{DDIO}). A clamping diode may be used to clamp a buffered clock signal to ensure that the logic-high level does not exceed V_{DD} (1.8 V or 1.9 V) or a 1.8-V oscillator may be used.
X2	M9	76	57	O		Oscillator Output
XCLKOUT	F11	119	87	O	−	Output clock derived from SYSCLKOUT to be used for external wait-state generation and as a general-purpose clock source. XCLKOUT is either the same frequency, 1/2 the frequency, or 1/4 the frequency of SYSCLKOUT. At reset, XCLKOUT = SYSCLKOUT/4. The XCLKOUT signal can be turned off by setting bit 3 (CLKOFF) of the XINTCNF2 register to 1. Unlike other GPIO pins, the XCLKOUT pin is not placed in a high impedance state during reset.
TESTSEL	A13	134	97	I	PD	Test Pin. Reserved for TI. Must be connected to ground.
$\overline{\text{XRS}}$	D6	160	113	I/O	PU	Device Reset (in) and Watchdog Reset (out). Device reset. $\overline{\text{XRS}}$ causes the device to terminate execution. The PC will point to the address contained at the location 0x3FFFC0. When $\overline{\text{XRS}}$ is brought to a high level, execution begins at the location pointed to by the PC. This pin is driven low by the DSP when a watchdog reset occurs. During watchdog reset, the $\overline{\text{XRS}}$ pin will be driven low for the watchdog reset duration of 512 XCLKIN cycles. The output buffer of this pin is an open-drain with an internal pullup (100 μA, typical). It is recommended that this pin be driven by an open-drain device.
TEST1	M7	67	51	I/O	−	Test Pin. Reserved for TI. On F281x devices, TEST1 must be left unconnected. On C281x devices, this pin is a “no connect (NC)” (i.e., this pin is not connected to any circuitry internal to the device).
TEST2	N7	66	50	I/O	−	Test Pin. Reserved for TI. On F281x devices, TEST2 must be left unconnected. On C281x devices, this pin is a “no connect (NC)” (i.e., this pin is not connected to any circuitry internal to the device).

† Typical drive strength of the output buffer for all pins is 4 mA except for TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins, which are 8 mA.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2−2. Signal Descriptions† (Continued)

NAME	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
	179-PIN GHH	176-PIN PGF	128-PIN PBK			
JTAG						
$\overline{\text{TRST}}$	B12	135	98	I	PD	JTAG test reset with internal pulldown. $\overline{\text{TRST}}$, when driven high, gives the scan system control of the operations of the device. If this signal is not connected or driven low, the device operates in its functional mode, and the test reset signals are ignored. NOTE: Do not use pullup resistors on $\overline{\text{TRST}}$; it has an internal pulldown device. In a low-noise environment, $\overline{\text{TRST}}$ can be left floating. In a high-noise environment, an additional pulldown resistor may be needed. The value of this resistor should be based on drive strength of the debugger pods applicable to the design. A 2.2-kΩ resistor generally offers adequate protection. Since this is application-specific, it is recommended that each target board is validated for proper operation of the debugger and the application.
TCK	A12	136	99	I	PU	JTAG test clock with internal pullup
TMS	D13	126	92	I	PU	JTAG test-mode select (TMS) with internal pullup. This serial control input is clocked into the TAP controller on the rising edge of TCK.
TDI	C13	131	96	I	PU	JTAG test data input (TDI) with internal pullup. TDI is clocked into the selected register (instruction or data) on a rising edge of TCK.
TDO	D12	127	93	O/Z	−	JTAG scan out, test data output (TDO). The contents of the selected register (instruction or data) is shifted out of TDO on the falling edge of TCK.
EMU0	D11	137	100	I/O/Z	PU	Emulator pin 0. When $\overline{\text{TRST}}$ is driven high, this pin is used as an interrupt to or from the emulator system and is defined as input/output through the JTAG scan.
EMU1	C9	146	105	I/O/Z	PU	Emulator pin 1. When $\overline{\text{TRST}}$ is driven high, this pin is used as an interrupt to or from the emulator system and is defined as input/output through the JTAG scan.
ADC ANALOG INPUT SIGNALS						
ADCINA7	B5	167	119	I		8-Channel analog inputs for Sample-and-Hold A. The ADC pins should not be driven before V_{DDA1}, V_{DDA2}, and V_{DDAIO} pins have been fully powered up.
ADCINA6	D5	168	120	I		
ADCINA5	E5	169	121	I		
ADCINA4	A4	170	122	I		
ADCINA3	B4	171	123	I		
ADCINA2	C4	172	124	I		
ADCINA1	D4	173	125	I		
ADCINA0	A3	174	126	I		

† Typical drive strength of the output buffer for all pins is 4 mA except for TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins, which are 8 mA.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2−2. Signal Descriptions† (Continued)

NAME	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
	179-PIN GHH	176-PIN PGF	128-PIN PBK			
ADC ANALOG INPUT SIGNALS (CONTINUED)						
ADCINB7	F5	9	9	I		8-Channel Analog Inputs for Sample-and-Hold B. The ADC pins should not be driven before the V_{DDA1}, V_{DDA2}, and V_{DDAIO} pins have been fully powered up.
ADCINB6	D1	8	8	I		
ADCINB5	D2	7	7	I		
ADCINB4	D3	6	6	I		
ADCINB3	C1	5	5	I		
ADCINB2	B1	4	4	I		
ADCINB1	C3	3	3	I		
ADCINB0	C2	2	2	I		
ADCREFP	E2	11	11	I/O		ADC Voltage Reference Output (2 V). Requires a low ESR (50 mΩ − 1.5 Ω) ceramic bypass capacitor of 10 µF to analog ground. (Can accept external reference input (2 V) if the software bit is enabled for this mode. 1−10 µF low ESR capacitor can be used in the external reference mode.)
ADCREFM	E4	10	10	I/O		ADC Voltage Reference Output (1 V). Requires a low ESR (50 mΩ − 1.5 Ω) ceramic bypass capacitor of 10 µF to analog ground. (Can accept external reference input (1 V) if the software bit is enabled for this mode. 1−10 µF low ESR capacitor can be used in the external reference mode.)
ADCRESEXT	F2	16	16	O		ADC External Current Bias Resistor (24.9 kΩ ±5%)
ADCBGREFIN	E6	164	116	I		Test Pin. Reserved for TI. Must be left unconnected.
AVSSREFBG	E3	12	12	I		ADC Analog GND
AVDDREFBG	E1	13	13	I		ADC Analog Power (3.3-V)
ADCLO	B3	175	127	I		Common Low Side Analog Input. Connect to analog ground.
V_{SSA1}	F3	15	15	I		ADC Analog GND
V_{SSA2}	C5	165	117	I		ADC Analog GND
V_{DDA1}	F4	14	14	I		ADC Analog 3.3-V Supply
V_{DDA2}	A5	166	118	I		ADC Analog 3.3-V Supply
V_{SS1}	C6	163	115	I		ADC Digital GND
V_{DD1}	A6	162	114	I		ADC Digital 1.8-V (or 1.9-V) Supply
V_{DDAIO}	B2	1	1			3.3-V Analog I/O Power Pin
V_{SSAIO}	A2	176	128			Analog I/O Ground Pin

† Typical drive strength of the output buffer for all pins is 4 mA except for TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins, which are 8 mA.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2−2. Signal Descriptions† (Continued)

NAME	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
	179-PIN GHH	176-PIN PGF	128-PIN PBK			
POWER SIGNALS						
V_{DD}	H1	23	20			1.8-V or 1.9-V Core Digital Power Pins. See Section 6.2, Recommended Operating Conditions, for voltage requirements.
V_{DD}	L1	37	29			
V_{DD}	P5	56	42			
V_{DD}	P9	75	56			
V_{DD}	P12	−	63			
V_{DD}	K12	100	74			
V_{DD}	G12	112	82			
V_{DD}	C14	128	94			
V_{DD}	B10	143	102			
V_{DD}	C8	154	110			
V_{SS}	G4	19	17			Core and Digital I/O Ground Pins
V_{SS}	K1	32	26			
V_{SS}	L2	38	30			
V_{SS}	P4	52	39			
V_{SS}	K6	58	−			
V_{SS}	P8	70	53			
V_{SS}	M10	78	59			
V_{SS}	L11	86	62			
V_{SS}	K13	99	73			
V_{SS}	J14	105	−			
V_{SS}	G13	113	−			
V_{SS}	E14	120	88			
V_{SS}	B14	129	95			
V_{SS}	D10	142	−			
V_{SS}	C10	−	103			
V_{SS}	B8	153	109			
V_{DDIO}	J4	31	25			3.3-V I/O Digital Power Pins
V_{DDIO}	L7	64	49			
V_{DDIO}	L10	81	−			
V_{DDIO}	N14	−	−			
V_{DDIO}	G11	114	83			
V_{DDIO}	E9	145	104			
V_{DD3VFL}	N8	69	52			3.3-V Flash Core Power Pin. This pin should be connected to 3.3 V at all times after power-up sequence requirements have been met. This pin is used as VDDIO in ROM parts and must be connected to 3.3 V in ROM parts as well.

† Typical drive strength of the output buffer for all pins is 4 mA except for TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins, which are 8 mA.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2−2. Signal Descriptions† (Continued)

GPIO	PERIPHERAL SIGNAL	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
		179-PIN GHH	176-PIN PGF	128-PIN PBK			
GPIO OR PERIPHERAL SIGNALS							
GPIOA OR EVA SIGNALS							
GPIOA0	PWM1 (O)	M12	92	68	I/O/Z	PU	GPIO or PWM Output Pin #1
GPIOA1	PWM2 (O)	M14	93	69	I/O/Z	PU	GPIO or PWM Output Pin #2
GPIOA2	PWM3 (O)	L12	94	70	I/O/Z	PU	GPIO or PWM Output Pin #3
GPIOA3	PWM4 (O)	L13	95	71	I/O/Z	PU	GPIO or PWM Output Pin #4
GPIOA4	PWM5 (O)	K11	98	72	I/O/Z	PU	GPIO or PWM Output Pin #5
GPIOA5	PWM6 (O)	K14	101	75	I/O/Z	PU	GPIO or PWM Output Pin #6
GPIOA6	T1PWM_T1CMP (I)	J11	102	76	I/O/Z	PU	GPIO or Timer 1 Output
GPIOA7	T2PWM_T2CMP (I)	J13	104	77	I/O/Z	PU	GPIO or Timer 2 Output
GPIOA8	CAP1_QEP1 (I)	H10	106	78	I/O/Z	PU	GPIO or Capture Input #1
GPIOA9	CAP2_QEP2 (I)	H11	107	79	I/O/Z	PU	GPIO or Capture Input #2
GPIOA10	CAP3_QEPI1 (I)	H12	109	80	I/O/Z	PU	GPIO or Capture Input #3
GPIOA11	TDIRA (I)	F14	116	85	I/O/Z	PU	GPIO or Timer Direction
GPIOA12	TCLKINA (I)	F13	117	86	I/O/Z	PU	GPIO or Timer Clock Input
GPIOA13	$\overline{\text{C1TRIP}}$ (I)	E13	122	89	I/O/Z	PU	GPIO or Compare 1 Output Trip
GPIOA14	$\overline{\text{C2TRIP}}$ (I)	E11	123	90	I/O/Z	PU	GPIO or Compare 2 Output Trip
GPIOA15	$\overline{\text{C3TRIP}}$ (I)	F10	124	91	I/O/Z	PU	GPIO or Compare 3 Output Trip
GPIOB OR EVB SIGNALS							
GPIOB0	PWM7 (O)	N2	45	33	I/O/Z	PU	GPIO or PWM Output Pin #7
GPIOB1	PWM8 (O)	P2	46	34	I/O/Z	PU	GPIO or PWM Output Pin #8
GPIOB2	PWM9 (O)	N3	47	35	I/O/Z	PU	GPIO or PWM Output Pin #9
GPIOB3	PWM10 (O)	P3	48	36	I/O/Z	PU	GPIO or PWM Output Pin #10
GPIOB4	PWM11 (O)	L4	49	37	I/O/Z	PU	GPIO or PWM Output Pin #11
GPIOB5	PWM12 (O)	M4	50	38	I/O/Z	PU	GPIO or PWM Output Pin #12
GPIOB6	T3PWM_T3CMP (I)	K5	53	40	I/O/Z	PU	GPIO or Timer 3 Output
GPIOB7	T4PWM_T4CMP (I)	N5	55	41	I/O/Z	PU	GPIO or Timer 4 Output
GPIOB8	CAP4_QEP3 (I)	M5	57	43	I/O/Z	PU	GPIO or Capture Input #4
GPIOB9	CAP5_QEP4 (I)	M6	59	44	I/O/Z	PU	GPIO or Capture Input #5
GPIOB10	CAP6_QEPI2 (I)	P6	60	45	I/O/Z	PU	GPIO or Capture Input #6
GPIOB11	TDIRB (I)	L8	71	54	I/O/Z	PU	GPIO or Timer Direction
GPIOB12	TCLKINB (I)	K8	72	55	I/O/Z	PU	GPIO or Timer Clock Input
GPIOB13	$\overline{\text{C4TRIP}}$ (I)	N6	61	46	I/O/Z	PU	GPIO or Compare 4 Output Trip
GPIOB14	$\overline{\text{C5TRIP}}$ (I)	L6	62	47	I/O/Z	PU	GPIO or Compare 5 Output Trip
GPIOB15	$\overline{\text{C6TRIP}}$ (I)	K7	63	48	I/O/Z	PU	GPIO or Compare 6 Output Trip

† Typical drive strength of the output buffer for all pins [except TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins] is 4 mA typical.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2–2. Signal Descriptions† (Continued)

GPIO	PERIPHERAL SIGNAL	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
		179-PIN GHH	176-PIN PGF	128-PIN PBK			
GPIOD OR EVA SIGNALS							
GPIOD0	$\overline{\text{T1CTRIP\_PDPINTA}}$ (I)	H14	110	81	I/O/Z	PU	Timer 1 Compare Output Trip
GPIOD1	$\overline{\text{T2CTRIP}}$/$\overline{\text{EVASOC}}$ (I)	G10	115	84	I/O/Z	PU	Timer 2 Compare Output Trip or External ADC Start-of-Conversion EV-A
GPIOD OR EVB SIGNALS							
GPIOD5	$\overline{\text{T3CTRIP\_PDPINTB}}$ (I)	P10	79	60	I/O/Z	PU	Timer 3 Compare Output Trip
GPIOD6	$\overline{\text{T4CTRIP}}$/$\overline{\text{EVBSOC}}$ (I)	P11	83	61	I/O/Z	PU	Timer 4 Compare Output Trip or External ADC Start-of-Conversion EV-B
GPIOE OR INTERRUPT SIGNALS							
GPIOE0	XINT1_$\overline{\text{XBIO}}$ (I)	D9	149	106	I/O/Z	–	GPIO or XINT1 or $\overline{\text{XBIO}}$ input
GPIOE1	XINT2_ADCSOC (I)	D8	151	108	I/O/Z	–	GPIO or XINT2 or ADC start of conversion
GPIOE2	XNMI_XINT13 (I)	E8	150	107	I/O/Z	PU	GPIO or XNMI or XINT13
GPIOF OR SPI SIGNALS							
GPIOF0	SPISIMOA (O)	M1	40	31	I/O/Z	–	GPIO or SPI slave in, master out
GPIOF1	SPISOMIA (I)	N1	41	32	I/O/Z	–	GPIO or SPI slave out, master in
GPIOF2	SPICLKA (I/O)	K2	34	27	I/O/Z	–	GPIO or SPI clock
GPIOF3	SPISTEA (I/O)	K4	35	28	I/O/Z	–	GPIO or SPI slave transmit enable
GPIOF OR SCI-A SIGNALS							
GPIOF4	SCITXDA (O)	C7	155	111	I/O/Z	PU	GPIO or SCI asynchronous serial port TX data
GPIOF5	SCIRXDA (I)	A7	157	112	I/O/Z	PU	GPIO or SCI asynchronous serial port RX data
GPIOF OR CAN SIGNALS							
GPIOF6	CANTXA (O)	N12	87	64	I/O/Z	PU	GPIO or eCAN transmit data
GPIOF7	CANRXA (I)	N13	89	65	I/O/Z	PU	GPIO or eCAN receive data
GPIOF OR McBSP SIGNALS							
GPIOF8	MCLKXA (I/O)	J1	28	23	I/O/Z	PU	GPIO or transmit clock
GPIOF9	MCLKRA (I/O)	H2	25	21	I/O/Z	PU	GPIO or receive clock
GPIOF10	MFSXA (I/O)	H4	26	22	I/O/Z	PU	GPIO or transmit frame synch
GPIOF11	MFSRA (I/O)	J2	29	24	I/O/Z	PU	GPIO or receive frame synch
GPIOF12	MDXA (O)	G1	22	19	I/O/Z	–	GPIO or transmitted serial data
GPIOF13	MDRA (I)	G2	20	18	I/O/Z	PU	GPIO or received serial data

† Typical drive strength of the output buffer for all pins [except TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins] is 4 mA typical.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

Table 2−2. Signal Descriptions† (Continued)

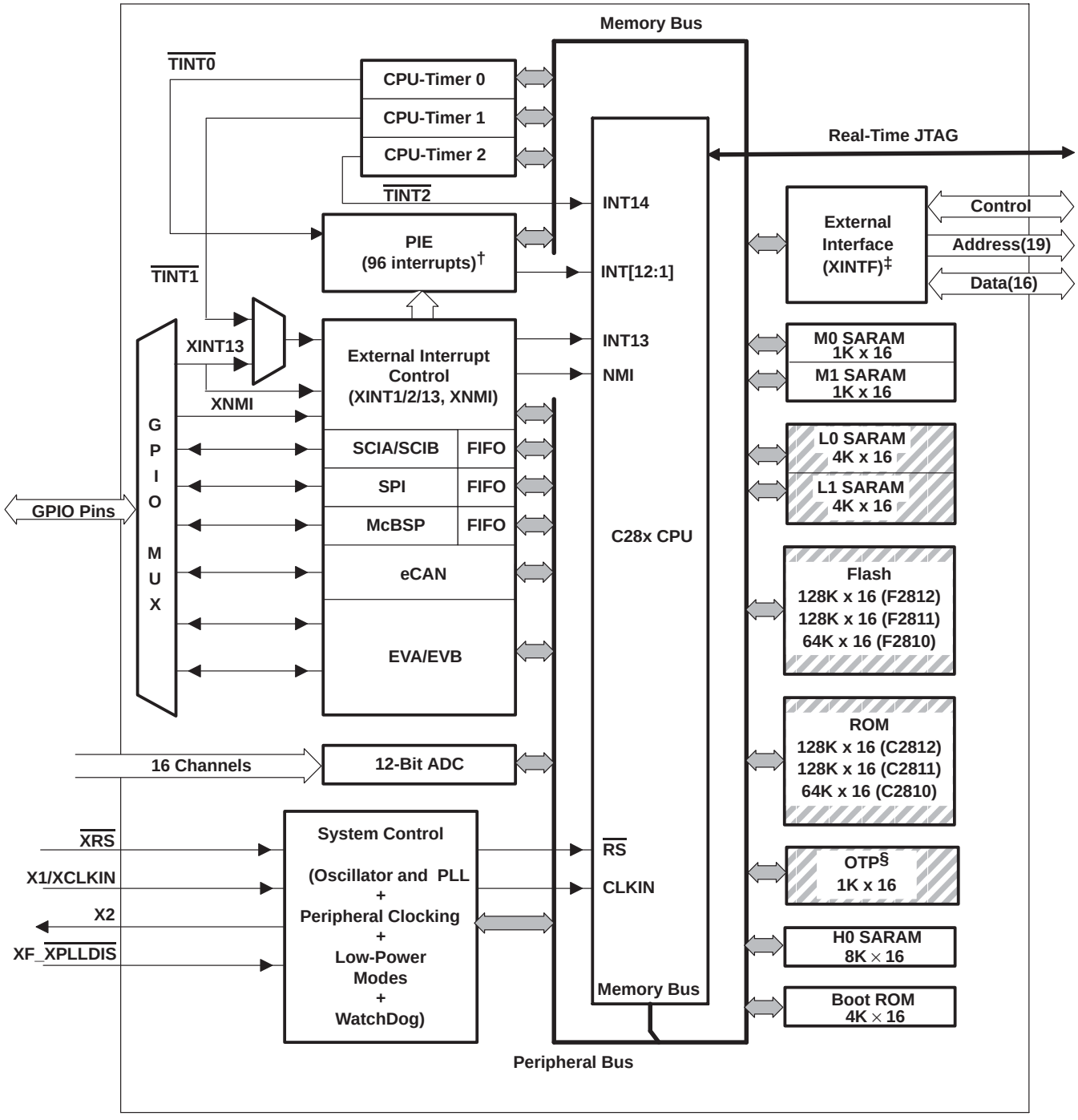
GPIO	PERIPHERAL SIGNAL	PIN NO.			I/O/Z‡	PU/PD§	DESCRIPTION
		179-PIN GHH	176-PIN PGF	128-PIN PBK			
GPIOF OR XF CPU OUTPUT SIGNAL							
GPIOF14	XF_$\overline{\text{XPLLDIS}}$ (O)	A11	140	101	I/O/Z	PU	This pin has three functions: 1. XF − General-purpose output pin. 2. XPLLDIS − This pin will be sampled during reset to check if the PLL needs to be disabled. The PLL will be disabled if this pin is sensed low. HALT and STANDBY modes cannot be used when the PLL is disabled. 3. GPIO − GPIO function
GPIOG OR SCI-B SIGNALS							
GPIOG4	SCITXDB (O)	P14	90	66	I/O/Z	−	GPIO or SCI asynchronous serial port transmit data
GPIOG5	SCIRXDB (I)	M13	91	67	I/O/Z	−	GPIO or SCI asynchronous serial port receive data

† Typical drive strength of the output buffer for all pins [except TDO, XCLKOUT, XF, XINTF, EMU0, and EMU1 pins] is 4 mA typical.
‡ I = Input, O = Output, Z = High impedance
§ PU = pin has internal pullup; PD = pin has internal pulldown. Pullup/pulldown strength is given in Section 6.3.

NOTE:

Other than the power supply pins, no pin should be driven before the 3.3-V rail has reached recommended operating conditions. However, it is acceptable for an I/O pin to ramp along with the 3.3-V supply.

3 Functional Overview

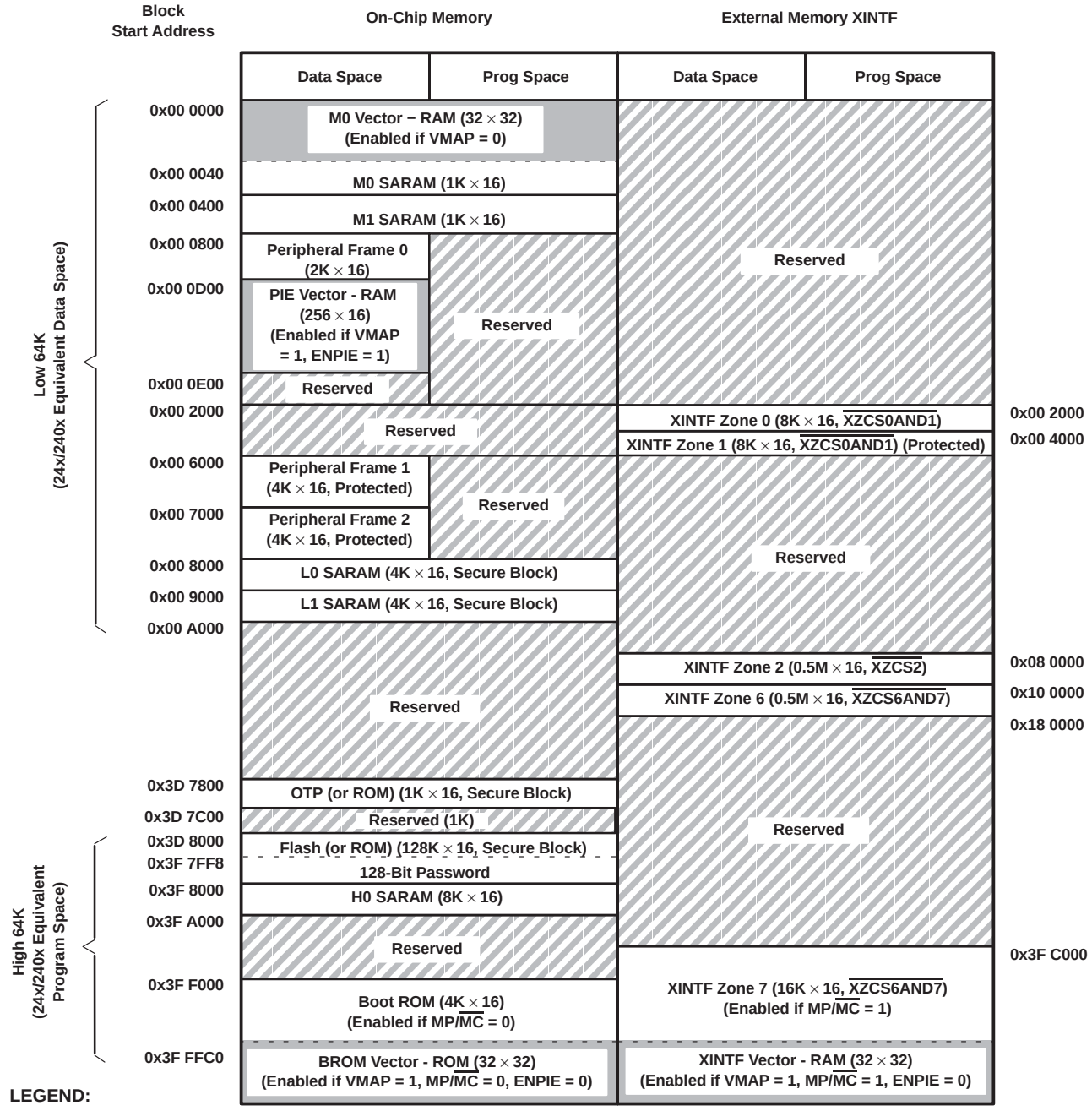


Protected by the code-security module.

† 45 of the possible 96 interrupts are used on the devices.
‡ XINTF is available on the F2812 and C2812 devices only.
§ On C281x devices, the OTP is replaced with a 1K X 16 block of ROM

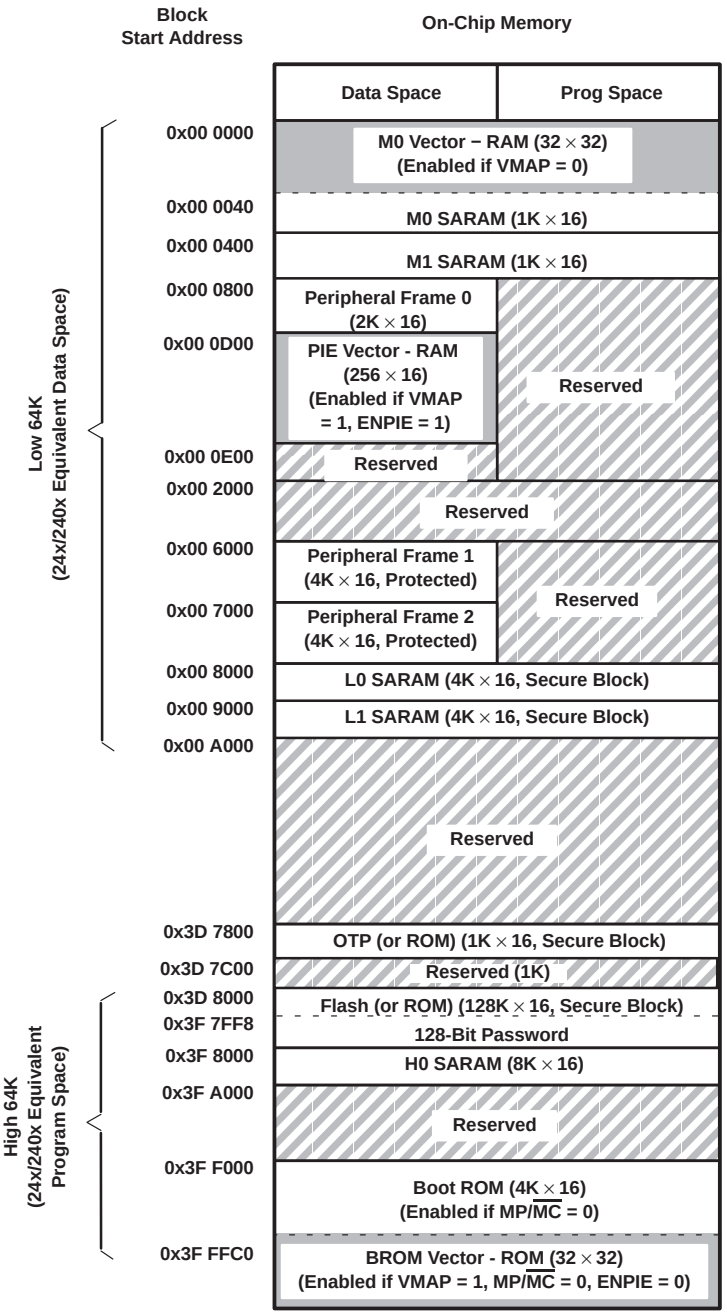
Figure 3−1. Functional Block Diagram

3.1 Memory Map



NOTES: A. Memory blocks are not to scale.
B. Reserved locations are reserved for future expansion. Application should not access these areas.
C. Boot ROM and Zone 7 memory maps are active either in on-chip or XINTF zone depending on MP/$\overline{MC}$, not in both.
D. Peripheral Frame 0, Peripheral Frame 1, and Peripheral Frame 2 memory maps are restricted to data memory only. User program cannot access these memory maps in program space.
E. "Protected" means the order of Write followed by Read operations is preserved rather than the pipeline order.
F. Certain memory ranges are EALLOW protected against spurious writes after configuration.
G. Zones 0 and 1 and Zones 6 and 7 share the same chip select; hence, these memory blocks have mirrored locations.

Figure 3−2. F2812/C2812 Memory Map (See Notes A through E)

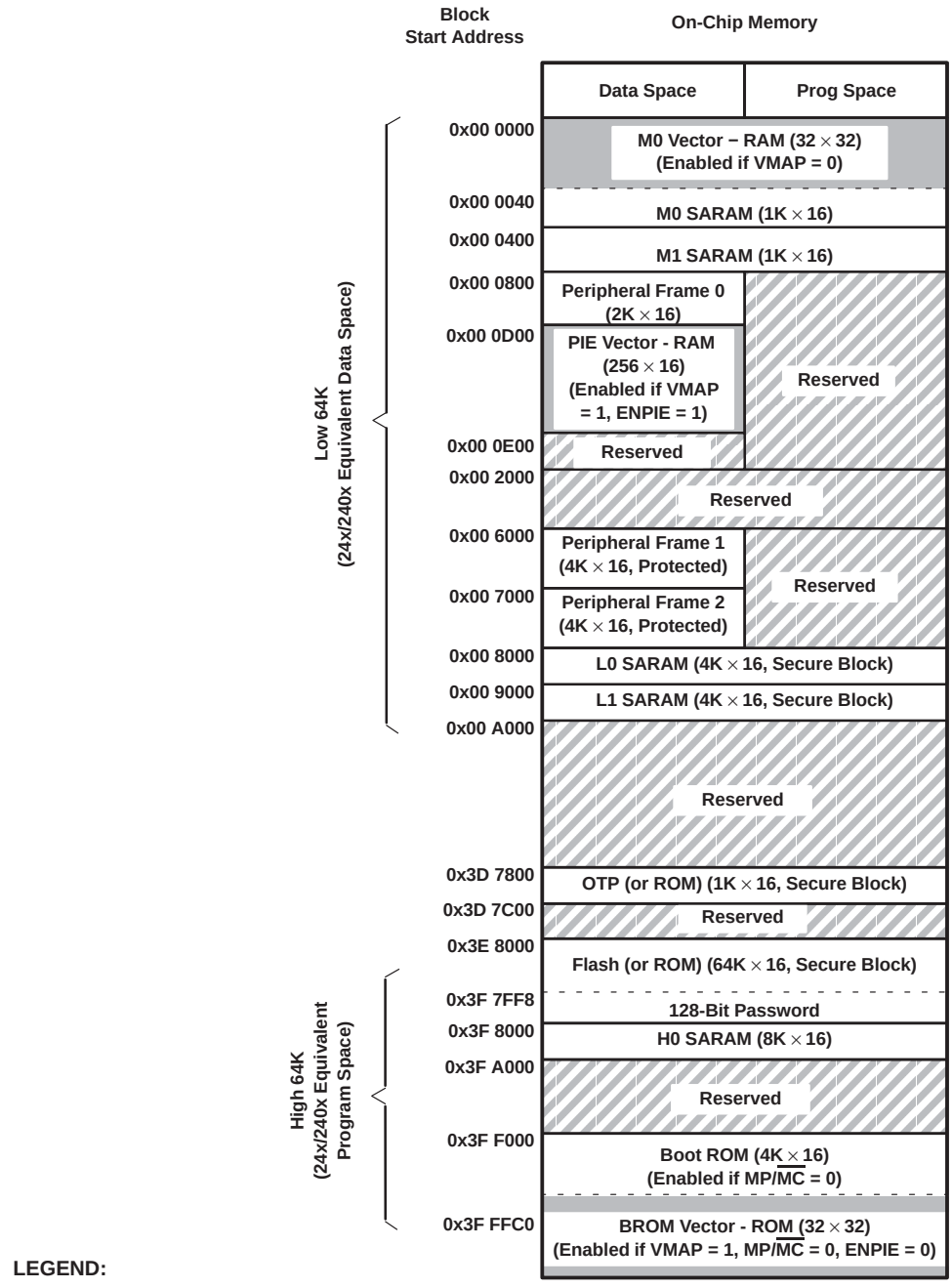


LEGEND:

Only one of these vector maps—M0 vector, PIE vector, BROM vector, XINTF vector—should be enabled at a time.

NOTES: A. Memory blocks are not to scale.
B. Reserved locations are reserved for future expansion. Application should not access these areas.
C. Peripheral Frame 0, Peripheral Frame 1, and Peripheral Frame 2 memory maps are restricted to data memory only. User program cannot access these memory maps in program space.
D. “Protected” means the order of Write followed by Read operations is preserved rather than the pipeline order.
E. Certain memory ranges are EALLOW protected against spurious writes after configuration.

Figure 3−3. F2811/C2811 Memory Map (See Notes A through E)



NOTES: A. Memory blocks are not to scale.
B. Reserved locations are reserved for future expansion. Application should not access these areas.
C. Peripheral Frame 0, Peripheral Frame 1, and Peripheral Frame 2 memory maps are restricted to data memory only. User program cannot access these memory maps in program space.
D. “Protected” means the order of Write followed by Read operations is preserved rather than the pipeline order.
E. Certain memory ranges are EALLOW protected against spurious writes after configuration.

Figure 3−4. F2810/C2810 Memory Map (See Notes A through E)

Table 3−1. Addresses of Flash Sectors in F2812 and F2811

ADDRESS RANGE	PROGRAM AND DATA SPACE
0x3D 8000 0x3D 9FFF	Sector J, 8K x 16
0x3D A000 0x3D BFFF	Sector I, 8K x 16
0x3D C000 0x3D FFFF	Sector H, 16K x 16
0x3E 0000 0x3E 3FFF	Sector G, 16K x 16
0x3E 4000 0x3E 7FFF	Sector F, 16K x 16
0x3E 8000 0x3E BFFF	Sector E, 16K x 16
0x3E C000 0x3E FFFF	Sector D, 16K x 16
0x3F 0000 0x3F 3FFF	Sector C, 16K x 16
0x3F 4000 0x3F 5FFF	Sector B, 8K x 16
0x3F 6000	Sector A, 8K x 16
0x3F 7F80 0x3F 7FF5	Program to 0x0000 when using the Code Security Module
0x3F 7FF6 0x3F 7FF7	Boot-to-Flash (or ROM) Entry Point (program branch instruction here)
0x3F 7FF8 0x3F 7FFF	Security Password (128-Bit) (Do not program to all zeros)

Table 3−2. Addresses of Flash Sectors in F2810

ADDRESS RANGE	PROGRAM AND DATA SPACE
0x3E 8000 0x3E BFFF	Sector E, 16K x 16
0x3E C000 0x3E FFFF	Sector D, 16K x 16
0x3F 0000 0x3F 3FFF	Sector C, 16K x 16
0x3F 4000 0x3F 5FFF	Sector B, 8K x 16
0x3F 6000	Sector A, 8K x 16
0x3F 7F80 0x3F 7FF5	Program to 0x0000 when using the Code Security Module
0x3F 7FF6 0x3F 7FF7	Boot-to-Flash (or ROM) Entry Point (program branch instruction here)
0x3F 7FF8 0x3F 7FFF	Security Password (128-Bit) (Do not program to all zeros)

The “Low 64K” of the memory address range maps into the data space of the 240x. The “High 64K” of the memory address range maps into the program space of the 24x/240x. 24x/240x-compatible code will only execute from the “High 64K” memory area. Hence, the top 32K of Flash/ROM and H0 SARAM block can be used to run 24x/240x-compatible code (if MP/$\overline{MC}$ mode is low) or, on the 2812, code can be executed from XINTF Zone 7 (if MP/$\overline{MC}$ mode is high).

The XINTF consists of five independent zones. One zone has its own chip select and the remaining four zones share two chip selects. Each zone can be programmed with its own timing (wait states) and to either sample or ignore external ready signal. This makes interfacing to external peripherals easy and glueless.

NOTE:

The chip selects of XINTF Zone 0 and Zone 1 are merged together into a single chip select ($\overline{XZCS0AND1}$); and the chip selects of XINTF Zone 6 and Zone 7 are merged together into a single chip select ($\overline{XZCS6AND7}$). See Section 3.5, “External Interface, XINTF (2812 only)”, for details.

Peripheral Frame 1, Peripheral Frame 2, and XINTF Zone 1 are grouped together so as to enable these blocks to be “write/read peripheral block protected”. The “protected” mode ensures that all accesses to these blocks happen as written. Because of the C28x pipeline, a write immediately followed by a read, to different memory locations, will appear in reverse order on the memory bus of the CPU. This can cause problems in certain peripheral applications where the user expected the write to occur first (as written). The C28x CPU supports a block protection mode where a region of memory can be protected so as to make sure that operations occur as written (the penalty is extra cycles are added to align the operations). This mode is programmable and by default, it will protect the selected zones.

On the 2812, at reset, XINTF Zone 7 is accessed if the XMP/$\overline{MC}$ pin is pulled high. This signal selects microprocessor or microcomputer mode of operation. In microprocessor mode, Zone 7 is mapped to high memory such that the vector table is fetched externally. The Boot ROM is disabled in this mode. In microcomputer mode, Zone 7 is disabled such that the vectors are fetched from Boot ROM. This allows the user to either boot from on-chip memory or from off-chip memory. The state of the XMP/$\overline{MC}$ signal on reset is stored in an MP/$\overline{MC}$ mode bit in the XINTCNF2 register. The user can change this mode in software and hence control the mapping of Boot ROM and XINTF Zone 7. No other memory blocks are affected by XMP/$\overline{MC}$.

I/O space is not supported on the 2812 XINTF.

The wait states for the various spaces in the memory map area are listed in Table 3−3.

Table 3−3. Wait States

AREA	WAIT-STATES	COMMENTS
M0 and M1 SARAMs	0-wait	Fixed
Peripheral Frame 0	0-wait	Fixed
Peripheral Frame 1	0-wait (writes) 2-wait (reads)	Fixed
Peripheral Frame 2	0-wait (writes) 2-wait (reads)	Fixed
L0 & L1 SARAMs	0-wait	Fixed
OTP (or ROM)	Programmable, 1-wait minimum	Programmed via the Flash registers. 1-wait-state operation is possible at a reduced CPU frequency. See Section 3.2.6, Flash (F281x Only), for more information.
Flash (or ROM)	Programmable, 0-wait minimum	Programmed via the Flash registers. 0-wait-state operation is possible at reduced CPU frequency. The CSM password locations are hardwired for 16 wait-states. See Section 3.2.6, Flash (F281x Only), for more information.
H0 SARAM	0-wait	Fixed
Boot-ROM	1-wait	Fixed
XINTF	Programmable, 1-wait minimum	Programmed via the XINTF registers. Cycles can be extended by external memory or peripheral. 0-wait operation is not possible.

3.2 Brief Descriptions

3.2.1 C28x CPU

The C28x™ DSP generation is the newest member of the TMS320C2000™ DSP platform. The C28x is source code compatible to the 24x/240x DSP devices, hence existing 240x users can leverage their significant software investment. Additionally, the C28x is a very efficient C/C++ engine, hence enabling users to develop not only their system control software in a high-level language, but also enables math algorithms to be developed using C/C++. The C28x is as efficient in DSP math tasks as it is in system control tasks that typically are handled by microcontroller devices. This efficiency removes the need for a second processor in many systems. The 32 x 32-bit MAC capabilities of the C28x and its 64-bit processing capabilities, enable the C28x to efficiently handle higher numerical resolution problems that would otherwise demand a more expensive floating-point processor solution. Add to this the fast interrupt response with automatic context save of critical registers, resulting in a device that is capable of servicing many asynchronous events with minimal latency. The C28x has an 8-level-deep protected pipeline with pipelined memory accesses. This pipelining enables the C28x to execute at high speeds without resorting to expensive high-speed memories. Special branch-look-ahead hardware minimizes the latency for conditional discontinuities. Special store conditional operations further improve performance.

C28x and TMS320C2000 are trademarks of Texas Instruments.

3.2.2 Memory Bus (Harvard Bus Architecture)

As with many DSP type devices, multiple busses are used to move data between the memories and peripherals and the CPU. The C28x memory bus architecture contains a program read bus, data read bus and data write bus. The program read bus consists of 22 address lines and 32 data lines. The data read and write busses consist of 32 address lines and 32 data lines each. The 32-bit-wide data busses enable single cycle 32-bit operations. The multiple bus architecture, commonly termed “Harvard Bus”, enables the C28x to fetch an instruction, read a data value and write a data value in a single cycle. All peripherals and memories attached to the memory bus will prioritize memory accesses. Generally, the priority of Memory Bus accesses can be summarized as follows:

Highest:	Data Writes†
	Program Writes†
	Data Reads
	Program Reads‡
Lowest:	Fetches‡

3.2.3 Peripheral Bus

To enable migration of peripherals between various Texas Instruments (TI) DSP family of devices, the F281x and C281x adopt a peripheral bus standard for peripheral interconnect. The peripheral bus bridge multiplexes the various busses that make up the processor “Memory Bus” into a single bus consisting of 16 address lines and 16 or 32 data lines and associated control signals. Two versions of the peripheral bus are supported on the F281x and C281x. One version only supports 16-bit accesses (called peripheral frame 2) and this retains compatibility with C240x-compatible peripherals. The other version supports both 16- and 32-bit accesses (called peripheral frame 1).

3.2.4 Real-Time JTAG and Analysis

The F281x and C281x implement the standard IEEE 1149.1 JTAG interface. Additionally, the F281x and C281x support real-time mode of operation whereby the contents of memory, peripheral and register locations can be modified while the processor is running and executing code and servicing interrupts. The user can also single step through non-time critical code while enabling time-critical interrupts to be serviced without interference. The F281x and C281x implement the real-time mode in hardware within the CPU. This is a unique feature to the F281x and C281x, no software monitor is required. Additionally, special analysis hardware is provided which allows the user to set hardware breakpoint or data/address watch-points and generate various user selectable break events when a match occurs.

3.2.5 External Interface (XINTF) (2812 Only)

This asynchronous interface consists of 19 address lines, 16 data lines, and three chip-select lines. The chip-select lines are mapped to five external zones, Zones 0, 1, 2, 6, and 7. Zones 0 and 1 share a single chip-select; Zones 6 and 7 also share a single chip-select. Each of the five zones can be programmed with a different number of wait states, strobe signal setup and hold timing and each zone can be programmed for extending wait states externally or not. The programmable wait-state, chip-select and programmable strobe timing enables glueless interface to external memories and peripherals.

† Simultaneous Data and Program writes cannot occur on the Memory Bus.
‡ Simultaneous Program Reads and Fetches cannot occur on the Memory Bus.

3.2.6 Flash (F281x Only)

The F2812 and F2811 contain 128K x 16 of embedded flash memory, segregated into four 8K X 16 sectors, and six 16K X 16 sectors. The F2810 has 64K X 16 of embedded flash, segregated into two 8K X 16 sectors, and three 16K X 16 sectors. All three devices also contain a single 1K x 16 of OTP memory at address range 0x3D 7800 − 0x3D 7BFF. The user can individually erase, program, and validate a flash sector while leaving other sectors untouched. However, it is not possible to use one sector of the flash or the OTP to execute flash algorithms that erase/program other sectors. Special memory pipelining is provided to enable the flash module to achieve higher performance. The flash/OTP is mapped to both program and data space; therefore, it can be used to execute code or store data information.

NOTE:

The F2810/F2811/F2812 Flash and OTP wait states can be configured by the application. This allows applications running at slower frequencies to configure the flash to use fewer wait states.

Flash effective performance can be improved by enabling the flash pipeline mode in the Flash options register. With this mode enabled, effective performance of linear code execution will be much faster than the raw performance indicated by the wait state configuration alone. The exact performance gain when using the Flash pipeline mode is application-dependent. The pipeline mode is not available for the OTP block.

For more information on the Flash options, Flash wait-state, and OTP wait-state registers, see the *TMS320x281x System Control and Interrupts Reference Guide* (literature number SPRU078).

3.2.7 ROM (C281x Only)

The C2812 and C2811 contain 128K x 16 of ROM. The C2810 has 64K x 16 of ROM. In addition to this, there is a 1K X 16 ROM block that replaces the OTP memory available in flash devices. For information on how to submit ROM codes to TI, see the *TMS320C28x CPU and Instruction Set Reference Guide* (literature number SPRU430).

3.2.8 M0, M1 SARAMs

All C28x devices contain these two blocks of single access memory, each 1K x 16 in size. The stack pointer points to the beginning of block M1 on reset. The M0 block overlaps the 240x device B0, B1, B2 RAM blocks and hence the mapping of data variables on the 240x devices can remain at the same physical address on C28x devices. The M0 and M1 blocks, like all other memory blocks on C28x devices, are mapped to both program and data space. Hence, the user can use M0 and M1 to execute code or for data variables. The partitioning is performed within the linker. The C28x device presents a unified memory map to the programmer. This makes for easier programming in high-level languages.

3.2.9 L0, L1, H0 SARAMs

The F281x and C281x contain an additional 16K x 16 of single-access RAM, divided into 3 blocks (4K + 4K + 8K). Each block can be independently accessed hence minimizing pipeline stalls. Each block is mapped to both program and data space.

3.2.10 Boot ROM

The Boot ROM is factory-programmed with boot-loading software. The Boot ROM program executes after device reset and checks several GPIO pins to determine which boot mode to enter. For example, the user can select to execute code already present in the internal Flash or download new software to internal RAM through one of several serial ports. Other boot modes exist as well. The Boot ROM also contains standard tables, such as SIN/COS waveforms, for use in math-related algorithms. Table 3−4 shows the details of how various boot modes may be invoked. See the *TMS320x281x DSP Boot ROM Reference Guide* (literature number SPRU095), for more information.

Table 3−4. Boot Mode Selection

	GPIOF4	GPIOF12	GPIOF3	GPIOF2
	(SCITXDA)	(MDXA)	(SPISTEA)	(SPICLK)
Mode Selected†	PU	No PU	No PU	No PU
Jump to Flash/ROM address 0x3F 7FF6 A branch instruction must have been programmed here prior to reset to re−direct code excution as desired.	1	x	x	x
Call SPI_Boot to load from an external serial SPI EEPROM	0	1	x	x
Call SCI_Boot to load from SCI-A	0	0	1	1
Jump to H0 SARAM address 0x3F 8000	0	0	1	0
Jump to OTP address 0x3D 7800	0	0	0	1
Call Parallel_Boot to load from GPIO Port B	0	0	0	0

† PU = pin has an internal pullup No PU = pin does not have an internal pullup
‡ Extra care must be taken due to any affect toggling SPICLK to select a boot mode may have on external logic.
§ If the boot mode selected is Flash, H0, or OTP, then no external code is loaded by the bootloader.

3.2.11 Security

The F281x and C281x support high levels of security to protect the user firmware from being reverse-engineered. The security features a 128-bit password (hardcoded for 16 wait states), which the user programs into the flash. One code security module (CSM) is used to protect the flash/ROM/OTP and the L0/L1 SARAM blocks. The security feature prevents unauthorized users from examining the memory contents via the JTAG port, executing code from external memory or trying to boot-load some undesirable software that would export the secure memory contents. To enable access to the secure blocks, the user must write the correct 128-bit ”KEY” value, which matches the value stored in the password locations within the Flash/ROM.

NOTE:

For code security operation, all addresses between 0x3F7F80 and 0x3F7FF5 cannot be used as program code or data, but must be programmed to 0x0000 when the Code Security Passwords are programmed. If security is not a concern, then these addresses may be used for code or data.

The 128-bit password (at 0x3F 7FF8 − 0x3F 7FFF) must *not* be programmed to zeros. Doing so would permanently lock the device.

Code Security Module Disclaimer

The Code Security Module (“CSM”) included on this device was designed to password protect the data stored in the associated memory (either ROM or Flash) and is warranted by Texas Instruments (TI), in accordance with its standard terms and conditions, to conform to TI’s published specifications for the warranty period applicable for this device.

TI DOES NOT, HOWEVER, WARRANT OR REPRESENT THAT THE CSM CANNOT BE COMPROMISED OR BREACHED OR THAT THE DATA STORED IN THE ASSOCIATED MEMORY CANNOT BE ACCESSED THROUGH OTHER MEANS. MOREOVER, EXCEPT AS SET FORTH ABOVE, TI MAKES NO WARRANTIES OR REPRESENTATIONS CONCERNING THE CSM OR OPERATION OF THIS DEVICE, INCLUDING ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

IN NO EVENT SHALL TI BE LIABLE FOR ANY CONSEQUENTIAL, SPECIAL, INDIRECT, INCIDENTAL, OR PUNITIVE DAMAGES, HOWEVER CAUSED, ARISING IN ANY WAY OUT OF YOUR USE OF THE CSM OR THIS DEVICE, WHETHER OR NOT TI HAS BEEN ADVISED OF THE POSSIBILITY OF SUCH DAMAGES. EXCLUDED DAMAGES INCLUDE, BUT ARE NOT LIMITED TO LOSS OF DATA, LOSS OF GOODWILL, LOSS OF USE OR INTERRUPTION OF BUSINESS OR OTHER ECONOMIC LOSS.

3.2.12 Peripheral Interrupt Expansion (PIE) Block

The PIE block serves to multiplex numerous interrupt sources into a smaller set of interrupt inputs. The PIE block can support up to 96 peripheral interrupts. On the F281x and C281x, 45 of the possible 96 interrupts are used by peripherals. The 96 interrupts are grouped into blocks of 8 and each group is fed into 1 of 12 CPU interrupt lines (INT1 to INT12). Each of the 96 interrupts is, supported by its own vector stored in a dedicated RAM block that can be overwritten by the user. The vector is, automatically fetched by the CPU on servicing the interrupt. It takes 8 CPU clock cycles to fetch the vector and save critical CPU registers. Hence the CPU can quickly respond to interrupt events. Prioritization of interrupts is controlled in hardware and software. Each individual interrupt can be enabled/disabled within the PIE block.

3.2.13 External Interrupts (XINT1, XINT2, XINT13, XNMI)

The F281x and C281x support three masked external interrupts (XINT1, 2, 13). XINT13 is combined with one non-masked external interrupt (XNMI). The combined signal name is XNMI_XINT13. Each of the interrupts can be selected for negative or positive edge triggering and can also be enabled/disabled (including the XNMI). The masked interrupts also contain a 16-bit free running up counter, which is reset to zero when a valid interrupt edge is detected. This counter can be used to accurately time stamp the interrupt.

3.2.14 Oscillator and PLL

The F281x and C281x can be clocked by an external oscillator or by a crystal attached to the on-chip oscillator circuit. A PLL is provided supporting up to 10-input clock-scaling ratios. The PLL ratios can be changed on-the-fly in software, enabling the user to scale back on operating frequency if lower power operation is desired. Refer to the Electrical Specification section for timing details. The PLL block can be set in bypass mode.

3.2.15 Watchdog

The F281x and C281x support a watchdog timer. The user software must regularly reset the watchdog counter within a certain time frame; otherwise, the watchdog will generate a reset to the processor. The watchdog can be disabled if necessary.

3.2.16 Peripheral Clocking

The clocks to each individual peripheral can be enabled/disabled so as to reduce power consumption when a peripheral is not in use. Additionally, the system clock to the serial ports (except eCAN) and the event managers, CAP and QEP blocks can be scaled relative to the CPU clock. This enables the timing of peripherals to be decoupled from increasing CPU clock speeds.

3.2.17 Low-Power Modes

The F281x and C281x devices are full static CMOS devices. Three low-power modes are provided:

IDLE:	Place CPU into low-power mode. Peripheral clocks may be turned off selectively and only those peripherals that need to function during IDLE are left operating. An enabled interrupt from an active peripheral will wake the processor from IDLE mode.
STANDBY:	Turn off clock to CPU and peripherals. This mode leaves the oscillator and PLL functional. An external interrupt event will wake the processor and the peripherals. Execution begins on the next valid cycle after detection of the interrupt event.
HALT:	Turn off oscillator. This mode basically shuts down the device and places it in the lowest possible power consumption mode. Only a reset or XNMI will wake the device from this mode.

3.2.18 Peripheral Frames 0, 1, 2 (PFn)

The F281x and C281x segregate peripherals into three sections. The mapping of peripherals is as follows:

PF0:	XINTF:	External Interface Configuration Registers (2812 only)
	PIE:	PIE Interrupt Enable and Control Registers Plus PIE Vector Table
	Flash:	Flash Control, Programming, Erase, Verify Registers
	Timers:	CPU-Timers 0, 1, 2 Registers
	CSM:	Code Security Module KEY Registers
PF1:	eCAN:	eCAN Mailbox and Control Registers
PF2:	SYS:	System Control Registers
	GPIO:	GPIO Mux Configuration and Control Registers
	EV:	Event Manager (EVA/EVB) Control Registers
	McBSP:	McBSP Control and TX/RX Registers
	SCI:	Serial Communications Interface (SCI) Control and RX/TX Registers
	SPI:	Serial Peripheral Interface (SPI) Control and RX/TX Registers
	ADC:	12-Bit ADC Registers

3.2.19 General-Purpose Input/Output (GPIO) Multiplexer

Most of the peripheral signals are multiplexed with general-purpose I/O (GPIO) signals. This enables the user to use a pin as GPIO if the peripheral signal or function is not used. On reset, all GPIO pins are configured as inputs. The user can then individually program each pin for GPIO mode or Peripheral Signal mode. For specific inputs, the user can also select the number of input qualification cycles. This is to filter unwanted noise glitches.

3.2.20 32-Bit CPU-Timers (0, 1, 2)

CPU-Timers 0, 1, and 2 are identical 32-bit timers with presettable periods and with 16-bit clock prescaling. The timers have a 32-bit count down register, which generates an interrupt when the counter reaches zero. The counter is decremented at the CPU clock speed divided by the prescale value setting. When the counter reaches zero, it is automatically reloaded with a 32-bit period value. CPU-Timer 2 is reserved for Real-Time OS (RTOS)/BIOS applications. CPU-Timer 1 is also reserved for TI system functions. CPU-Timer 2 is connected to INT14 of the CPU. CPU-Timer 1 can be connected to INT13 of the CPU. CPU-Timer 0 is for general use and is connected to the PIE block.

3.2.21 Control Peripherals

The F281x and C281x support the following peripherals which are used for embedded control and communication:

EV: The event manager module includes general-purpose timers, full-compare/PWM units, capture inputs (CAP) and quadrature-encoder pulse (QEP) circuits. Two such event managers are provided which enable two three-phase motors to be driven or four two-phase motors. The event managers on the F281x and C281x are compatible to the event managers on the 240x devices (with some minor enhancements).

ADC: The ADC block is a 12-bit converter, single ended, 16-channels. It contains two sample-and-hold units for simultaneous sampling.

3.2.22 Serial Port Peripherals

The F281x and C281x support the following serial communication peripherals:

eCAN: This is the enhanced version of the CAN peripheral. It supports 32 mailboxes, time stamping of messages, and is CAN 2.0B-compliant.

McBSP: This is the multichannel buffered serial port that is used to connect to E1/T1 lines, phone-quality codecs for modem applications or high-quality stereo-quality Audio DAC devices. The McBSP receive and transmit registers are supported by a 16-level FIFO. This significantly reduces the overhead for servicing this peripheral.

SPI: The SPI is a high-speed, synchronous serial I/O port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmable bit-transfer rate. Normally, the SPI is used for communications between the DSP controller and external peripherals or another processor. Typical applications include external I/O or peripheral expansion through devices such as shift registers, display drivers, and ADCs. Multi-device communications are supported by the master/slave operation of the SPI. On the F281x and C281x, the port supports a 16-level, receive and transmit FIFO for reducing servicing overhead.

SCI: The serial communications interface is a two-wire asynchronous serial port, commonly known as UART. On the F281x and C281x, the port supports a 16-level, receive and transmit FIFO for reducing servicing overhead.

3.3 Register Map

The F281x and C281x devices contain three peripheral register spaces. The spaces are categorized as follows:

- Peripheral Frame 0: These are peripherals that are mapped directly to the CPU memory bus. See Table 3−5.
- Peripheral Frame 1: These are peripherals that are mapped to the 32-bit peripheral bus. See Table 3−6.
- Peripheral Frame 2: These are peripherals that are mapped to the 16-bit peripheral bus. See Table 3−7.

Table 3−5. Peripheral Frame 0 Registers†

NAME	ADDRESS RANGE	SIZE (x16)	ACCESS TYPE‡
Device Emulation Registers	0x00 0880 0x00 09FF	384	EALLOW protected
reserved	0x00 0A00 0x00 0A7F	128	
FLASH Registers§	0x00 0A80 0x00 0ADF	96	EALLOW protected CSM Protected
Code Security Module Registers	0x00 0AE0 0x00 0AEF	16	EALLOW protected
reserved	0x00 0AF0 0x00 0B1F	48	
XINTF Registers	0x00 0B20 0x00 0B3F	32	Not EALLOW protected
reserved	0x00 0B40 0x00 0BFF	192	
CPU-TIMER0/1/2 Registers	0x00 0C00 0x00 0C3F	64	Not EALLOW protected
reserved	0x00 0C40 0x00 0CDF	160	
PIE Registers	0x00 0CE0 0x00 0CFF	32	Not EALLOW protected
PIE Vector Table	0x00 0D00 0x00 0DFF	256	EALLOW protected
Reserved	0x00 0E00 0x00 0FFF	512	

† Registers in Frame 0 support 16-bit and 32-bit accesses.
‡ If registers are EALLOW protected, then writes cannot be performed until the user executes the EALLOW instruction. The EDIS instruction disables writes. This prevents stray code or pointers from corrupting register contents.
§ The Flash Registers are also protected by the Code Security Module (CSM).

Table 3−6. Peripheral Frame 1 Registers¶

NAME	ADDRESS RANGE	SIZE (x16)	ACCESS TYPE
eCAN Registers	0x00 6000 0x00 60FF	256 (128 x 32)	Some eCAN control registers (and selected bits in other eCAN control registers) are EALLOW-protected.
eCAN Mailbox RAM	0x00 6100 0x00 61FF	256 (128 x 32)	Not EALLOW-protected
reserved	0x00 6200 0x00 6FFF	3584	

¶ The eCAN control registers only support 32-bit read/write operations. All 32-bit accesses are aligned to even address boundaries.

Table 3−7. Peripheral Frame 2 Registers†

NAME	ADDRESS RANGE	SIZE (x16)	ACCESS TYPE
reserved	0x00 7000 0x00 700F	16	
System Control Registers	0x00 7010 0x00 702F	32	EALLOW Protected
reserved	0x00 7030 0x00 703F	16	
SPI-A Registers	0x00 7040 0x00 704F	16	Not EALLOW Protected
SCI-A Registers	0x00 7050 0x00 705F	16	Not EALLOW Protected
reserved	0x00 7060 0x00 706F	16	
External Interrupt Registers	0x00 7070 0x00 707F	16	Not EALLOW Protected
reserved	0x00 7080 0x00 70BF	64	
GPIO Mux Registers	0x00 70C0 0x00 70DF	32	EALLOW Protected
GPIO Data Registers	0x00 70E0 0x00 70FF	32	Not EALLOW Protected
ADC Registers	0x00 7100 0x00 711F	32	Not EALLOW Protected
reserved	0x00 7120 0x00 73FF	736	
EV-A Registers	0x00 7400 0x00 743F	64	Not EALLOW Protected
reserved	0x00 7440 0x00 74FF	192	
EV-B Registers	0x00 7500 0x00 753F	64	Not EALLOW Protected
reserved	0x00 7540 0x00 774F	528	
SCI-B Registers	0x00 7750 0x00 775F	16	Not EALLOW Protected
reserved	0x00 7760 0x00 77FF	160	
McBSP Registers	0x00 7800 0x00 783F	64	Not EALLOW Protected
reserved	0x00 7840 0x00 7FFF	1984	

† Peripheral Frame 2 only allows 16-bit accesses. All 32-bit accesses are ignored (invalid data may be returned or written).

3.4 Device Emulation Registers

These registers are used to control the protection mode of the C28x CPU and to monitor some critical device signals. The registers are defined in Table 3−8.

Table 3−8. Device Emulation Registers

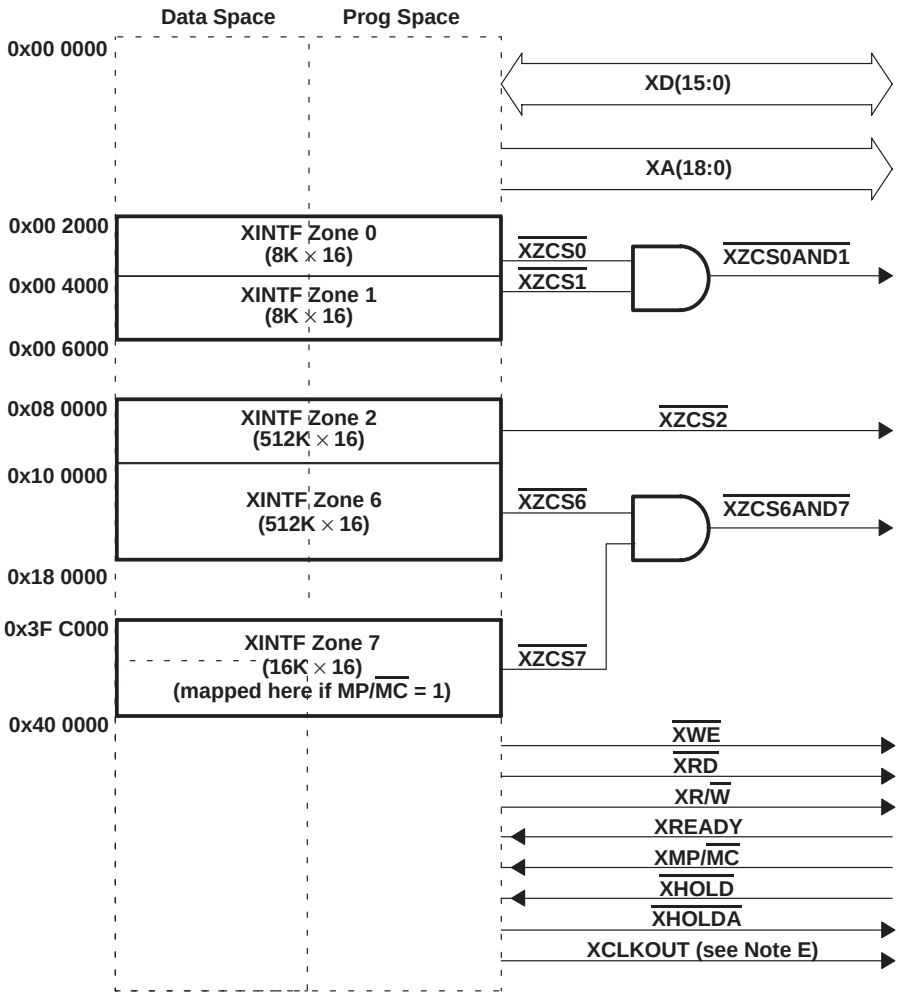
NAME	ADDRESS RANGE	SIZE (x16)	DESCRIPTION
DEVICECNF	0x00 0880 0x00 0881	2	Device Configuration Register
reserved	0x00 0882	1	Not supported on Revision C and later silicon
DEVICEID	0x00 0883	1	Device ID Register (0x0003 − Silicon − Rev. C and D) Device ID Register (0x0004 − Reserved) Device ID Register (0x0005 − Silicon − Rev. E)
PROTSTART	0x00 0884	1	Block Protection Start Address Register
PROTRANGE	0x00 0885	1	Block Protection Range Address Register
reserved	0x00 0886 0x00 09FF	378	

3.5 External Interface, XINTF (2812 Only)

This section gives a top-level view of the external interface (XINTF) that is implemented on the 2812 devices.

The external interface is a non-multiplexed asynchronous bus, similar to the C240x external interface. The external interface on the 2812 is mapped into five fixed zones shown in Figure 3−5.

Figure 3−5 shows the 2812 XINTF signals.



NOTES: A. The mapping of XINTF Zone 7 is dependent on the XMP/$\overline{MC}$ device input signal and the MP/$\overline{MC}$ mode bit (bit 8 of XINTCNF2 register). Zones 0, 1, 2, and 6 are always enabled.

B. Each zone can be programmed with different wait states, setup and hold timing, and is supported by zone chip selects ($\overline{XZCS0AND1}$, $\overline{XZCS2}$, $\overline{XZCS6AND7}$), which toggle when an access to a particular zone is performed. These features enable glueless connection to many external memories and peripherals.

C. The chip selects for Zone 0 and 1 are ANDed internally together to form one chip select ($\overline{XZCS0AND1}$). Any external memory that is connected to $\overline{XZCS0AND1}$ is dually mapped to both Zones 0 and Zone 1.

D. The chip selects for Zone 6 and 7 are ANDed internally together to form one chip select ($\overline{XZCS6AND7}$). Any external memory that is connected to $\overline{XZCS6AND7}$ is dually mapped to both Zones 6 and Zone 7. This means that if Zone 7 is disabled (via the MP/$\overline{MC}$ mode) then any external memory is still accessible via Zone 6 address space.

E. XCLKOUT is also pinned out on the 2810 and 2811.

Figure 3−5. External Interface Block Diagram

The operation and timing of the external interface, can be controlled by the registers listed in Table 3−9.

Table 3−9. XINTF Configuration and Control Register Mappings

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
XTIMING0	0x00 0B20	2	XINTF Timing Register, Zone 0 can access as two 16-bit registers or one 32-bit register
XTIMING1	0x00 0B22	2	XINTF Timing Register, Zone 1 can access as two 16-bit registers or one 32-bit register
XTIMING2	0x00 0B24	2	XINTF Timing Register, Zone 2 can access as two 16-bit registers or one 32-bit register
XTIMING6	0x00 0B2C	2	XINTF Timing Register, Zone 6 can access as two 16-bit registers or one 32-bit register
XTIMING7	0x00 0B2E	2	XINTF Timing Register, Zone 7 can access as two 16-bit registers or one 32-bit register
XINTCNF2	0x00 0B34	2	XINTF Configuration Register can access as two 16-bit registers or one 32-bit register
XBANK	0x00 0B38	1	XINTF Bank Control Register
XREVISION	0x00 0B3A	1	XINTF Revision Register

3.5.1 Timing Registers

XINTF signal timing can be tuned to match specific external device requirements such as setup and hold times to strobe signals for contention avoidance and maximizing bus efficiency. The timing parameters can be configured individually for each zone. This allows the programmer to maximize the efficiency of the bus, based on the type of memory or peripheral that the user needs to access. All XINTF timing values are with respect to XTIMCLK, which is equal to or one-half of the SYSCLKOUT rate, as shown in Figure 6−29.

For detailed information on the XINTF timing and configuration register bit fields, see the *TMS320x281x DSP External Interface (XINTF) Reference Guide* (literature number SPRU067).

3.5.2 XREVISION Register

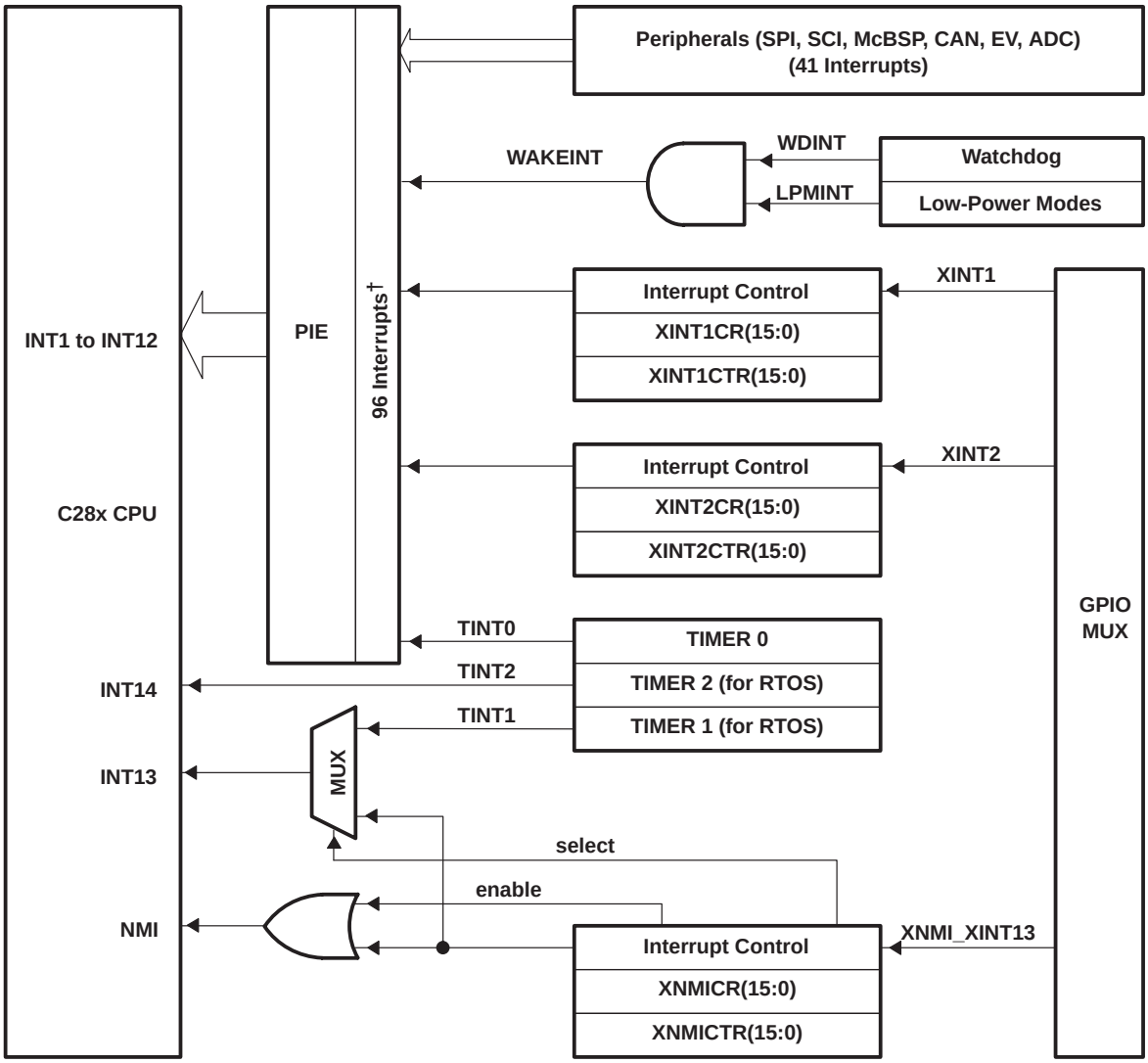
The XREVISION register contains a unique number to identify the particular version of XINTF used in the product. For the 2812, this register will be configured as described in Table 3−10.

Table 3−10. XREVISION Register Bit Definitions

BIT(S)	NAME	TYPE	RESET	DESCRIPTION
15−0	REVISION	R	0x0004	Current XINTF Revision. For internal use/reference. Test purposes only. Subject to change.

3.6 Interrupts

Figure 3−6 shows how the various interrupt sources are multiplexed within the F281x and C281x devices.



† Out of a possible 96 interrupts, 45 are currently used by peripherals.

Figure 3−6. Interrupt Sources

Eight PIE block interrupts are grouped into one CPU interrupt. In total, 12 CPU interrupt groups, with 8 interrupts per group equals 96 possible interrupts. On the F281x and C281x, 45 of these are used by peripherals as shown in Table 3−11.

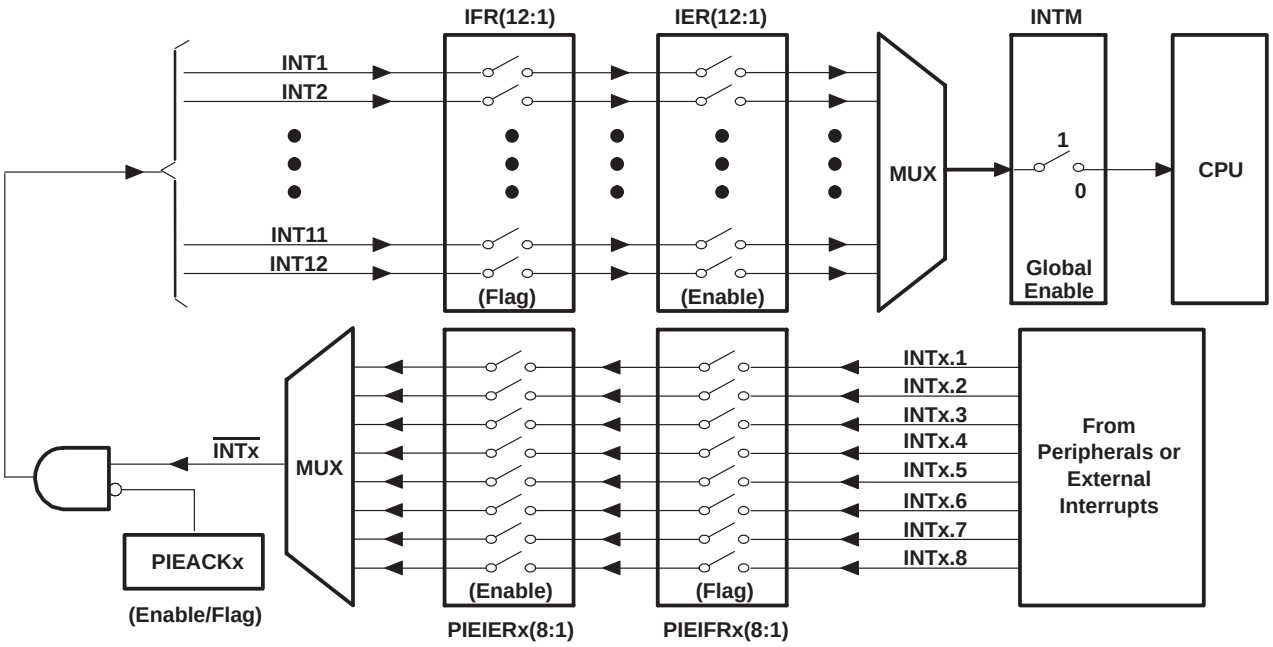


Figure 3−7. Multiplexing of Interrupts Using the PIE Block

Table 3−11. PIE Peripheral Interrupts†

CPU INTERRUPTS	PIE INTERRUPTS							
	INTx.8	INTx.7	INTx.6	INTx.5	INTx.4	INTx.3	INTx.2	INTx.1
INT1	$\overline{\text{WAKEINT}}$ (LPM/WD)	TINT0 (TIMER 0)	ADCINT (ADC)	XINT2	XINT1	reserved	PDPINTB (EV-B)	PDPINTA (EV-A)
INT2	reserved	T1OFINT (EV-A)	T1UFINT (EV-A)	T1CINT (EV-A)	T1PINT (EV-A)	CMP3INT (EV-A)	CMP2INT (EV-A)	CMP1INT (EV-A)
INT3	reserved	CAPINT3 (EV-A)	CAPINT2 (EV-A)	CAPINT1 (EV-A)	T2OFINT (EV-A)	T2UFINT (EV-A)	T2CINT (EV-A)	T2PINT (EV-A)
INT4	reserved	T3OFINT (EV-B)	T3UFINT (EV-B)	T3CINT (EV-B)	T3PINT (EV-B)	CMP6INT (EV-B)	CMP5INT (EV-B)	CMP4INT (EV-B)
INT5	reserved	CAPINT6 (EV-B)	CAPINT5 (EV-B)	CAPINT4 (EV-B)	T4OFINT (EV-B)	T4UFINT (EV-B)	T4CINT (EV-B)	T4PINT (EV-B)
INT6	reserved	reserved	MXINT (McBSP)	MRINT (McBSP)	reserved	reserved	SPITXINTA (SPI)	SPIRXINTA (SPI)
INT7	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT8	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT9	reserved	reserved	ECAN1INT (CAN)	ECAN0INT (CAN)	SCITXINTB (SCI-B)	SCIRXINTB (SCI-B)	SCITXINTA (SCI-A)	SCIRXINTA (SCI-A)
INT10	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT11	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved
INT12	reserved	reserved	reserved	reserved	reserved	reserved	reserved	reserved

† Out of the 96 possible interrupts, 45 interrupts are currently used. The remaining interrupts are reserved for future devices. These interrupts can be used as software interrupts if they are enabled at the PIEIFRx level, provided none of the interrupts within the group is being used by a peripheral. Otherwise, interrupts coming in from peripherals may be lost by accidentally clearing their flag while modifying the PIEIFR.

To summarize, there are two safe cases when the reserved interrupts could be used as software interrupts:
1) No peripheral within the group is asserting interrupts.
2) No peripheral interrupts are assigned to the group (example PIE group 12).

Table 3−12. PIE Configuration and Control Registers

NAME	ADDRESS	Size (x16)	DESCRIPTION
PIECTRL	0x0000−0CE0	1	PIE, Control Register
PIEACK	0x0000−0CE1	1	PIE, Acknowledge Register
PIEIER1	0x0000−0CE2	1	PIE, INT1 Group Enable Register
PIEIFR1	0x0000−0CE3	1	PIE, INT1 Group Flag Register
PIEIER2	0x0000−0CE4	1	PIE, INT2 Group Enable Register
PIEIFR2	0x0000−0CE5	1	PIE, INT2 Group Flag Register
PIEIER3	0x0000−0CE6	1	PIE, INT3 Group Enable Register
PIEIFR3	0x0000−0CE7	1	PIE, INT3 Group Flag Register
PIEIER4	0x0000−0CE8	1	PIE, INT4 Group Enable Register
PIEIFR4	0x0000−0CE9	1	PIE, INT4 Group Flag Register
PIEIER5	0x0000−0CEA	1	PIE, INT5 Group Enable Register
PIEIFR5	0x0000−0CEB	1	PIE, INT5 Group Flag Register
PIEIER6	0x0000−0CEC	1	PIE, INT6 Group Enable Register
PIEIFR6	0x0000−0CED	1	PIE, INT6 Group Flag Register
PIEIER7	0x0000−0CEE	1	PIE, INT7 Group Enable Register
PIEIFR7	0x0000−0CEF	1	PIE, INT7 Group Flag Register
PIEIER8	0x0000−0CF0	1	PIE, INT8 Group Enable Register
PIEIFR8	0x0000−0CF1	1	PIE, INT8 Group Flag Register
PIEIER9	0x0000−0CF2	1	PIE, INT9 Group Enable Register
PIEIFR9	0x0000−0CF3	1	PIE, INT9 Group Flag Register
PIEIER10	0x0000−0CF4	1	PIE, INT10 Group Enable Register
PIEIFR10	0x0000−0CF5	1	PIE, INT10 Group Flag Register
PIEIER11	0x0000−0CF6	1	PIE, INT11 Group Enable Register
PIEIFR11	0x0000−0CF7	1	PIE, INT11 Group Flag Register
PIEIER12	0x0000−0CF8	1	PIE, INT12 Group Enable Register
PIEIFR12	0x0000−0CF9	1	PIE, INT12 Group Flag Register
Reserved	0x0000−0CFA 0x0000−0CFF	6	Reserved

Note: The PIE configuration and control registers are not protected by EALLOW mode. The PIE vector table is protected.

3.6.1 External Interrupts

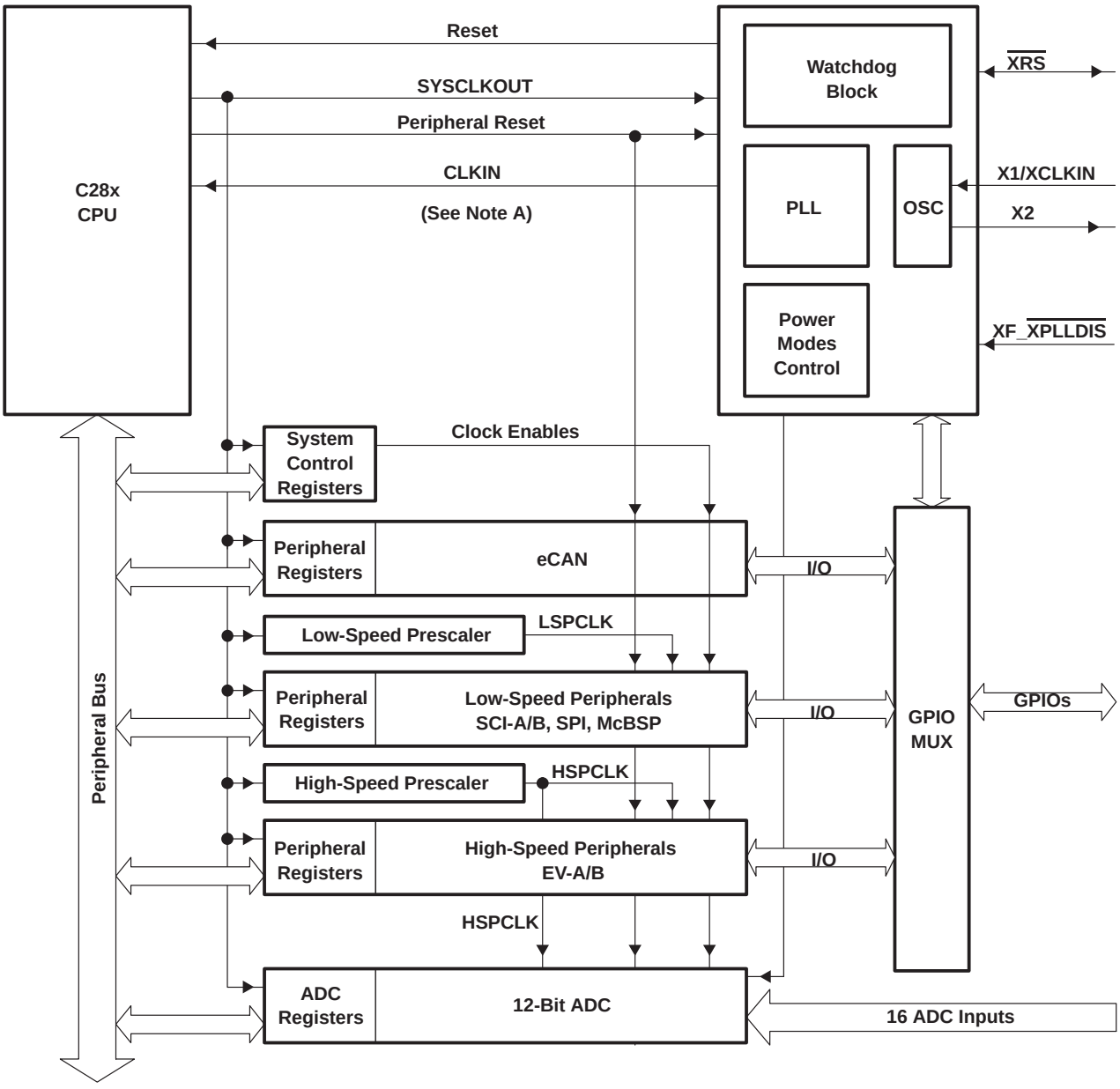
Table 3−13. External Interrupts Registers

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
XINT1CR	0x00 7070	1	XINT1 control register
XINT2CR	0x00 7071	1	XINT2 control register
reserved	0x00 7072 0x00 7076	5	
XNMICR	0x00 7077	1	XNMI control register
XINT1CTR	0x00 7078	1	XINT1 counter register
XINT2CTR	0x00 7079	1	XINT2 counter register
reserved	0x00 707A 0x00 707E	5	
XNMICTR	0x00 707F	1	XNMI counter register

Each external interrupt can be enabled/disabled or qualified using positive or negative going edge. For more information, see the *TMS320x281x System Control and Interrupts Reference Guide* (literature number SPRU078).

3.7 System Control

This section describes the F281x and C281x oscillator, PLL and clocking mechanisms, the watchdog function and the low power modes. Figure 3−8 shows the various clock and reset domains in the F281x and C281x devices that will be discussed.



NOTE A: CLKIN is the clock input to the CPU. SYSCLKOUT is the output clock of the CPU. They are of the same frequency.

Figure 3−8. Clock and Reset Domains

The PLL, clocking, watchdog and low-power modes, are controlled by the registers listed in Table 3−14.

Table 3−14. PLL, Clocking, Watchdog, and Low-Power Mode Registers†

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
reserved	0x00 7010 0x00 7017	8	
reserved	0x00 7018	1	
reserved	0x00 7019	1	
HISPCP	0x00 701A	1	High-Speed Peripheral Clock Prescaler Register for HSPCLK clock
LOSPCP	0x00 701B	1	Low-Speed Peripheral Clock Prescaler Register for LSPCLK clock
PCLKCR	0x00 701C	1	Peripheral Clock Control Register
reserved	0x00 701D	1	
LPMCR0	0x00 701E	1	Low Power Mode Control Register 0
LPMCR1	0x00 701F	1	Low Power Mode Control Register 1
reserved	0x00 7020	1	
PLLCR	0x00 7021	1	PLL Control Register‡
SCSR	0x00 7022	1	System Control & Status Register
WDCNTR	0x00 7023	1	Watchdog Counter Register
reserved	0x00 7024	1	
WDKEY	0x00 7025	1	Watchdog Reset Key Register
reserved	0x00 7026 0x00 7028	3	
WDCR	0x00 7029	1	Watchdog Control Register
reserved	0x00 702A 0x00 702F	6	

† All of the above registers can only be accessed, by executing the EALLOW instruction.
‡ The PLL control register (PLLCR) is reset to a known state by the $\overline{\text{XRS}}$ signal only. Emulation reset (through Code Composer Studio) will not reset PLLCR.

3.8 OSC and PLL Block

Figure 3−9 shows the OSC and PLL block on the F281x and C281x.

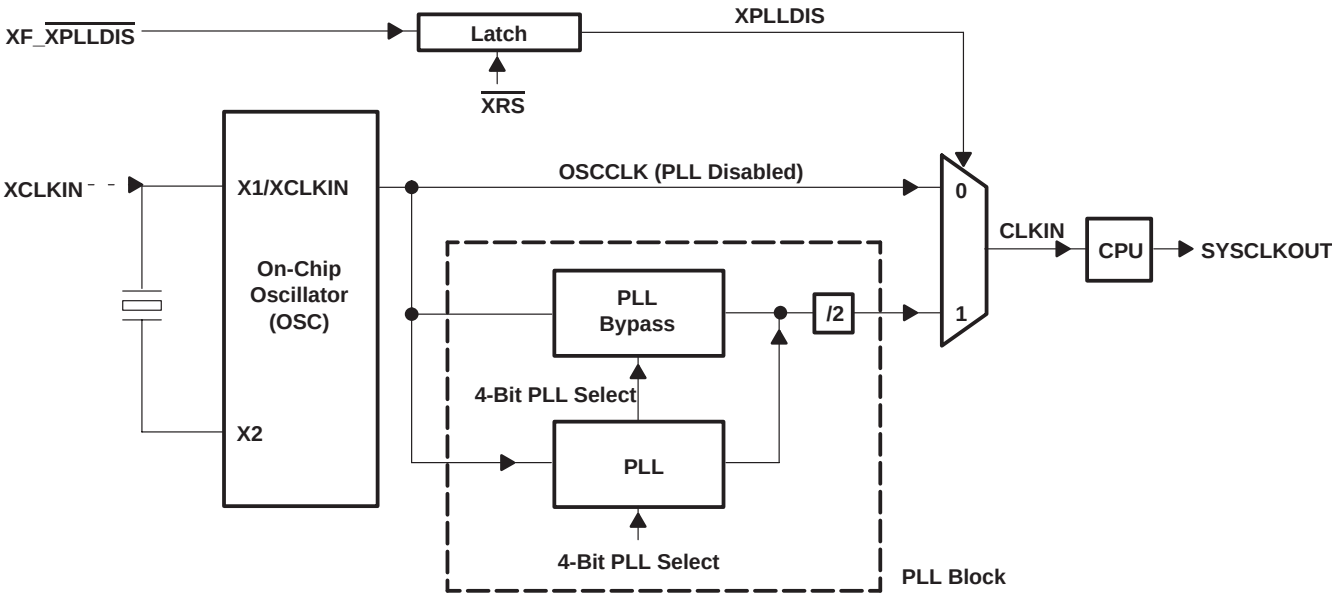


Figure 3−9. OSC and PLL Block

The on-chip oscillator circuit enables a crystal to be attached to the F281x and C281x devices using the X1/XCLKIN and X2 pins. If a crystal is not used, then an external oscillator can be directly connected to the X1/XCLKIN pin and the X2 pin is left unconnected. The logic-high level in this case should not exceed V_{DD}. The PLLCR bits [3:0] set the clocking ratio.

Table 3−15. PLLCR Register Bit Definitions

BIT(S)	NAME	TYPE	$\overline{\text{XRS}}$ RESET†	DESCRIPTION
15:4	reserved	R = 0	0:0	
3:0	DIV	R/W	0,0,0,0	SYSCLKOUT = (XCLKIN * n)/2, where n is the PLL multiplication factor. **Bit Value** **n** **SYSCLKOUT** 0000 PLL Bypassed XCLKIN/2 0001 1 XCLKIN/2 0010 2 XCLKIN 0011 3 XCLKIN * 1.5 0100 4 XCLKIN * 2 0101 5 XCLKIN * 2.5 0110 6 XCLKIN * 3 0111 7 XCLKIN * 3.5 1000 8 XCLKIN * 4 1001 9 XCLKIN * 4.5 1010 10 XCLKIN * 5 1011 11 Reserved 1100 12 Reserved 1101 13 Reserved 1110 14 Reserved 1111 15 Reserved

† The PLLCR register is reset to a known state by the $\overline{\text{XRS}}$ reset line. **If a reset is issued by the debugger, the PLL clocking ratio is not changed.**

3.8.1 Loss of Input Clock

In PLL enabled mode, if the input clock XCLKIN or the oscillator clock is removed or absent, the PLL will still issue a “limp-mode” clock. The limp-mode clock will continue to clock the CPU and peripherals at a typical frequency of 1−4 MHz. The PLLCR register should have been written to with a non-zero value for this feature to work.

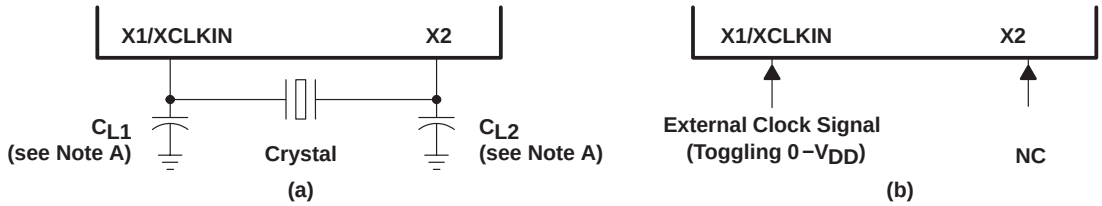
Normally, when the input clocks are present, the watchdog counter will decrement to initiate a watchdog reset or WDINT interrupt. However, when the external input clock fails, the watchdog counter will stop decrementing (i.e., the watchdog counter does not change with the limp-mode clock). This condition could be used by the application firmware to detect the input clock failure and initiate necessary shut-down procedure for the system.

3.9 PLL-Based Clock Module

The F281x and C281x have an on-chip, PLL-based clock module. This module provides all the necessary clocking signals for the device, as well as control for low-power mode entry. The PLL has a 4-bit ratio control to select different CPU clock rates. The watchdog module should be disabled before writing to the PLLCR register. It can be re-enabled (if need be) after the PLL module has stabilized, which takes 131072 XCLKIN cycles.

The PLL-based clock module provides two modes of operation:

- Crystal-operation
  This mode allows the use of an external crystal/resonator to provide the time base to the device.
- External clock source operation
  This mode allows the internal oscillator to be bypassed. The device clocks are generated from an external clock source input on the X1/XCLKIN pin.



NOTE A: TI recommends that customers have the resonator/crystal vendor characterize the operation of their device with the DSP chip. The resonator/crystal vendor has the equipment and expertise to tune the tank circuit. The vendor can also advise the customer regarding the proper tank component values that will ensure start-up and stability over the entire operating range.

Figure 3−10. Recommended Crystal/Clock Connection

Table 3−16. Possible PLL Configuration Modes

PLL MODE	REMARKS	SYSCLKOUT
PLL Disabled	Invoked by tying $\overline{\text{XPLLDIS}}$ pin low upon reset. PLL block is completely disabled. Clock input to the CPU (CLKIN) is directly derived from the clock signal present at the X1/XCLKIN pin.	XCLKIN
PLL Bypassed	Default PLL configuration upon power-up, if PLL is not disabled. The PLL itself is bypassed. However, the /2 module in the PLL block divides the clock input at the X1/XCLKIN pin by two before feeding it to the CPU.	XCLKIN/2
PLL Enabled	Achieved by writing a non-zero value “n” into PLLCR register. The /2 module in the PLL block now divides the output of the PLL by two before feeding it to the CPU.	(XCLKIN * n) / 2

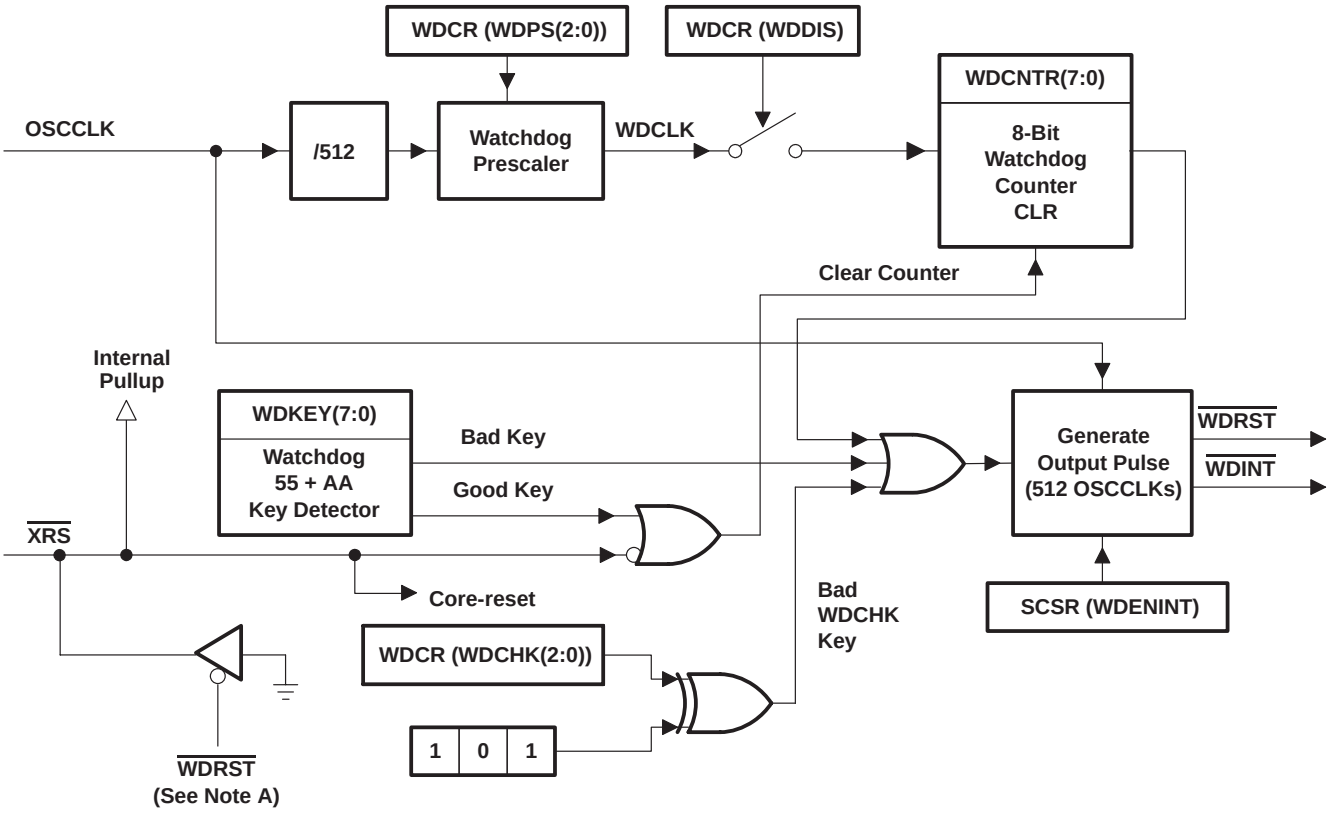
3.10 External Reference Oscillator Clock Option

The typical specifications for the external quartz crystal for a frequency of 30 MHz are listed below:

- Fundamental mode, parallel resonant
- C_L (load capacitance) = 12 pF
- $C_{L1} = C_{L2}$ = 24 pF
- C_{shunt} = 6 pF
- ESR range = 25 to 40 Ω

3.11 Watchdog Block

The watchdog block on the F281x and C281x is identical to the one used on the 240x devices. The watchdog module generates an output pulse, 512 oscillator clocks wide (OSCCLK), whenever the 8-bit watchdog up counter has reached its maximum value. To prevent this, the user disables the counter or the software must periodically write a 0x55 + 0xAA sequence into the watchdog key register which will reset the watchdog counter. Figure 3−11 shows the various functional blocks within the watchdog module.



NOTE A: The $\overline{\text{WDRST}}$ signal is driven low for 512 OSCCLK cycles.

Figure 3−11. Watchdog Module

The $\overline{\text{WDINT}}$ signal enables the watchdog to be used as a wakeup from IDLE/STANDBY mode timer.

In STANDBY mode, all peripherals are turned off on the device. The only peripheral that remains functional is the watchdog. The WATCHDOG module will run off the PLL clock or the oscillator clock. The $\overline{\text{WDINT}}$ signal is fed to the LPM block so that it can wake the device from STANDBY (if enabled). See Section 3.12, Low-Power Modes Block, for more details.

In IDLE mode, the $\overline{\text{WDINT}}$ signal can generate an interrupt to the CPU, via the PIE, to take the CPU out of IDLE mode.

In HALT mode, this feature cannot be used because the oscillator (and PLL) are turned off and hence so is the WATCHDOG.

3.12 Low-Power Modes Block

The low-power modes on the F281x and C281x are similar to the 240x devices. Table 3−17 summarizes the various modes.

Table 3−17. F281x and C281x Low-Power Modes

MODE	LPM(1:0)	OSCCLK	CLKIN	SYSCLKOUT	EXIT†
Normal	X,X	on	on	on	–
IDLE	0,0	on	on	on‡	$\overline{\text{XRS}}$, $\overline{\text{WDINT}}$, Any Enabled Interrupt, XNMI Debugger§
STANDBY	0,1	on (watchdog still running)	off	off	$\overline{\text{XRS}}$, $\overline{\text{WDINT}}$, XINT1, XNMI, $\overline{\text{T1/2/3/4CTRIP}}$, $\overline{\text{C1/2/3/4/5/6TRIP}}$, SCIRXDA, SCIRXDB, CANRX, Debugger§
HALT	1,X	off (oscillator and PLL turned off, watchdog not functional)	off	off	$\overline{\text{XRS}}$, XNMI, Debugger§

† The Exit column lists which signals or under what conditions the low power mode will be exited. A low signal, on any of the signals, will exit the low power condition. This signal must be kept low long enough for an interrupt to be recognized by the device. Otherwise the IDLE mode will not be exited and the device will go back into the indicated low power mode.

‡ The IDLE mode on the C28x behaves differently than on the 24x/240x. On the C28x, the clock output from the core (SYSCLKOUT) is still functional while on the 24x/240x the clock is turned off.

§ On the C28x, the JTAG port can still function even if the core clock (CLKIN) is turned off.

The various low-power modes operate as follows:

IDLE Mode: This mode is exited by any enabled interrupt or an XNMI that is recognized by the processor. The LPM block performs no tasks during this mode as long as the LPMCR0(LPM) bits are set to 0,0.

STANDBY Mode: All other signals (including XNMI) will wake the device from STANDBY mode if selected by the LPMCR1 register. The user will need to select which signal(s) will wake the device. The selected signal(s) are also qualified by the OSCCLK before waking the device. The number of OSCCLKs is specified in the LPMCR0 register.

HALT Mode: Only the $\overline{\text{XRS}}$ and XNMI external signals can wake the device from HALT mode. The XNMI input to the core has an enable/disable bit. Hence, it is safe to use the XNMI signal for this function.

NOTE: The low-power modes do not affect the state of the output pins (PWM pins included). They will be in whatever state the code left them in when the IDLE instruction was executed.

4 Peripherals

The integrated peripherals of the F281x and C281x are described in the following subsections:

- Three 32-bit CPU-Timers
- Two event-manager modules (EVA, EVB)
- Enhanced analog-to-digital converter (ADC) module
- Enhanced controller area network (eCAN) module
- Multichannel buffered serial port (McBSP) module
- Serial communications interface modules (SCI-A, SCI-B)
- Serial peripheral interface (SPI) module
- Digital I/O and shared pin functions

4.1 32-Bit CPU-Timers 0/1/2

There are three 32-bit CPU-timers on the F281x and C281x devices (CPU-TIMER0/1/2).

CPU-Timer 1 is reserved for TI system functions and Timer 2 is reserved for DSP/BIOS. CPU-Timer 0 can be used in user applications. These timers are different from the general-purpose (GP) timers that are present in the Event Manager modules (EVA, EVB).

NOTE: If the application is not using DSP/BIOS, then CPU-Timers 1 and 2 can be used in the application.

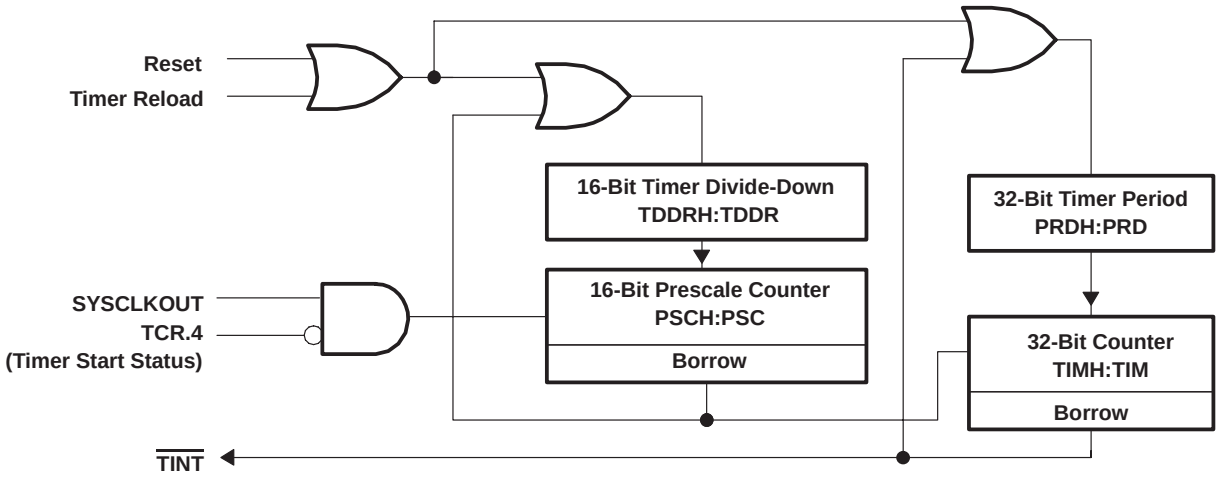
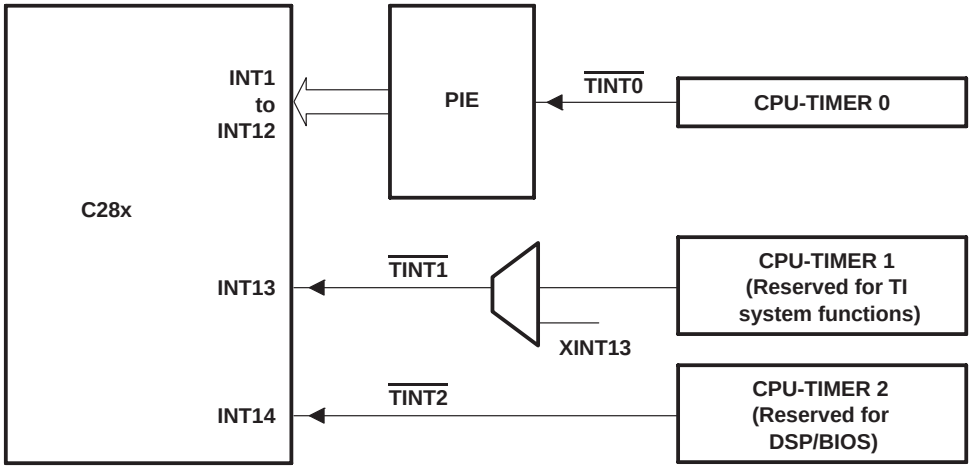


Figure 4−1. CPU-Timers

In the F281x and C281x devices, the timer interrupt signals ($\overline{\text{TINT0}}$, $\overline{\text{TINT1}}$, $\overline{\text{TINT2}}$) are connected as shown in Figure 4−2.



NOTES: A. The timer registers are connected to the Memory Bus of the C28x processor.
B. The timing of the timers is synchronized to SYSCLKOUT of the processor clock.

Figure 4−2. CPU-Timer Interrupts Signals and Output Signal (See Notes A and B)

The general operation of the timer is as follows: The 32-bit counter register “TIMH:TIM” is loaded with the value in the period register “PRDH:PRD”. The counter register, decrements at the SYSCLKOUT rate of the C28x. When the counter reaches 0, a timer interrupt output signal generates an interrupt pulse. The registers listed in Table 4−1 are used to configure the timers. For more information, see the *TMS320x281x System Control and Interrupts Reference Guide* (literature number SPRU078).

Table 4−1. CPU-Timers 0, 1, 2 Configuration and Control Registers

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
TIMER0TIM	0x00 0C00	1	CPU-Timer 0, Counter Register
TIMER0TIMH	0x00 0C01	1	CPU-Timer 0, Counter Register High
TIMER0PRD	0x00 0C02	1	CPU-Timer 0, Period Register
TIMER0PRDH	0x00 0C03	1	CPU-Timer 0, Period Register High
TIMER0TCR	0x00 0C04	1	CPU-Timer 0, Control Register
reserved	0x00 0C05	1	
TIMER0TPR	0x00 0C06	1	CPU-Timer 0, Prescale Register
TIMER0TPRH	0x00 0C07	1	CPU-Timer 0, Prescale Register High
TIMER1TIM	0x00 0C08	1	CPU-Timer 1, Counter Register
TIMER1TIMH	0x00 0C09	1	CPU-Timer 1, Counter Register High
TIMER1PRD	0x00 0C0A	1	CPU-Timer 1, Period Register
TIMER1PRDH	0x00 0C0B	1	CPU-Timer 1, Period Register High
TIMER1TCR	0x00 0C0C	1	CPU-Timer 1, Control Register
reserved	0x00 0C0D	1	
TIMER1TPR	0x00 0C0E	1	CPU-Timer 1, Prescale Register
TIMER1TPRH	0x00 0C0F	1	CPU-Timer 1, Prescale Register High
TIMER2TIM	0x00 0C10	1	CPU-Timer 2, Counter Register
TIMER2TIMH	0x00 0C11	1	CPU-Timer 2, Counter Register High
TIMER2PRD	0x00 0C12	1	CPU-Timer 2, Period Register
TIMER2PRDH	0x00 0C13	1	CPU-Timer 2, Period Register High
TIMER2TCR	0x00 0C14	1	CPU-Timer 2, Control Register
reserved	0x00 0C15	1	
TIMER2TPR	0x00 0C16	1	CPU-Timer 2, Prescale Register
TIMER2TPRH	0x00 0C17	1	CPU-Timer 2, Prescale Register High
reserved	0x00 0C18 0x00 0C3F	40	

4.2 Event Manager Modules (EVA, EVB)

The event-manager modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits. EVA and EVB timers, compare units, and capture units function identically. However, timer/unit names differ for EVA and EVB. Table 4−2 shows the module and signal names used. Table 4−2 shows the features and functionality available for the event-manager modules and highlights EVA nomenclature.

Event managers A and B have identical peripheral register sets with EVA starting at 7400h and EVB starting at 7500h. The paragraphs in this section describe the function of GP timers, compare units, capture units, and QEPs using EVA nomenclature. These paragraphs are applicable to EVB with regard to function—however, module/signal names would differ. Table 4−3 lists the EVA registers. For more information, see the *TMS320x281x DSP Event Manager (EV) Reference Guide* (literature number SPRU065).

Table 4−2. Module and Signal Names for EVA and EVB

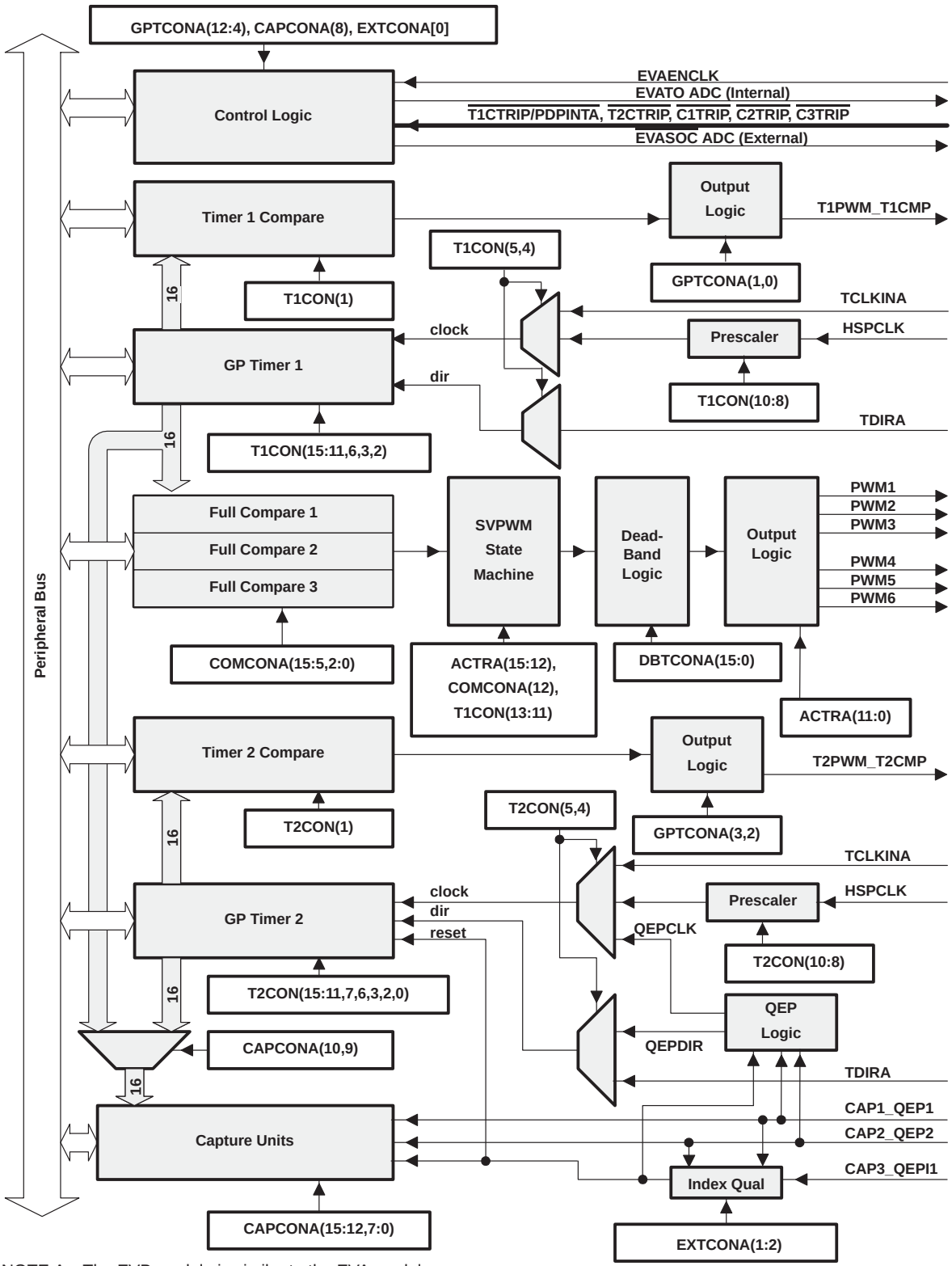
EVENT MANAGER MODULES	EVA		EVB	
	MODULE	SIGNAL	MODULE	SIGNAL
GP Timers	GP Timer 1 GP Timer 2	T1PWM/T1CMP T2PWM/T2CMP	GP Timer 3 GP Timer 4	T3PWM/T3CMP T4PWM/T4CMP
Compare Units	Compare 1 Compare 2 Compare 3	PWM1/2 PWM3/4 PWM5/6	Compare 4 Compare 5 Compare 6	PWM7/8 PWM9/10 PWM11/12
Capture Units	Capture 1 Capture 2 Capture 3	CAP1 CAP2 CAP3	Capture 4 Capture 5 Capture 6	CAP4 CAP5 CAP6
QEP Channels	QEP1 QEP2 QEPI1	QEP1 QEP2	QEP3 QEP4 QEPI2	QEP3 QEP4
External Clock Inputs	Direction External Clock	TDIRA TCLKINA	Direction External Clock	TDIRB TCLKINB
External Trip Inputs	Compare	$\overline{\text{C1TRIP}}$ $\overline{\text{C2TRIP}}$ $\overline{\text{C3TRIP}}$	Compare	$\overline{\text{C4TRIP}}$ $\overline{\text{C5TRIP}}$ $\overline{\text{C6TRIP}}$
External Trip Inputs		$\overline{\text{T1CTRIP\_PDPINTA}}$† $\overline{\text{T2CTRIP/EVASOC}}$		$\overline{\text{T3CTRIP\_PDPINTB}}$† $\overline{\text{T4CTRIP/EVBSOC}}$

† In the 24x/240x-compatible mode, the $\overline{\text{T1CTRIP\_PDPINTA}}$ pin functions as $\overline{\text{PDPINTA}}$ and the $\overline{\text{T3CTRIP\_PDPINTB}}$ pin functions as $\overline{\text{PDPINTB}}$.

Table 4−3. EVA Registers†

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
GPTCONA	0x00 7400	1	GP Timer Control Register A
T1CNT	0x00 7401	1	GP Timer 1 Counter Register
T1CMPR	0x00 7402	1	GP Timer 1 Compare Register
T1PR	0x00 7403	1	GP Timer 1 Period Register
T1CON	0x00 7404	1	GP Timer 1 Control Register
T2CNT	0x00 7405	1	GP Timer 2 Counter Register
T2CMPR	0x00 7406	1	GP Timer 2 Compare Register
T2PR	0x00 7407	1	GP Timer 2 Period Register
T2CON	0x00 7408	1	GP Timer 2 Control Register
EXTCONA‡	0x00 7409	1	GP Extension Control Register A
COMCONA	0x00 7411	1	Compare Control Register A
ACTRA	0x00 7413	1	Compare Action Control Register A
DBTCONA	0x00 7415	1	Dead-Band Timer Control Register A
CMPR1	0x00 7417	1	Compare Register 1
CMPR2	0x00 7418	1	Compare Register 2
CMPR3	0x00 7419	1	Compare Register 3
CAPCONA	0x00 7420	1	Capture Control Register A
CAPFIFOA	0x00 7422	1	Capture FIFO Status Register A
CAP1FIFO	0x00 7423	1	Two-Level Deep Capture FIFO Stack 1
CAP2FIFO	0x00 7424	1	Two-Level Deep Capture FIFO Stack 2
CAP3FIFO	0x00 7425	1	Two-Level Deep Capture FIFO Stack 3
CAP1FBOT	0x00 7427	1	Bottom Register Of Capture FIFO Stack 1
CAP2FBOT	0x00 7428	1	Bottom Register Of Capture FIFO Stack 2
CAP3FBOT	0x00 7429	1	Bottom Register Of Capture FIFO Stack 3
EVAIMRA	0x00 742C	1	Interrupt Mask Register A
EVAIMRB	0x00 742D	1	Interrupt Mask Register B
EVAIMRC	0x00 742E	1	Interrupt Mask Register C
EVAIFRA	0x00 742F	1	Interrupt Flag Register A
EVAIFRB	0x00 7430	1	Interrupt Flag Register B
EVAIFRC	0x00 7431	1	Interrupt Flag Register C

† The EV-B register set is identical except the address range is from 0x00−7500 to 0x00−753F. The above registers are mapped to Zone 2. This space allows only 16-bit accesses. 32-bit accesses produce undefined results.
‡ New register compared to 24x/240x



NOTE A: The EVB module is similar to the EVA module.

Figure 4−3. Event Manager A Functional Block Diagram (See Note A)

4.2.1 General-Purpose (GP) Timers

There are two GP timers. The GP timer x (x = 1 or 2 for EVA; x = 3 or 4 for EVB) includes:

- A 16-bit timer, up-/down-counter, TxCNT, for reads or writes
- A 16-bit timer-compare register, TxCMPR (double-buffered with shadow register), for reads or writes
- A 16-bit timer-period register, TxPR (double-buffered with shadow register), for reads or writes
- A 16-bit timer-control register,TxCON, for reads or writes
- Selectable internal or external input clocks
- A programmable prescaler for internal or external clock inputs
- Control and interrupt logic, for four maskable interrupts: *underflow*, *overflow*, *timer compare*, and *period interrupts*
- A selectable direction input pin (TDIRx) (to count up or down when directional up-/down-count mode is selected)

The GP timers can be operated independently or synchronized with each other. The compare register associated with each GP timer can be used for compare function and PWM-waveform generation. There are three continuous modes of operations for each GP timer in up- or up/down-counting operations. Internal or external input clocks with programmable prescaler are used for each GP timer. GP timers also provide the time base for the other event-manager submodules: GP timer 1 for all the compares and PWM circuits, GP timer 2/1 for the capture units and the quadrature-pulse counting operations. Double-buffering of the period and compare registers allows programmable change of the timer (PWM) period and the compare/PWM pulse width as needed.

4.2.2 Full-Compare Units

There are three full-compare units on each event manager. These compare units use GP timer1 as the time base and generate six outputs for compare and PWM-waveform generation using programmable deadband circuit. The state of each of the six outputs is configured independently. The compare registers of the compare units are double-buffered, allowing programmable change of the compare/PWM pulse widths as needed.

4.2.3 Programmable Deadband Generator

Deadband generation can be enabled/disabled for each compare unit output individually. The deadband-generator circuit produces two outputs (with or without deadband zone) for each compare unit output signal. The output states of the deadband generator are configurable and changeable as needed by way of the double-buffered ACTRx register.

4.2.4 PWM Waveform Generation

Up to eight PWM waveforms (outputs) can be generated simultaneously by each event manager: three independent pairs (six outputs) by the three full-compare units with *programmable deadbands*, and two independent PWMs by the GP-timer compares.

4.2.5 Double Update PWM Mode

The F281x and C281x Event Manager supports “Double Update PWM Mode.” This mode refers to a PWM operation mode in which the position of the leading edge and the position of the trailing edge of a PWM pulse are independently modifiable in each PWM period. To support this mode, the compare register that determines the position of the edges of a PWM pulse must allow (buffered) compare value update once at the beginning of a PWM period and another time in the middle of a PWM period. The compare registers in F281x and C281x Event Managers are all buffered and support three compare value reload/update (value in buffer becoming active) modes. These modes have earlier been documented as compare value reload conditions. The reload condition that supports double update PWM mode is reloaded on Underflow (beginning of PWM period) OR Period (middle of PWM period). Double update PWM mode can be achieved by using this condition for compare value reload.

4.2.6 PWM Characteristics

Characteristics of the PWMs are as follows:

- 16-bit registers
- Wide range of programmable deadband for the PWM output pairs
- Change of the PWM carrier frequency for PWM frequency wobbling as needed
- Change of the PWM pulse widths within and after each PWM period as needed
- External-maskable power and drive-protection interrupts
- Pulse-pattern-generator circuit, for programmable generation of asymmetric, symmetric, and four-space vector PWM waveforms
- Minimized CPU overhead using auto-reload of the compare and period registers
- The PWM pins are driven to a high-impedance state when the $\overline{\text{PDPINTx}}$ pin is driven low and **after** $\overline{\text{PDPINTx}}$ signal qualification. The $\overline{\text{PDPINTx}}$ pin (after qualification) is reflected in bit 8 of the COMCONx register.
  - $\overline{\text{PDPINTA}}$ pin status is reflected in bit 8 of COMCONA register.
  - $\overline{\text{PDPINTB}}$ pin status is reflected in bit 8 of COMCONB register.
- EXTCON register bits provide options to individually trip control for each PWM pair of signals

4.2.7 Capture Unit

The capture unit provides a logging function for different events or transitions. The values of the selected GP timer counter is captured and stored in the two-level-deep FIFO stacks when selected transitions are detected on capture input pins, CAPx (x = 1, 2, or 3 for EVA; and x = 4, 5, or 6 for EVB). The capture unit consists of three capture circuits.

- Capture units include the following features:
  - One 16-bit capture control register, CAPCONx (R/W)
  - One 16-bit capture FIFO status register, CAPFIFOx
  - Selection of GP timer 1/2 (for EVA) or 3/4 (for EVB) as the time base
  - Three 16-bit 2-level-deep FIFO stacks, one for each capture unit
  - Three capture input pins (CAP1/2/3 for EVA, CAP4/5/6 for EVB)—one input pin per capture unit. [All inputs are synchronized with the device (CPU) clock. In order for a transition to be captured, the input must hold at its current level to meet the input qualification circuitry requirements. The input pins CAP1/2 and CAP4/5 can also be used as QEP inputs to the QEP circuit.]
  - User-specified transition (rising edge, falling edge, or both edges) detection
  - Three maskable interrupt flags, one for each capture unit
  - The capture pins can also be used as general-purpose interrupt pins, if they are not used for the capture function.

4.2.8 Quadrature-Encoder Pulse (QEP) Circuit

Two capture inputs (CAP1 and CAP2 for EVA; CAP4 and CAP5 for EVB) can be used to interface the on-chip QEP circuit with a quadrature encoder pulse. Full synchronization of these inputs is performed on-chip. Direction or leading-quadrature pulse sequence is detected, and GP timer 2/4 is incremented or decremented by the rising and falling edges of the two input signals (four times the frequency of either input pulse).

With EXTCONA register bits, the EVA QEP circuit can use CAP3 as a capture index pin as well. Similarly, with EXTCONB register bits, the EVB QEP circuit can use CAP6 as a capture index pin.

4.2.9 External ADC Start-of-Conversion

EVA/EVB start-of-conversion (SOC) can be sent to an external pin ($\overline{\text{EVASOC}}$/$\overline{\text{EVBSOC}}$) for external ADC interface. $\overline{\text{EVASOC}}$ and $\overline{\text{EVBSOC}}$ are MUXed with $\overline{\text{T2CTRIP}}$ and $\overline{\text{T4CTRIP}}$, respectively.

4.3 Enhanced Analog-to-Digital Converter (ADC) Module

A simplified functional block diagram of the ADC module is shown in Figure 4−4. The ADC module consists of a 12-bit ADC with a built-in sample-and-hold (S/H) circuit. Functions of the ADC module include:

- 12-bit ADC core with built-in S/H
- Analog input: 0.0 V to 3.0 V (Voltages above 3.0 V produce full-scale conversion results.)
- Fast conversion rate: 80 ns at 25-MHz ADC clock, 12.5 MSPS
- 16-channel, MUXed inputs
- Autosequencing capability provides up to 16 “autoconversions” in a single session. Each conversion can be programmed to select any 1 of 16 input channels
- Sequencer can be operated as two independent 8-state sequencers or as one large 16-state sequencer (i.e., two cascaded 8-state sequencers)
- Sixteen result registers (individually addressable) to store conversion values
  - The digital value of the input analog voltage is derived by:

$$\text{Digital Value} = 0, \quad \text{when input} \le 0\ \text{V}$$

$$\text{Digital Value} = 4096 \times \frac{\text{Input Analog Voltage} - \text{ADCLO}}{3}, \quad \text{when } 0\ \text{V} < \text{input} < 3\ \text{V}$$

$$\text{Digital Value} = 4095, \quad \text{when input} \ge 3\ \text{V}$$

- Multiple triggers as sources for the start-of-conversion (SOC) sequence
  - S/W − software immediate start
  - EVA − Event manager A (multiple event sources within EVA)
  - EVB − Event manager B (multiple event sources within EVB)
- Flexible interrupt control allows interrupt request on every end-of-sequence (EOS) or every other EOS
- Sequencer can operate in “start/stop” mode, allowing multiple “time-sequenced triggers” to synchronize conversions
- EVA and EVB triggers can operate independently in dual-sequencer mode
- Sample-and-hold (S/H) acquisition time window has separate prescale control

The ADC module in the F281x and C281x has been enhanced to provide flexible interface to event managers A and B. The ADC interface is built around a fast, 12-bit ADC module with a fast conversion rate of 80 ns at 25-MHz ADC clock. The ADC module has 16 channels, configurable as two independent 8-channel modules to service event managers A and B. The two independent 8-channel modules can be cascaded to form a 16-channel module. Although there are multiple input channels and two sequencers, there is only one converter in the ADC module. Figure 4−4 shows the block diagram of the F281x and C281x ADC module.

The two 8-channel modules have the capability to autosequence a series of conversions, each module has the choice of selecting any one of the respective eight channels available through an analog MUX. In the cascaded mode, the autosequencer functions as a single 16-channel sequencer. On each sequencer, once the conversion is complete, the selected channel value is stored in its respective RESULT register. Autosequencing allows the system to convert the same channel multiple times, allowing the user to perform oversampling algorithms. This gives increased resolution over traditional single-sampled conversion results.

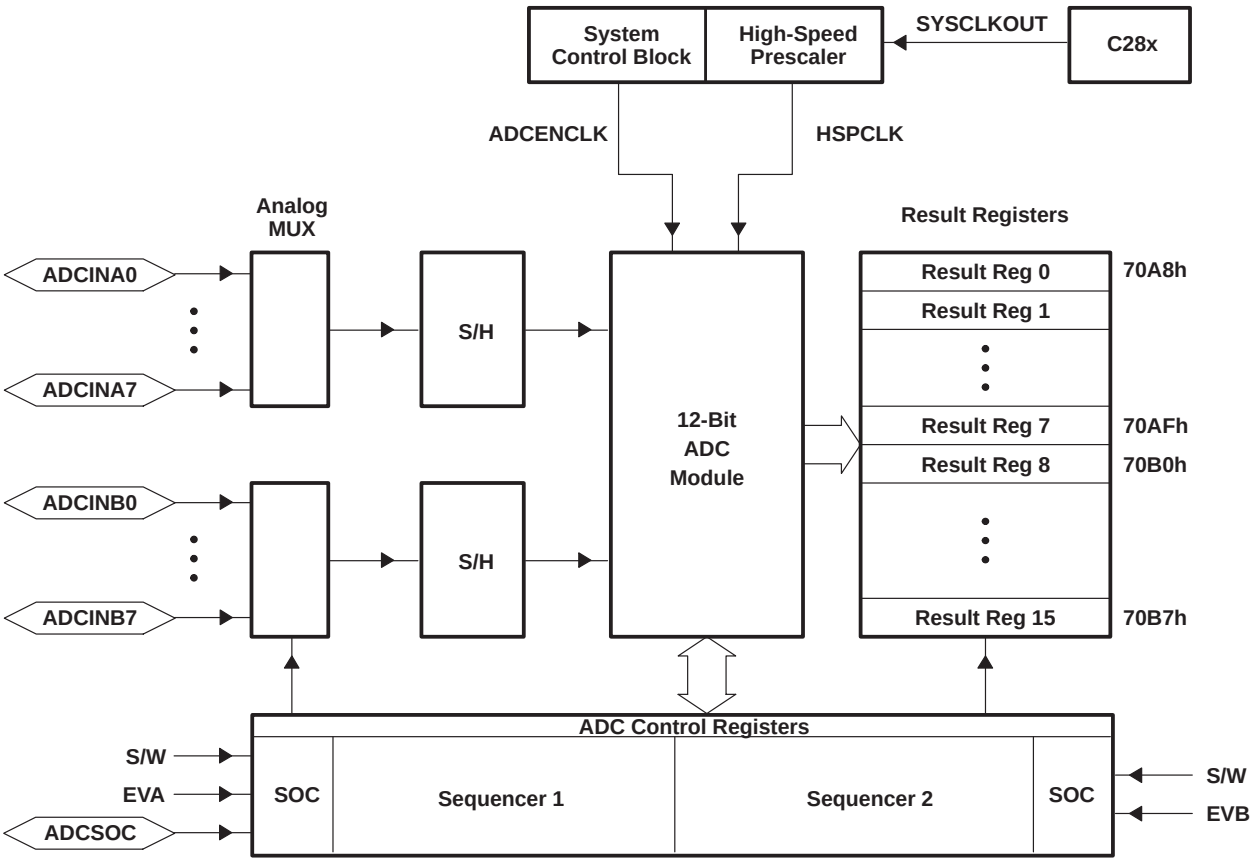


Figure 4−4. Block Diagram of the F281x and C281x ADC Module

To obtain the specified accuracy of the ADC, proper board layout is very critical. To the best extent possible, traces leading to the ADCIN pins should not run in close proximity to the digital signal paths. This is to minimize switching noise on the digital lines from getting coupled to the ADC inputs. Furthermore, proper isolation techniques must be used to isolate the ADC module power pins (V_{DDA1}/V_{DDA2}, $AV_{DDREFBG}$) from the digital supply. Figure 4−5 shows the ADC pin connections for the F281x and C281x devices.

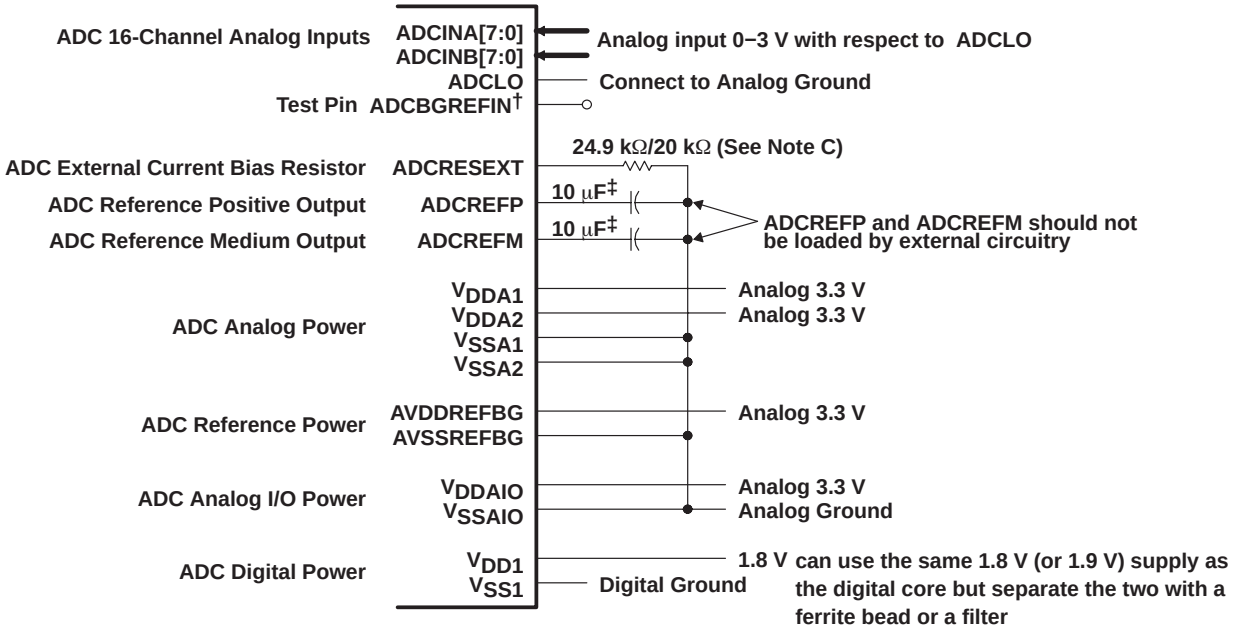
Notes:

1. The ADC registers are accessed at the SYSCLKOUT rate. The internal timing of the ADC module is controlled by the high-speed peripheral clock (HSPCLK).
2. The behavior of the ADC module based on the state of the ADCENCLK and HALT signals is as follows:

   ADCENCLK: On reset, this signal will be low. While reset is active-low ($\overline{XRS}$) the clock to the register will still function. This is necessary to make sure all registers and modes go into their default reset state. The analog module will however be in a low-power inactive state. As soon as reset goes high, then the clock to the registers will be disabled. When the user sets the ADCENCLK signal high, then the clocks to the registers will be enabled and the analog module will be enabled. There will be a certain time delay (ms range) before the ADC is stable and can be used.

   HALT: This signal only affects the analog module. It does not affect the registers. If low, the ADC module is powered. If high, the ADC module goes into low-power mode. The HALT mode will stop the clock to the CPU, which will stop the HSPCLK. Therefore the ADC register logic will be turned off indirectly.

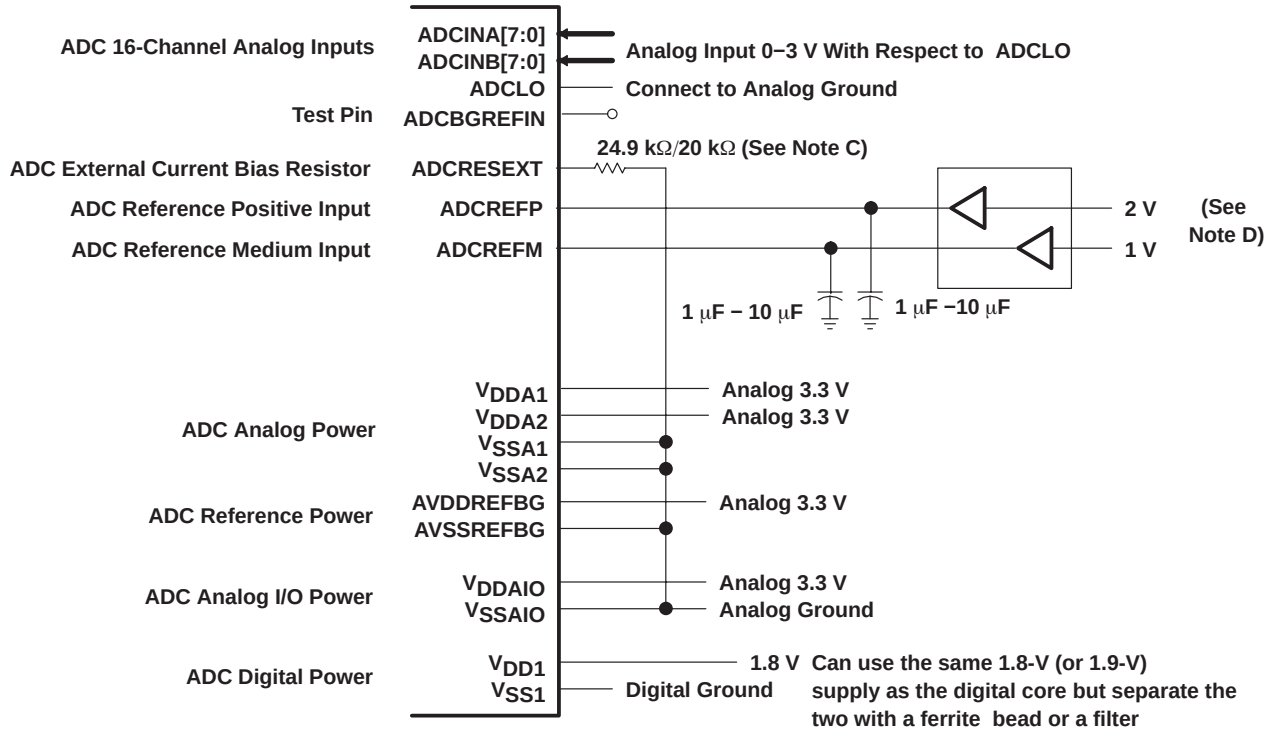
Figure 4−5 shows the ADC pin-biasing for internal reference and Figure 4−6 shows the ADC pin-biasing for external reference.



† Provide access to this pin in PCB layouts. Intended for test purposes only.
‡ TAIYO YUDEN EMK325F106ZH, EMK325BJ106MD, or equivalent
NOTES: A. External decoupling capacitors are recommended on all power pins.
B. Analog inputs must be driven from an operational amplifier that does not degrade the ADC performance.
C. Use 24.9 kΩ for ADC clock range 1 − 18.75 MHz; use 20 kΩ for ADC clock range 18.75 − 25 MHz.

Figure 4−5. ADC Pin Connections With Internal Reference (See Notes A and B)

NOTE:
The temperature rating of any recommended component must match the rating of the end product.



NOTES: A. External decoupling capacitors are recommended on all power pins.
B. Analog inputs must be driven from an operational amplifier that does not degrade the ADC performance.
C. Use 24.9 kΩ for ADC clock range 1 − 18.75 MHz; use 20 kΩ for ADC clock range 18.75 − 25 MHz.
D. It is recommended that buffered external references be provided with a voltage difference of (ADCREFP-ADCREFM) = 1 V ± 0.1% or better.

External reference is enabled using bit 8 in the ADCTRL3 Register at ADC power up. In this mode, the accuracy of external reference is critical for overall gain. The voltage ADCREFP−ADCREFM will determine the overall accuracy. Do not enable internal references when external references are connected to ADCREFP and ADCREFM. See the *TMS320x281x DSP Analog-to-Digital Converter (ADC) Reference Guide* (literature number SPRU060) for more information.

Figure 4−6. ADC Pin Connections With External Reference

The ADC operation is configured, controlled, and monitored by the registers listed in Table 4−4.

Table 4−4. ADC Registers†

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
ADCTRL1	0x00 7100	1	ADC Control Register 1
ADCTRL2	0x00 7101	1	ADC Control Register 2
ADCMAXCONV	0x00 7102	1	ADC Maximum Conversion Channels Register
ADCCHSELSEQ1	0x00 7103	1	ADC Channel Select Sequencing Control Register 1
ADCCHSELSEQ2	0x00 7104	1	ADC Channel Select Sequencing Control Register 2
ADCCHSELSEQ3	0x00 7105	1	ADC Channel Select Sequencing Control Register 3
ADCCHSELSEQ4	0x00 7106	1	ADC Channel Select Sequencing Control Register 4
ADCASEQSR	0x00 7107	1	ADC Auto-Sequence Status Register
ADCRESULT0	0x00 7108	1	ADC Conversion Result Buffer Register 0
ADCRESULT1	0x00 7109	1	ADC Conversion Result Buffer Register 1
ADCRESULT2	0x00 710A	1	ADC Conversion Result Buffer Register 2
ADCRESULT3	0x00 710B	1	ADC Conversion Result Buffer Register 3
ADCRESULT4	0x00 710C	1	ADC Conversion Result Buffer Register 4
ADCRESULT5	0x00 710D	1	ADC Conversion Result Buffer Register 5
ADCRESULT6	0x00 710E	1	ADC Conversion Result Buffer Register 6
ADCRESULT7	0x00 710F	1	ADC Conversion Result Buffer Register 7
ADCRESULT8	0x00 7110	1	ADC Conversion Result Buffer Register 8
ADCRESULT9	0x00 7111	1	ADC Conversion Result Buffer Register 9
ADCRESULT10	0x00 7112	1	ADC Conversion Result Buffer Register 10
ADCRESULT11	0x00 7113	1	ADC Conversion Result Buffer Register 11
ADCRESULT12	0x00 7114	1	ADC Conversion Result Buffer Register 12
ADCRESULT13	0x00 7115	1	ADC Conversion Result Buffer Register 13
ADCRESULT14	0x00 7116	1	ADC Conversion Result Buffer Register 14
ADCRESULT15	0x00 7117	1	ADC Conversion Result Buffer Register 15
ADCTRL3	0x00 7118	1	ADC Control Register 3
ADCST	0x00 7119	1	ADC Status Register
reserved	0x00 711C 0x00 711F	4	

† The above registers are Peripheral Frame 2 Registers.

4.4 Enhanced Controller Area Network (eCAN) Module

The CAN module has the following features:

- Fully compliant with CAN protocol, version 2.0B
- Supports data rates up to 1 Mbps
- Thirty-two mailboxes, each with the following properties:
  - Configurable as receive or transmit
  - Configurable with standard or extended identifier
  - Has a programmable receive mask
  - Supports data and remote frame
  - Composed of 0 to 8 bytes of data
  - Uses a 32-bit time stamp on receive and transmit message
  - Protects against reception of new message
  - Holds the dynamically programmable priority of transmit message
  - Employs a programmable interrupt scheme with two interrupt levels
  - Employs a programmable alarm on transmission or reception time-out
- Low-power mode
- Programmable wake-up on bus activity
- Automatic reply to a remote request message
- Automatic retransmission of a frame in case of loss of arbitration or error
- 32-bit local network time counter synchronized by a specific message (communication in conjunction with mailbox 16)
- Self-test mode
  - Operates in a loopback mode receiving its own message. A “dummy” acknowledge is provided, thereby eliminating the need for another node to provide the acknowledge bit.

NOTE: For a SYSCLKOUT of 150 MHz, the smallest bit rate possible is 23.4 kbps.

The 28x CAN has passed the conformance test per ISO/DIS 16845. Contact TI for further details.

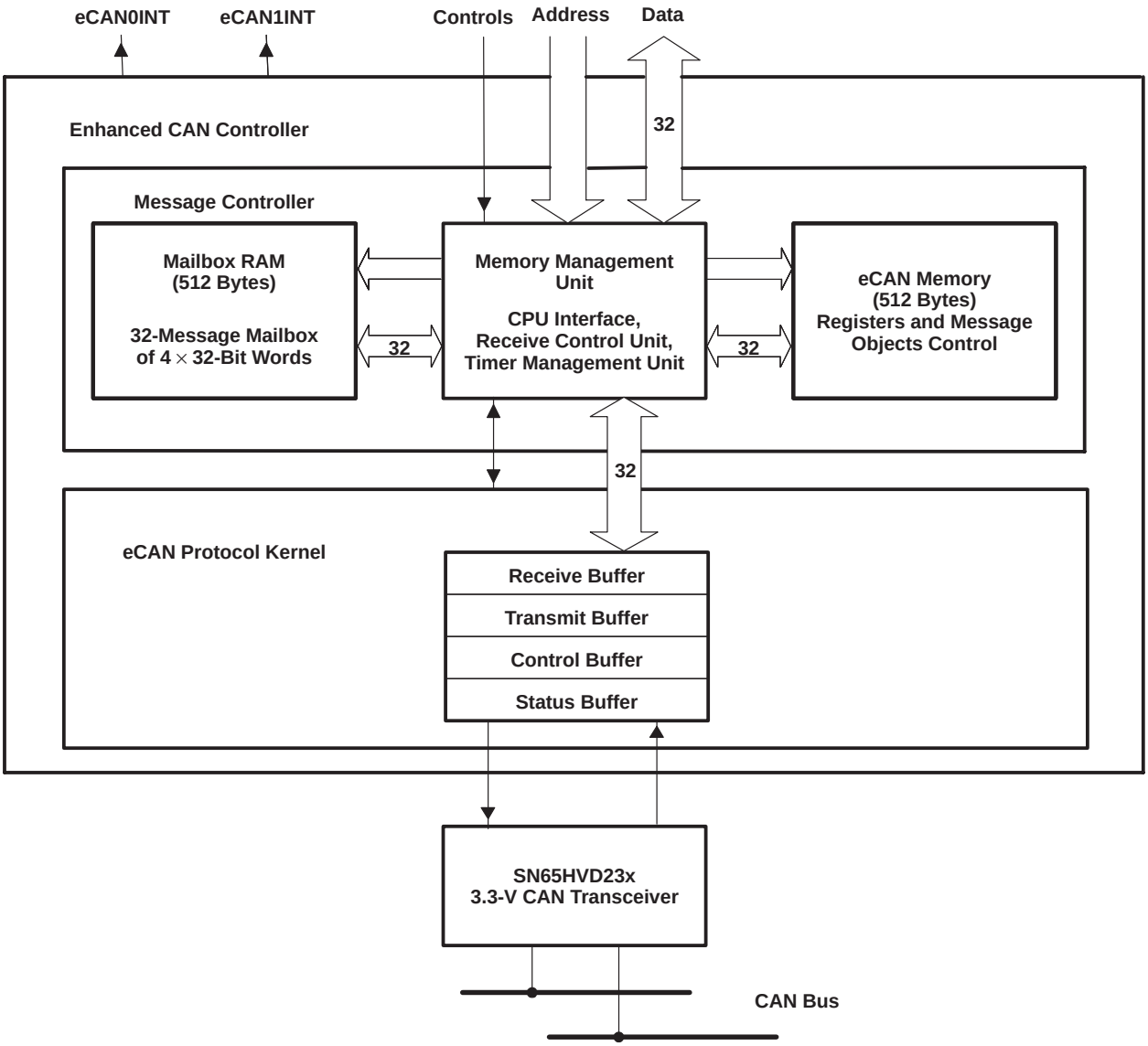


Figure 4−7. eCAN Block Diagram and Interface Circuit

Table 4−5. 3.3-V eCAN Transceivers for the TMS320F281x and TMS320C281x DSPs

PART NUMBER	SUPPLY VOLTAGE	LOW-POWER MODE	SLOPE CONTROL	VREF	OTHER	T_A
SN65HVD230	3.3 V	Standby	Adjustable	Yes	−−	−40°C to 85°C
SN65HVD230Q	3.3 V	Standby	Adjustable	Yes	−−	−40°C to 125°C
SN65HVD231	3.3 V	Sleep	Adjustable	Yes	−−	−40°C to 85°C
SN65HVD231Q	3.3 V	Sleep	Adjustable	Yes	−−	−40°C to 125°C
SN65HVD232	3.3 V	None	None	None	−−	−40°C to 85°C
SN65HVD232Q	3.3 V	None	None	None	−−	−40°C to 125°C
SN65HVD233	3.3 V	Standby	Adjustable	None	Diagnostic Loopback	−40°C to 125°C
SN65HVD234	3.3 V	Standby & Sleep	Adjustable	None	−−	−40°C to 125°C
SN65HVD235	3.3 V	Standby	Adjustable	None	Autobaud Loopback	−40°C to 125°C

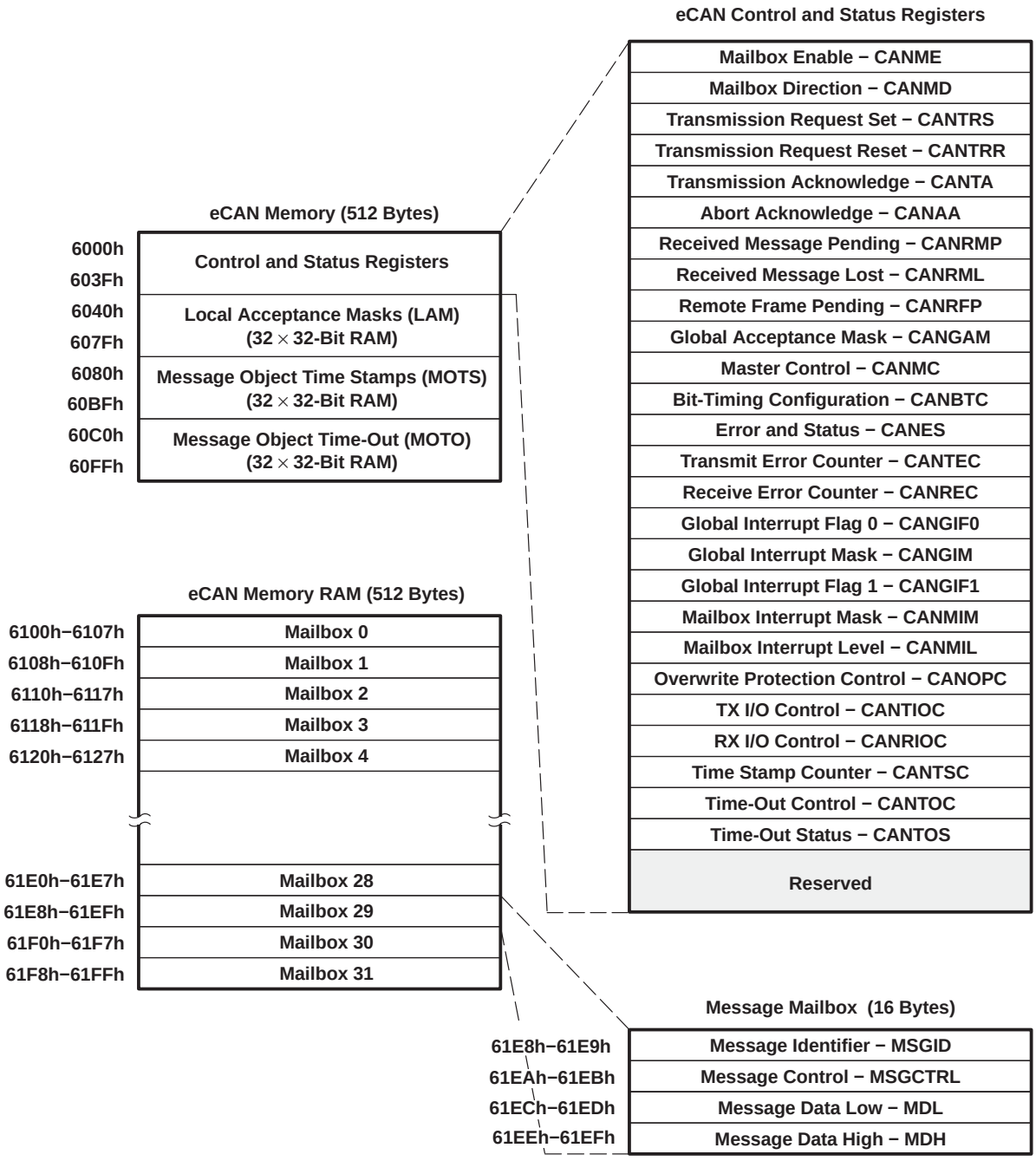


Figure 4−8. eCAN Memory Map

The CAN registers listed in Table 4−6 are used by the CPU to configure and control the CAN controller and the message objects. eCAN control registers only support 32-bit read/write operations. Mailbox RAM can be accessed as 16 bits or 32 bits. 32-bit accesses are aligned to an even boundary.

Table 4−6. CAN Registers Map†

REGISTER NAME	ADDRESS	SIZE (x32)	DESCRIPTION
CANME	0x00 6000	1	Mailbox enable
CANMD	0x00 6002	1	Mailbox direction
CANTRS	0x00 6004	1	Transmit request set
CANTRR	0x00 6006	1	Transmit request reset
CANTA	0x00 6008	1	Transmission acknowledge
CANAA	0x00 600A	1	Abort acknowledge
CANRMP	0x00 600C	1	Receive message pending
CANRML	0x00 600E	1	Receive message lost
CANRFP	0x00 6010	1	Remote frame pending
CANGAM	0x00 6012	1	Global acceptance mask
CANMC	0x00 6014	1	Master control
CANBTC	0x00 6016	1	Bit-timing configuration
CANES	0x00 6018	1	Error and status
CANTEC	0x00 601A	1	Transmit error counter
CANREC	0x00 601C	1	Receive error counter
CANGIF0	0x00 601E	1	Global interrupt flag 0
CANGIM	0x00 6020	1	Global interrupt mask
CANGIF1	0x00 6022	1	Global interrupt flag 1
CANMIM	0x00 6024	1	Mailbox interrupt mask
CANMIL	0x00 6026	1	Mailbox interrupt level
CANOPC	0x00 6028	1	Overwrite protection control
CANTIOC	0x00 602A	1	TX I/O control
CANRIOC	0x00 602C	1	RX I/O control
CANTSC	0x00 602E	1	Time stamp counter (Reserved in SCC mode)
CANTOC	0x00 6030	1	Time-out control (Reserved in SCC mode)
CANTOS	0x00 6032	1	Time-out status (Reserved in SCC mode)

† These registers are mapped to Peripheral Frame 1.

4.5 Multichannel Buffered Serial Port (McBSP) Module

The McBSP module has the following features:

- Compatible to McBSP in TMS320C54x™ /TMS320C55x™ DSP devices, except the DMA features
- Full-duplex communication
- Double-buffered data registers which allow a continuous data stream
- Independent framing and clocking for receive and transmit
- External shift clock generation or an internal programmable frequency shift clock
- A wide selection of data sizes including 8-, 12-, 16-, 20-, 24-, or 32-bits
- 8-bit data transfers with LSB or MSB first
- Programmable polarity for both frame synchronization and data clocks
- HIghly programmable internal clock and frame generation
- Support A-bis mode
- Direct interface to industry-standard CODECs, Analog Interface Chips (AICs), and other serially connected A/D and D/A devices
- Works with SPI-compatible devices
- Two 16 x 16-level FIFO for Transmit channel
- Two 16 x 16-level FIFO for Receive channel

The following application interfaces can be supported on the McBSP:

- T1/E1 framers
- MVIP switching-compatible and ST-BUS-compliant devices including:
  - MVIP framers
  - H.100 framers
  - SCSA framers
  - IOM-2 compliant devices
  - AC97-compliant devices (the necessary multiphase frame synchronization capability is provided.)
  - IIS-compliant devices
- McBSP clock rate = CLKG = $\frac{\text{CLKSRG}}{(1 + \text{CLKGDIV})}$, where CLKSRG source could be LSPCLK, CLKX, or CLKR.†

TMS320C54x and TMS320C55x are trademarks of Texas Instruments.

† Serial port performance is limited by I/O buffer switching speed. Internal prescalers must be adjusted such that the peripheral speed is less than the I/O buffer speed limit—20-MHz maximum.

Figure 4−9 shows the block diagram of the McBSP module with FIFO, interfaced to the F281x and C281x version of Peripheral Frame 2.

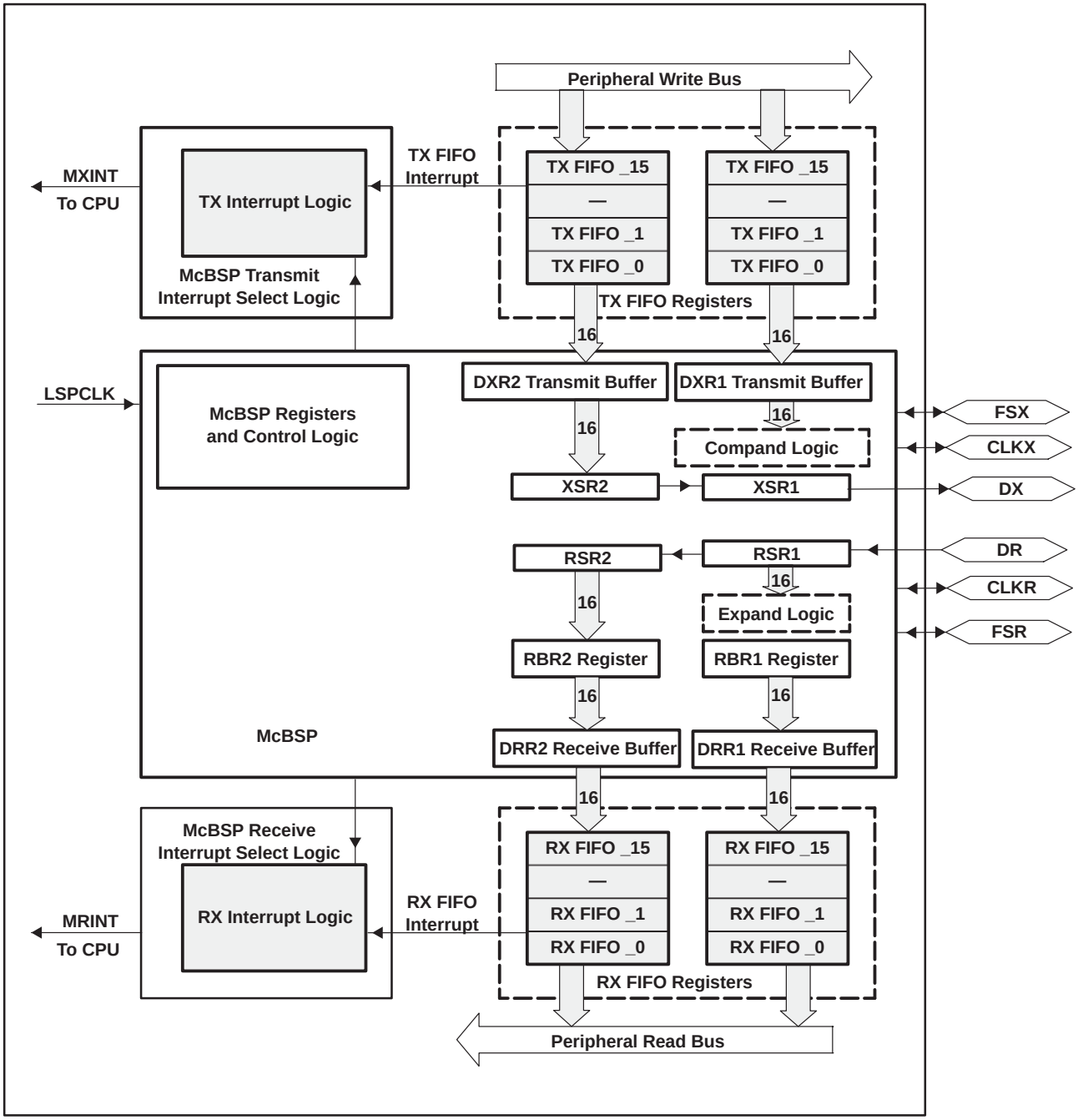


Figure 4−9. McBSP Module With FIFO

Table 4−7 provides a summary of the McBSP registers.

Table 4−7. McBSP Register Summary

NAME	ADDRESS 0x00 78xxh	TYPE (R/W)	RESET VALUE (HEX)	DESCRIPTION
DATA REGISTERS, RECEIVE, TRANSMIT†				
–	–	–	0x0000	McBSP Receive Buffer Register
–	–	–	0x0000	McBSP Receive Shift Register
–	–	–	0x0000	McBSP Transmit Shift Register
DRR2	00	R	0x0000	McBSP Data Receive Register 2 − Read First if the word size is greater than 16 bits, else ignore DRR2
DRR1	01	R	0x0000	McBSP Data Receive Register 1 − Read Second if the word size is greater than 16 bits, else read DRR1 only
DXR2	02	W	0x0000	McBSP Data Transmit Register 2 − Write First if the word size is greater than 16 bits, else ignore DXR2
DXR1	03	W	0x0000	McBSP Data Transmit Register 1 − Write Second if the word size is greater than 16 bits, else write to DXR1 only
McBSP CONTROL REGISTERS				
SPCR2	04	R/W	0x0000	McBSP Serial Port Control Register 2
SPCR1	05	R/W	0x0000	McBSP Serial Port Control Register 1
RCR2	06	R/W	0x0000	McBSP Receive Control Register 2
RCR1	07	R/W	0x0000	McBSP Receive Control Register 1
XCR2	08	R/W	0x0000	McBSP Transmit Control Register 2
XCR1	09	R/W	0x0000	McBSP Transmit Control Register 1
SRGR2	0A	R/W	0x0000	McBSP Sample Rate Generator Register 2
SRGR1	0B	R/W	0x0000	McBSP Sample Rate Generator Register 1
MULTICHANNEL CONTROL REGISTERS				
MCR2	0C	R/W	0x0000	McBSP Multichannel Register 2
MCR1	0D	R/W	0x0000	McBSP Multichannel Register 1
RCERA	0E	R/W	0x0000	McBSP Receive Channel Enable Register Partition A
RCERB	0F	R/W	0x0000	McBSP Receive Channel Enable Register Partition B
XCERA	10	R/W	0x0000	McBSP Transmit Channel Enable Register Partition A
XCERB	11	R/W	0x0000	McBSP Transmit Channel Enable Register Partition B
PCR	12	R/W	0x0000	McBSP Pin Control Register
RCERC	13	R/W	0x0000	McBSP Receive Channel Enable Register Partition C
RCERD	14	R/W	0x0000	McBSP Receive Channel Enable Register Partition D
XCERC	15	R/W	0x0000	McBSP Transmit Channel Enable Register Partition C
XCERD	16	R/W	0x0000	McBSP Transmit Channel Enable Register Partition D

† DRR2/DRR1 and DXR2/DXR1 share the same addresses of receive and transmit FIFO registers in FIFO mode.
‡ FIFO pointers advancing is based on order of access to DRR2/DRR1 and DXR2/DXR1 registers.

Table 4−7. McBSP Register Summary (Continued)

NAME	ADDRESS 0x00 78xxh	TYPE (R/W)	RESET VALUE (HEX)	DESCRIPTION
MULTICHANNEL CONTROL REGISTERS (CONTINUED)				
RCERE	17	R/W	0x0000	McBSP Receive Channel Enable Register Partition E
RCERF	18	R/W	0x0000	McBSP Receive Channel Enable Register Partition F
XCERE	19	R/W	0x0000	McBSP Transmit Channel Enable Register Partition E
XCERF	1A	R/W	0x0000	McBSP Transmit Channel Enable Register Partition F
RCERG	1B	R/W	0x0000	McBSP Receive Channel Enable Register Partition G
RCERH	1C	R/W	0x0000	McBSP Receive Channel Enable Register Partition H
XCERG	1D	R/W	0x0000	McBSP Transmit Channel Enable Register Partition G
XCERH	1E	R/W	0x0000	McBSP Transmit Channel Enable Register Partition H
FIFO MODE REGISTERS (applicable only in FIFO mode)				
FIFO Data Registers‡				
DRR2	00	R	0x0000	McBSP Data Receive Register 2 − Top of receive FIFO − Read First FIFO pointers will not advance
DRR1	01	R	0x0000	McBSP Data Receive Register 1 − Top of receive FIFO − Read Second for FIFO pointers to advance
DXR2	02	W	0x0000	McBSP Data Transmit Register 2 − Top of transmit FIFO − Write First FIFO pointers will not advance
DXR1	03	W	0x0000	McBSP Data Transmit Register 1 − Top of transmit FIFO − Write Second for FIFO pointers to advance
FIFO Control Registers				
MFFTX	20	R/W	0xA000	McBSP Transmit FIFO Register
MFFRX	21	R/W	0x201F	McBSP Receive FIFO Register
MFFCT	22	R/W	0x0000	McBSP FIFO Control Register
MFFINT	23	R/W	0x0000	McBSP FIFO Interrupt Register
MFFST	24	R/W	0x0000	McBSP FIFO Status Register

† DRR2/DRR1 and DXR2/DXR1 share the same addresses of receive and transmit FIFO registers in FIFO mode.
‡ FIFO pointers advancing is based on order of access to DRR2/DRR1 and DXR2/DXR1 registers.

4.6 Serial Communications Interface (SCI) Module

The F281x and C281x devices include two serial communications interface (SCI) modules. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format. The SCI receiver and transmitter are double-buffered, and each has its own separate enable and interrupt bits. Both can be operated independently or simultaneously in the full-duplex mode. To ensure data integrity, the SCI checks received data for break detection, parity, overrun, and framing errors. The bit rate is programmable to over 65000 different speeds through a 16-bit baud-select register.

Features of each SCI module include:

- Two external pins:
  - SCITXD: SCI transmit-output pin
  - SCIRXD: SCI receive-input pin

  NOTE: Both pins can be used as GPIO if not used for SCI.
- Baud rate programmable to 64K different rates†
  - Baud rate $= \frac{LSPCLK}{(BRR + 1) * 8}$, when $BRR \neq 0$

    $= \frac{LSPCLK}{16}$, when $BRR = 0$
- Data-word format
  - One start bit
  - Data-word length programmable from one to eight bits
  - Optional even/odd/no parity bit
  - One or two stop bits
- Four error-detection flags: parity, overrun, framing, and break detection
- Two wake-up multiprocessor modes: idle-line and address bit
- Half- or full-duplex operation
- Double-buffered receive and transmit functions
- Transmitter and receiver operations can be accomplished through interrupt-driven or polled algorithms with status flags.
  - Transmitter: TXRDY flag (transmitter-buffer register is ready to receive another character) and TX EMPTY flag (transmitter-shift register is empty)
  - Receiver: RXRDY flag (receiver-buffer register is ready to receive another character), BRKDT flag (break condition occurred), and RX ERROR flag (monitoring four interrupt conditions)
- Separate enable bits for transmitter and receiver interrupts (except BRKDT)
- Max bit rate $= \frac{150\ MHz}{2 \times 8} = 9.375 \times 10^6$ b/s

† Serial port performance is limited by I/O buffer switching speed. Internal prescalers must be adjusted such that the peripheral speed is less than the I/O buffer speed limit—20 MHz maximum.

- NRZ (non-return-to-zero) format
- Ten SCI module control registers located in the control register frame beginning at address 7050h

  NOTE: All registers in this module are 8-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7−0), and the upper byte (15−8) is read as zeros. Writing to the upper byte has no effect.

Enhanced features:

- Auto baud-detect hardware logic
- 16-level transmit/receive FIFO

The SCI port operation is configured and controlled by the registers listed in Table 4−8 and Table 4−9.

Table 4−8. SCI-A Registers†

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
SCICCRA	0x00 7050	1	SCI-A Communications Control Register
SCICTL1A	0x00 7051	1	SCI-A Control Register 1
SCIHBAUDA	0x00 7052	1	SCI-A Baud Register, High Bits
SCILBAUDA	0x00 7053	1	SCI-A Baud Register, Low Bits
SCICTL2A	0x00 7054	1	SCI-A Control Register 2
SCIRXSTA	0x00 7055	1	SCI-A Receive Status Register
SCIRXEMUA	0x00 7056	1	SCI-A Receive Emulation Data Buffer Register
SCIRXBUFA	0x00 7057	1	SCI-A Receive Data Buffer Register
SCITXBUFA	0x00 7059	1	SCI-A Transmit Data Buffer Register
SCIFFTXA	0x00 705A	1	SCI-A FIFO Transmit Register
SCIFFRXA	0x00 705B	1	SCI-A FIFO Receive Register
SCIFFCTA	0x00 705C	1	SCI-A FIFO Control Register
SCIPRIA	0x00 705F	1	SCI-A Priority Control Register

† Shaded registers are new registers for the FIFO mode.

Table 4−9. SCI-B Registers†‡

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
SCICCRB	0x00 7750	1	SCI-B Communications Control Register
SCICTL1B	0x00 7751	1	SCI-B Control Register 1
SCIHBAUDB	0x00 7752	1	SCI-B Baud Register, High Bits
SCILBAUDB	0x00 7753	1	SCI-B Baud Register, Low Bits
SCICTL2B	0x00 7754	1	SCI-B Control Register 2
SCIRXSTB	0x00 7755	1	SCI-B Receive Status Register
SCIRXEMUB	0x00 7756	1	SCI-B Receive Emulation Data Buffer Register
SCIRXBUFB	0x00 7757	1	SCI-B Receive Data Buffer Register
SCITXBUFB	0x00 7759	1	SCI-B Transmit Data Buffer Register
SCIFFTXB	0x00 775A	1	SCI-B FIFO Transmit Register
SCIFFRXB	0x00 775B	1	SCI-B FIFO Receive Register
SCIFFCTB	0x00 775C	1	SCI-B FIFO Control Register
SCIPRIB	0x00 775F	1	SCI-B Priority Control Register

† Shaded registers are new registers for the FIFO mode.
‡ Registers in this table are mapped to peripheral bus 16 space. This space only allows 16-bit accesses. 32-bit accesses produce undefined results.

Figure 4−10 shows the SCI module block diagram.

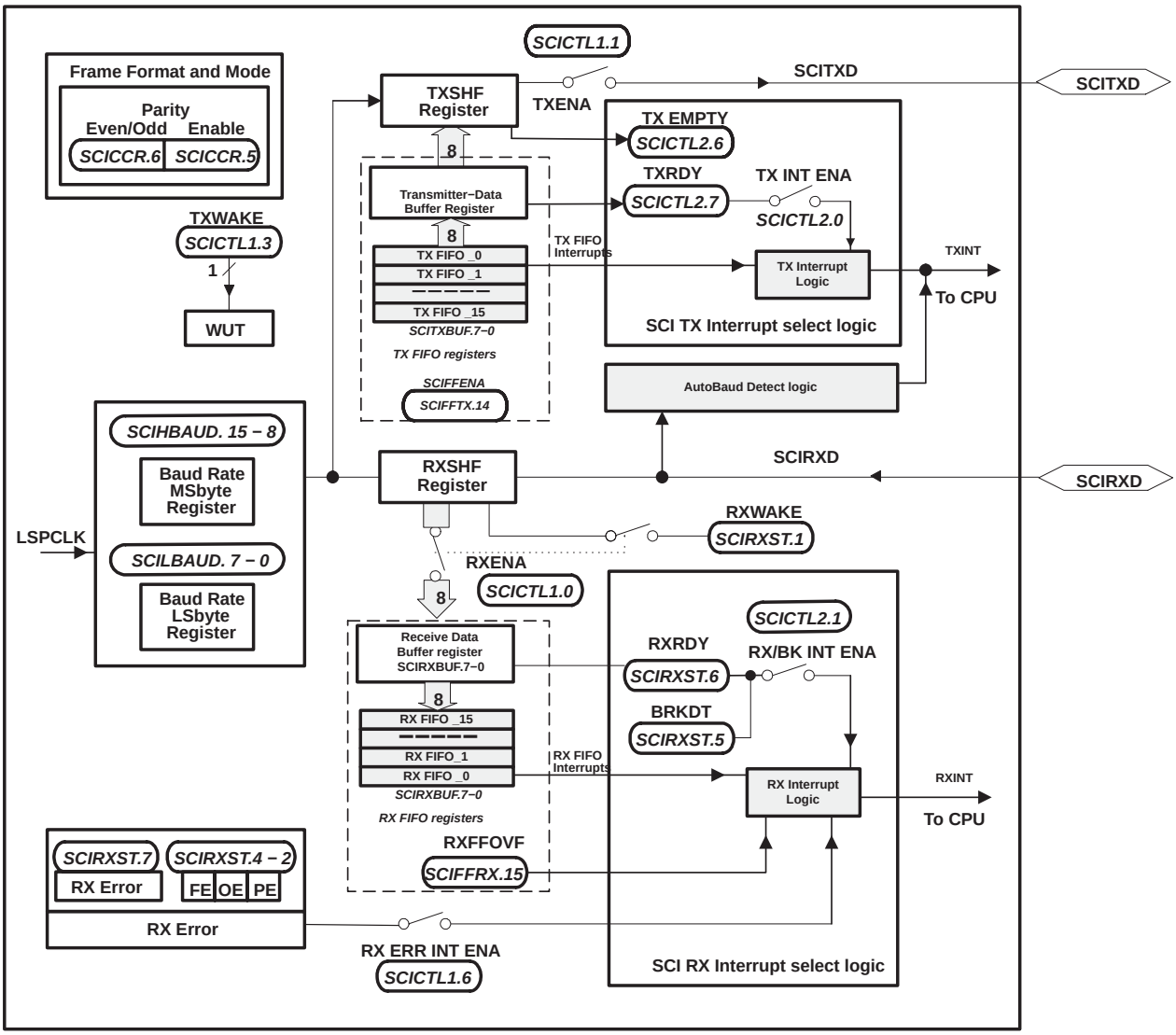


Figure 4−10. Serial Communications Interface (SCI) Module Block Diagram

4.7 Serial Peripheral Interface (SPI) Module

The F281x and C281x devices include the four-pin serial peripheral interface (SPI) module. The SPI is a high-speed, synchronous serial I/O port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmable bit-transfer rate. Normally, the SPI is used for communications between the DSP controller and external peripherals or another processor. Typical applications include external I/O or peripheral expansion through devices such as shift registers, display drivers, and ADCs. Multidevice communications are supported by the master/slave operation of the SPI.

The SPI module features include:

- Four external pins:
  - SPISOMI: SPI slave-output/master-input pin
  - SPISIMO: SPI slave-input/master-output pin
  - $\overline{\text{SPISTE}}$: SPI slave transmit-enable pin
  - SPICLK: SPI serial-clock pin

NOTE: All four pins can be used as GPIO, if the SPI module is not used.

- Two operational modes: master and slave
- Baud rate: 125 different programmable rates
  - Baud rate $= \frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$, when BRR $\neq 0$
    $= \frac{\text{LSPCLK}}{4}$, when BRR = 0, 1, 2, 3

    Serial port performance is limited by I/O buffer switching speed. Internal prescalers must be adjusted such that the peripheral speed is less than the I/O buffer speed limit—20 MHz maximum.
- Data word length: one to sixteen data bits
- Four clocking schemes (controlled by clock polarity and clock phase bits) include:
  - Falling edge without phase delay: SPICLK active-high. SPI transmits data on the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
  - Falling edge with phase delay: SPICLK active-high. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge without phase delay: SPICLK inactive-low. SPI transmits data on the rising edge of the SPICLK signal and receives data on the falling edge of the SPICLK signal.
  - Rising edge with phase delay: SPICLK inactive-low. SPI transmits data one half-cycle ahead of the falling edge of the SPICLK signal and receives data on the rising edge of the SPICLK signal.
- Simultaneous receive and transmit operation (transmit function can be disabled in software)
- Transmitter and receiver operations are accomplished through either interrupt-driven or polled algorithms.
- Nine SPI module control registers: Located in control register frame beginning at address 7040h.

  NOTE: All registers in this module are 16-bit registers that are connected to Peripheral Frame 2. When a register is accessed, the register data is in the lower byte (7−0), and the upper byte (15−8) is read as zeros. Writing to the upper byte has no effect.

Enhanced feature:

- 16-level transmit/receive FIFO
- Delayed transmit control

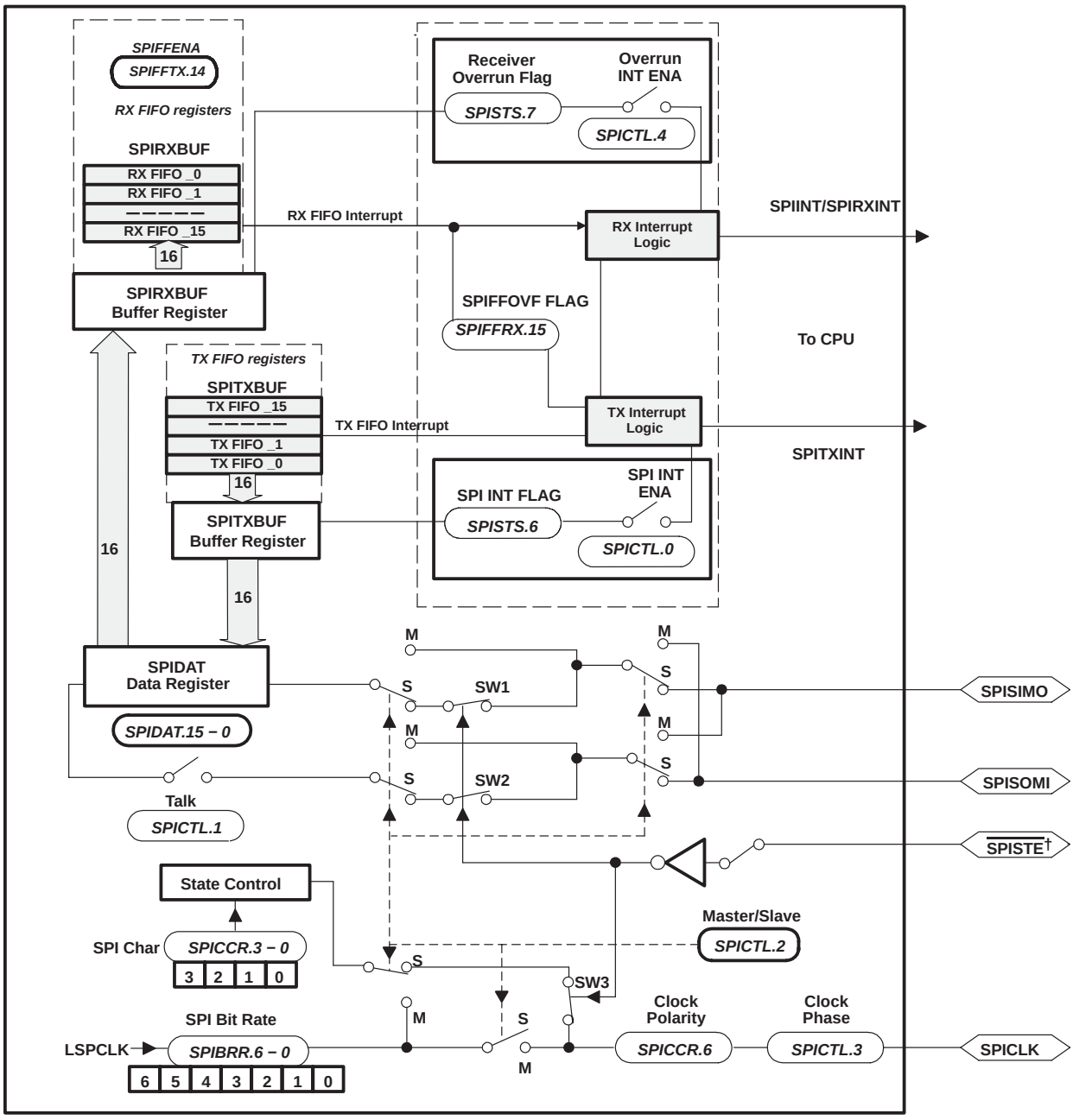
The SPI port operation is configured and controlled by the registers listed in Table 4−10.

Table 4−10. SPI Registers

NAME	ADDRESS	SIZE (x16)	DESCRIPTION
SPICCR	0x00 7040	1	SPI Configuration Control Register
SPICTL	0x00 7041	1	SPI Operation Control Register
SPISTS	0x00 7042	1	SPI Status Register
SPIBRR	0x00 7044	1	SPI Baud Rate Register
SPIRXEMU	0x00 7046	1	SPI Receive Emulation Buffer Register
SPIRXBUF	0x00 7047	1	SPI Serial Input Buffer Register
SPITXBUF	0x00 7048	1	SPI Serial Output Buffer Register
SPIDAT	0x00 7049	1	SPI Serial Data Register
SPIFFTX	0x00 704A	1	SPI FIFO Transmit Register
SPIFFRX	0x00 704B	1	SPI FIFO Receive Register
SPIFFCT	0x00 704C	1	SPI FIFO Control Register
SPIPRI	0x00 704F	1	SPI Priority Control Register

NOTE: The above registers are mapped to Peripheral Frame 2. This space only allows 16-bit accesses. 32-bit accesses produce undefined results.

Figure 4−11 is a block diagram of the SPI in slave mode.



† $\overline{\text{SPISTE}}$ is driven low by the master for a slave device.

Figure 4−11. Serial Peripheral Interface Module Block Diagram (Slave Mode)

4.8 GPIO MUX

The GPIO Mux registers, are used to select the operation of shared pins on the F281x and C281x devices. The pins can be individually selected to operate as “Digital I/O” or connected to “Peripheral I/O” signals (via the GPxMUX registers). If selected for “Digital I/O” mode, registers are provided to configure the pin direction (via the GPxDIR registers) and to qualify the input signal to remove unwanted noise (via the GPxQUAL) registers). Table 4−11 lists the GPIO Mux Registers.

Table 4−11. GPIO Mux Registers†‡§

NAME	ADDRESS	SIZE (x16)	REGISTER DESCRIPTION
GPAMUX	0x00 70C0	1	GPIO A Mux Control Register
GPADIR	0x00 70C1	1	GPIO A Direction Control Register
GPAQUAL	0x00 70C2	1	GPIO A Input Qualification Control Register
reserved	0x00 70C3	1	
GPBMUX	0x00 70C4	1	GPIO B Mux Control Register
GPBDIR	0x00 70C5	1	GPIO B Direction Control Register
GPBQUAL	0x00 70C6	1	GPIO B Input Qualification Control Register
reserved	0x00 70C7	1	
reserved	0x00 70C8	1	
reserved	0x00 70C9	1	
reserved	0x00 70CA	1	
reserved	0x00 70CB	1	
GPDMUX	0x00 70CC	1	GPIO D Mux Control Register
GPDDIR	0x00 70CD	1	GPIO D Direction Control Register
GPDQUAL	0x00 70CE	1	GPIO D Input Qualification Control Register
reserved	0x00 70CF	1	
GPEMUX	0x00 70D0	1	GPIO E Mux Control Register
GPEDIR	0x00 70D1	1	GPIO E Direction Control Register
GPEQUAL	0x00 70D2	1	GPIO E Input Qualification Control Register
reserved	0x00 70D3	1	
GPFMUX	0x00 70D4	1	GPIO F Mux Control Register
GPFDIR	0x00 70D5	1	GPIO F Direction Control Register
reserved	0x00 70D6	1	
reserved	0x00 70D7	1	
GPGMUX	0x00 70D8	1	GPIO G Mux Control Register
GPGDIR	0x00 70D9	1	GPIO G Direction Control Register
reserved	0x00 70DA	1	
reserved	0x00 70DB	1	
reserved	0x00 70DC 0x00 70DF	4	

† Reserved locations will return undefined values and writes will be ignored.
‡ Not all inputs will support input signal qualification.
§ These registers are EALLOW protected. This prevents spurious writes from overwriting the contents and corrupting the system.

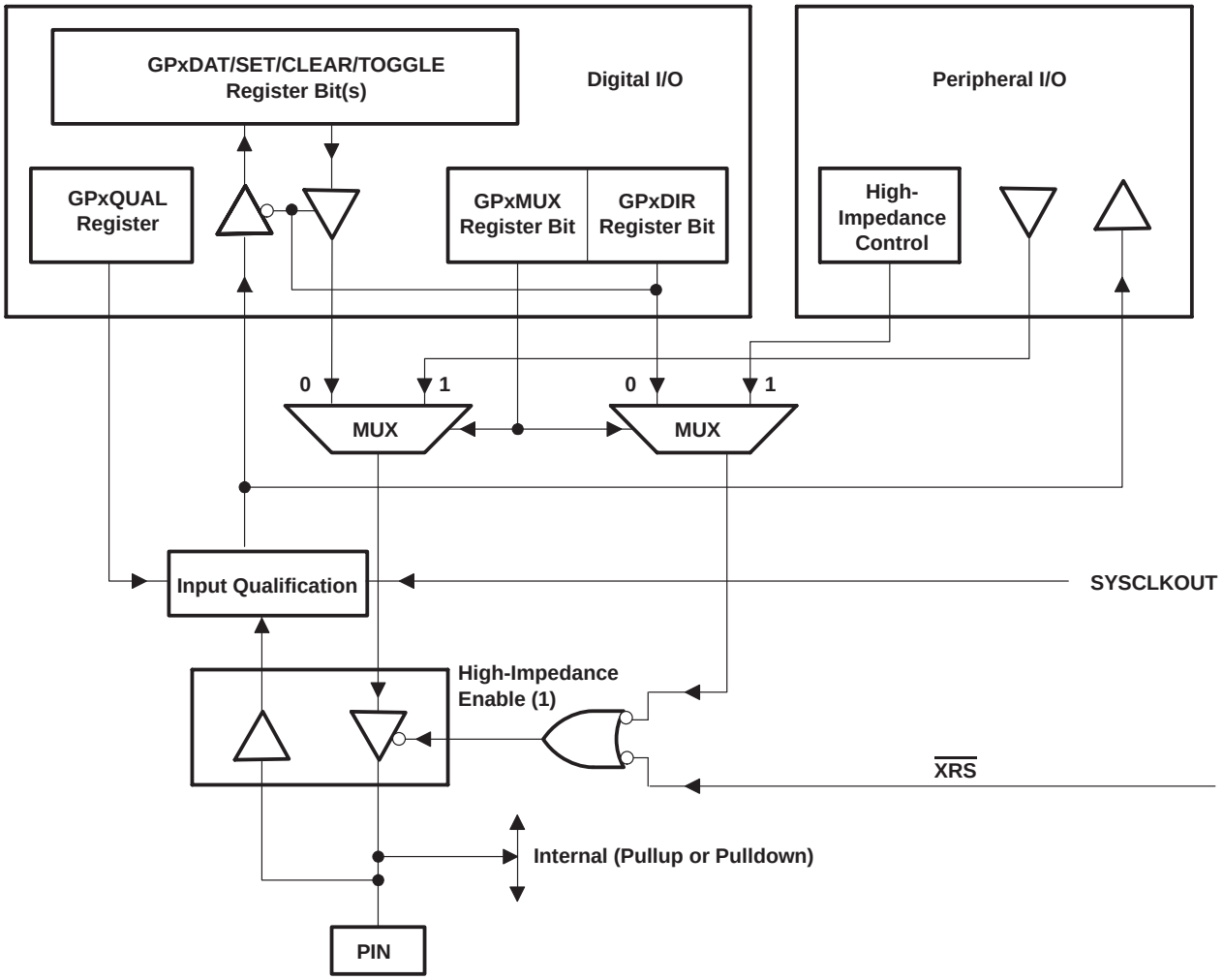
If configured for ”Digital I/O” mode, additional registers are provided for setting individual I/O signals (via the GPxSET registers), for clearing individual I/O signals (via the GPxCLEAR registers), for toggling individual I/O signals (via the GPxTOGGLE registers), or for reading/writing to the individual I/O signals (via the GPxDAT registers). Table 4−12 lists the GPIO Data Registers. For more information, see the *TMS320x281x System Control and Interrupts Reference Guide* (literature number SPRU078).

Table 4−12. GPIO Data Registers†‡

NAME	ADDRESS	SIZE (x16)	REGISTER DESCRIPTION
GPADAT	0x00 70E0	1	GPIO A Data Register
GPASET	0x00 70E1	1	GPIO A Set Register
GPACLEAR	0x00 70E2	1	GPIO A Clear Register
GPATOGGLE	0x00 70E3	1	GPIO A Toggle Register
GPBDAT	0x00 70E4	1	GPIO B Data Register
GPBSET	0x00 70E5	1	GPIO B Set Register
GPBCLEAR	0x00 70E6	1	GPIO B Clear Register
GPBTOGGLE	0x00 70E7	1	GPIO B Toggle Register
reserved	0x00 70E8	1	
reserved	0x00 70E9	1	
reserved	0x00 70EA	1	
reserved	0x00 70EB	1	
GPDDAT	0x00 70EC	1	GPIO D Data Register
GPDSET	0x00 70ED	1	GPIO D Set Register
GPDCLEAR	0x00 70EE	1	GPIO D Clear Register
GPDTOGGLE	0x00 70EF	1	GPIO D Toggle Register
GPEDAT	0x00 70F0	1	GPIO E Data Register
GPESET	0x00 70F1	1	GPIO E Set Register
GPECLEAR	0x00 70F2	1	GPIO E Clear Register
GPETOGGLE	0x00 70F3	1	GPIO E Toggle Register
GPFDAT	0x00 70F4	1	GPIO F Data Register
GPFSET	0x00 70F5	1	GPIO F Set Register
GPFCLEAR	0x00 70F6	1	GPIO F Clear Register
GPFTOGGLE	0x00 70F7	1	GPIO F Toggle Register
GPGDAT	0x00 70F8	1	GPIO G Data Register
GPGSET	0x00 70F9	1	GPIO G Set Register
GPGCLEAR	0x00 70FA	1	GPIO G Clear Register
GPGTOGGLE	0x00 70FB	1	GPIO G Toggle Register
reserved	0x00 70FC 0x00 70FF	4	

† Reserved locations will return undefined values and writes will be ignored.
‡ These registers are NOT EALLOW protected. The above registers will typically be accessed regularly by the user.

Figure 4−12 shows how the various register bits select the various modes of operation for GPIO function.



NOTES: A. In the GPIO mode, when the GPIO pin is configured for output operation, reading the GPxDAT data register only gives the value written, not the value at the pin. In the peripheral mode, the state of the pin can be read through the GPxDAT register, provided the corresponding direction bit is zero (input mode).

B. Some selected input signals are qualified by the SYSCLKOUT. The GPxQUAL register specifies the qualification sampling period. The sampling window is 6 samples wide and the output is only changed when all samples are the same (all 0's or all 1's). This feature removes unwanted spikes from the input signal.

Figure 4−12. GPIO/Peripheral Pin Multiplexing

NOTE:

The input function of the GPIO pin and the input path to the peripheral are always enabled. It is the output function of the GPIO pin that is multiplexed with the output path of the primary (peripheral) function. Since the output buffer of a pin connects back to the input buffer, any GPIO signal present at the pin will be propagated to the peripheral module as well. Therefore, when a pin is configured for GPIO operation, the corresponding peripheral functionality (and interrupt-generating capability) must be disabled. Otherwise, interrupts may be inadvertently triggered. This is especially critical when the $\overline{\text{PDPINTA}}$ and $\overline{\text{PDPINTB}}$ pins are used as GPIO pins, since a value of zero for GPDDAT.0 or GPDDAT.5 ($\overline{\text{PDPINTx}}$) will put PWM pins in a high-impedance state. The $\overline{\text{CxTRIP}}$ and $\overline{\text{TxCTRIP}}$ pins will also put the corresponding PWM pins in high impedance, if they are driven low (as GPIO pins) *and* bit EXTCONx.0 = 1.

5 Development Support

Texas Instruments (TI) offers an extensive line of development tools for the C28x™ generation of DSPs, including tools to evaluate the performance of the processors, generate code, develop algorithm implementations, and fully integrate and debug software and hardware modules.

The following products support development of F281x- and C281x-based applications:

Software Development Tools

- Code Composer Studio™ Integrated Development Environment (IDE)
  - C/C++ Compiler
  - Code generation tools
  - Assembler/Linker
  - Cycle Accurate Simulator
- Application algorithms
- Sample applications code

Hardware Development Tools

- 2812 eZdsp
- JTAG-based emulators − SPI515, XDS510PP, XDS510PP Plus, XDS510 USB
- Universal 5-V dc power supply
- Documentation and cables

5.1 Device and Development Support Tool Nomenclature

To designate the stages in the product development cycle, TI assigns prefixes to the part numbers of all [TMS320] DSP devices and support tools. Each [TMS320] DSP commercial family member has one of three prefixes: TMX, TMP, or TMS (e.g., TMS320F2812GHH). Texas Instruments recommends two of three possible prefix designators for its support tools: TMDX and TMDS. These prefixes represent evolutionary stages of product development from engineering prototypes (TMX/TMDX) through fully qualified production devices/tools (TMS/TMDS).

TMX Experimental device that is not necessarily representative of the final device's electrical specifications

TMP Final silicon die that conforms to the device's electrical specifications but has not completed quality and reliability verification

TMS Fully qualified production device

Support tool development evolutionary flow:

TMDX Development-support product that has not yet completed Texas Instruments internal qualification testing.

TMDS Fully qualified development-support product

TMX and TMP devices and TMDX development-support tools are shipped against the following disclaimer:

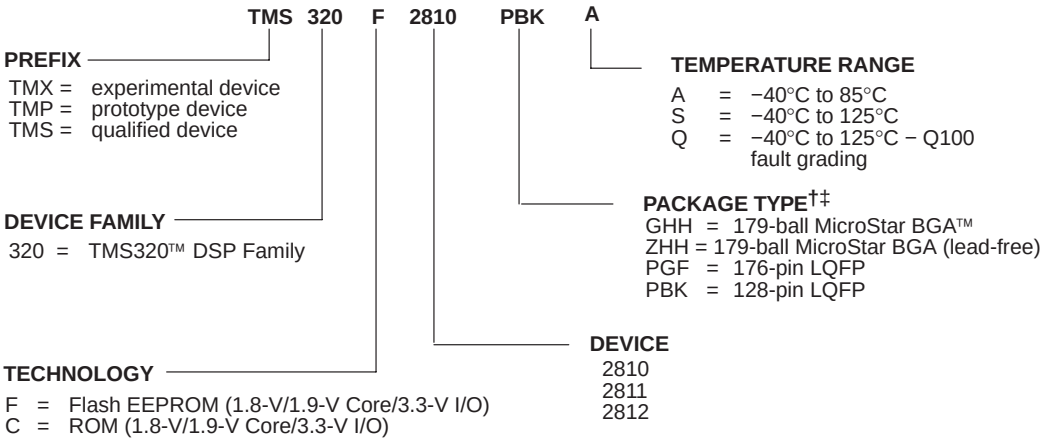
"Developmental product is intended for internal evaluation purposes."

TMS devices and TMDS development-support tools have been characterized fully, and the quality and reliability of the device have been demonstrated fully. TI's standard warranty applies.

Predictions show that prototype devices (TMX or TMP) have a greater failure rate than the standard production devices. Texas Instruments recommends that these devices not be used in any production system because their expected end-use failure rate still is undefined. Only qualified production devices are to be used.

TMS320 is a trademark of Texas Instruments.

TI device nomenclature also includes a suffix with the device family name. This suffix indicates the package type (for example, PBK) and temperature range (for example, A). Figure 5−1 provides a legend for reading the complete device name for any TMS320x28x family member.



† BGA = Ball Grid Array
LQFP = Low-Profile Quad Flatpack
‡ LQFP package not yet available lead (Pb)-free. For estimated conversion dates, go to www.ti.com/leadfree

Figure 5−1. TMS320x28x Device Nomenclature

5.2 Documentation Support

Extensive documentation supports all of the TMS320™ DSP family generations of devices from product announcement through applications development. The types of documentation available include: data sheets and data manuals, with design specifications; and hardware and software applications. Useful reference documentation includes:

TMS320C28x DSP CPU and Instruction Set Reference Guide (literature number SPRU430) describes the central processing unit (CPU) and the assembly language instructions of the TMS320C28x™ fixed-point digital signal processors (DSPs). It also describes emulation features available on these DSPs.

TMS320x281x Analog-to-Digital Converter (ADC) Reference Guide (literature number SPRU060) describes the ADC module. The module is a 12-bit pipelined ADC. The analog circuits of this converter, referred to as the core in this document, include the front-end analog multiplexers (MUXs), sample-and-hold (S/H) circuits, the conversion core, voltage regulators, and other analog supporting circuits. Digital circuits, referred to as the wrapper in this document, include programmable conversion sequencer, result registers, interface to analog circuits, interface to device peripheral bus, and interface to other on-chip modules.

TMS320x281x Boot ROM Reference Guide (literature number SPRU095) describes the purpose and features of the bootloader (factory-programmed boot-loading software). It also describes other contents of the device on-chip boot ROM and identifies where all of the information is located within that memory.

TMS320x281x Event Manager (EV) Reference Guide (literature number SPRU065) describes the EV modules that provide a broad range of functions and features that are particularly useful in motion control and motor control applications. The EV modules include general-purpose (GP) timers, full-compare/PWM units, capture units, and quadrature-encoder pulse (QEP) circuits.

TMS320x281x External Interface (XINTF) Reference Guide (literature number SPRU067) describes the external interface (XINTF) of the 281x digital signal processors (DSPs).

TMS320x281x Multi-channel Buffered Serial Ports (McBSPs) Reference Guide (literature number SPRU061) describes the McBSP) available on the 281x devices. The McBSPs allow direct interface between a DSP and other devices in a system.

TMS320x281x System Control and Interrupts Reference Guide (literature number SPRU078) describes the various interrupts and system control features of the 281x digital signal processors (DSPs).

TMS320x281x, 280x Enhanced Controller Area Network (eCAN) Reference Guide (literature number SPRU074) describes the eCAN that uses established protocol to communicate serially with other controllers in electrically noisy environments. With 32 fully configurable mailboxes and time-stamping feature, the eCAN module provides a versatile and robust serial communication interface. The eCAN module implemented in the C28x DSP is compatible with the CAN 2.0B standard (active).

TMS320x281x, 280x Peripheral Reference Guide (literature number SPRU566) describes the peripheral reference guides of the 28x digital signal processors (DSPs).

TMS320x281x, 280x Serial Communication Interface (SCI) Reference Guide (literature number SPRU051) describes the SCI that is a two-wire asynchronous serial port, commonly known as a UART. The SCI modules support digital communications between the CPU and other asynchronous peripherals that use the standard non-return-to-zero (NRZ) format.

TMS320x281x, 280x Serial Peripheral Interface (SPI) Reference Guide (literature number SPRU059) describes the SPI − a high-speed synchronous serial input/output (I/O) port that allows a serial bit stream of programmed length (one to sixteen bits) to be shifted into and out of the device at a programmed bit-transfer rate. The SPI is used for communications between the DSP controller and external peripherals or another controller.

3.3 V DSP for Digital Motor Control Application Report (literature number SPRA550). New generations of motor control digital signal processors (DSPs) lower their supply voltages from 5 V to 3.3 V to offer higher performance at lower cost. Replacing traditional 5-V digital control circuitry by 3.3-V designs introduce no additional system cost and no significant complication in interfacing with TTL and CMOS compatible components, as well as with mixed voltage ICs such as power transistor gate drivers. Just like 5-V based designs, good engineering practice should be exercised to minimize noise and EMI effects by proper component layout and PCB design when 3.3-V DSP, ADC, and digital circuitry are used in a mixed signal environment, with high and low voltage analog and switching signals, such as a motor control system. In addition, software techniques such as Random PWM method can be used by special features of the Texas Instruments (TI) TMS320x24xx DSP controllers to significantly reduce noise effects caused by EMI radiation.

This application report reviews designs of 3.3-V DSP versus 5-V DSP for low HP motor control applications. The application report first describes a scenario of a 3.3-V-only motor controller indicating that for most applications, no significant issue of interfacing between 3.3 V and 5 V exists. Cost-effective 3.3-V − 5-V interfacing techniques are then discussed for the situations where such interfacing is needed. On-chip 3.3-V ADC versus 5-V ADC is also discussed. Sensitivity and noise effects in 3.3-V and 5-V ADC conversions are addressed. Guidelines for component layout and printed circuit board (PCB) design that can reduce system's noise and EMI effects are summarized in the last section.

The TMS320C28x Instruction Set Simulator Technical Overview (literature number SPRU608) describes the simulator, available within the Code Composer Studio for TMS320C2000 IDE, that simulates the instruction set of the C28x core.

TMS320C28x DSP/BIOS Application Programming Interface (API) Reference Guide (literature number SPRU625) describes development using DSP/BIOS.

TMS320C28x Assembly Language Tools User's Guide (literature number SPRU513) describes the assembly language tools (assembler and other tools used to develop assembly language code), assembler directives, macros, common object file format, and symbolic debugging directives for the TMS320C28x™ device.

TMS320C28x Optimizing C Compiler User's Guide (literature number SPRU514) describes the TMS320C28x™ C/C++ compiler. This compiler accepts ANSI standard C/C++ source code and produces TMS320™ DSP assembly language source code for the TMS320C28x device.

Programming Examples for the TMS320F281x eCAN (literature number SPRA876) contains several programming examples to illustrate how the eCAN module is set up for different modes of operation. The objective is to help you come up to speed quickly in programming the eCAN. All programs have been extensively commented to aid easy understanding. The CANalyzer tool from Vector CANtech, Inc. was used to monitor and control the bus operation. All projects and CANalyzer configuration files are included in the attached SPRA876.zip file.

TMS320F2810, TMS320F2811, TMS320F2812 ADC Calibration (literature number SPRA989) describes a method for improving the absolute accuracy of the 12-bit analog-to-digital converter (ADC) found on the F2810/F2811/F2812 devices. Due to inherent gain and offset errors, the absolute accuracy of the ADC is impacted. The methods described in this application note can improve the absolute accuracy of the ADC to achieve levels better than 0.5%. This application note is accompanied by an example program (ADCcalibration.zip) that executes from RAM on the F2812 EzDSP.

A series of DSP textbooks is published by Prentice-Hall and John Wiley & Sons to support digital signal processing research and education. The TMS320™ DSP newsletter, *Details on Signal Processing*, is published quarterly and distributed to update TMS320™ DSP customers on product information.

Updated information on the TMS320™ DSP controllers can be found on the worldwide web at: **http://www.ti.com**.

To send comments regarding this TMS320F281x/TMS320C281x data manual (literature number SPRS174), use the *comments@books.sc.ti.com* email address, which is a repository for feedback. For questions and support, contact the Product Information Center listed at the **http://www.ti.com/sc/docs/pic/home.htm** site.

6 Electrical Specifications

This section provides the absolute maximum ratings and the recommended operating conditions for the TMS320F281x and TMS320C281x DSPs.

6.1 Absolute Maximum Ratings

Unless otherwise noted, the list of absolute maximum ratings are specified over operating temperature ranges. Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only, and functional operation of the device at these or any other conditions beyond those indicated under Section 6.2 is not implied. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability. All voltage values are with respect to V_{SS}.

Supply voltage range, V_{DDIO}, V_{DDA1}, V_{DDA2}, V_{DDAIO}, and $AV_{DDREFBG}$ . . . . . − 0.3 V to 4.6 V

Supply voltage range, V_{DD}, V_{DD1} . . . . . − 0.5 V to 2.5 V

V_{DD3VFL} range . . . . . − 0.3 V to 4.6 V

Input voltage range, V_{IN} . . . . . − 0.3 V to 4.6 V

Output voltage range, V_O . . . . . − 0.3 V to 4.6 V

Input clamp current, I_{IK} ($V_{IN} < 0$ or $V_{IN} > V_{DDIO}$)† . . . . . ± 20 mA

Output clamp current, I_{OK} ($V_O < 0$ or $V_O > V_{DDIO}$) . . . . . ± 20 mA

Operating ambient temperature ranges, T_A: A version (GHH, PGF, PBK)‡ . . . . . − 40°C to 85°C

T_A: S version (GHH, PGF, PBK)‡§ . . . . . − 40°C to 125°C

T_A: Q version (GHH, PGF, PBK)‡ . . . . . − 40°C to 125°C

Storage temperature range, T_{stg}† . . . . . − 65°C to 150°C

† Continuous clamp current per pin is± 2 mA

‡ Long-term high-temperature storage and/or extended use at maximum temperature conditions may result in a reduction of overall device life. For additional information, see *IC Package Thermal Metrics Application Report* (literature number SPRA953) and *Reliability Data for TMS320LF24x and TMS320F281x Devices Application Repor*t (literature number SPRA963).

§ Replaced by Q temperature option from silicon revision E onwards

6.2 Recommended Operating Conditions†

				MIN	NOM	MAX	UNIT
V_{DDIO}	Device supply voltage, I/O			3.14	3.3	3.47	V
V_{DD}, V_{DD1}	Device supply voltage, CPU		1.8 V (135 MHz)	1.71	1.8	1.89	V
			1.9 V (150 MHz)	1.81	1.9	2	
V_{SS}	Supply ground				0		V
V_{DDA1}, V_{DDA2}, $AV_{DDREFBG}$, V_{DDAIO}	ADC supply voltage			3.14	3.3	3.47	V
V_{DD3VFL}	Flash programming supply voltage			3.14	3.3	3.47	V
$f_{SYSCLKOUT}$	Device clock frequency (system clock)		V_{DD} = 1.9 V ± 5%	2		150	MHz
			V_{DD} = 1.8 V ± 5%	2		135	
V_{IH}	High-level input voltage		All inputs except XCLKIN	2		V_{DDIO}	V
			XCLKIN (@ 50 µA max)	$0.7V_{DD}$		V_{DD}	
V_{IL}	Low-level input voltage		All inputs except XCLKIN			0.8	V
			XCLKIN (@ 50 µA max)			$0.3V_{DD}$	
I_{OH}	High-level output source current, V_{OH} = 2.4 V		All I/Os except Group 2			− 4	mA
			Group 2‡			− 8	
I_{OL}	Low-level output sink current, V_{OL} = V_{OL} MAX		All I/Os except Group 2			4	mA
			Group 2‡			8	
T_A	Ambient temperature	A version		− 40		85	°C
		S version§		− 40		125	
		Q version		− 40		125	°C

† See Section 6.8 for power sequencing of V_{DDIO}, V_{DDAIO}, V_{DD}, V_{DDA1}/V_{DDA2}/$AV_{DDREFBG}$, and V_{DD3VFL}.
‡ Group 2 pins are as follows: XINTF pins, T1CTRIP_PDPINTA, TDO, XCLKOUT, XF, EMU0, and EMU1.
§ Replaced by Q temperature option from silicon revision E onwards

6.3 Electrical Characteristics Over Recommended Operating Conditions (Unless Otherwise Noted)

PARAMETER			TEST CONDITIONS		MIN	TYP	MAX	UNIT
V_{OH}	High-level output voltage		$I_{OH} = I_{OH}MAX$		2.4			V
			I_{OH} = 50 µA		V_{DDIO} − 0.2			
V_{OL}	Low-level output voltage		$I_{OL} = I_{OL}MAX$				0.4	V
I_{IL}†	Input current (low level)	With pullup	V_{DDIO} = 3.3 V, V_{IN} = 0 V		−80	−140	−190	µA
		With pulldown	V_{DDIO} = 3.3 V, V_{IN} = 0 V				±2	
I_{IL}‡	Input current (low level)	With pullup	V_{DDIO} = 3.3 V, V_{IN} = 0 V	All I/Os§ (including $\overline{XRS}$) except EVB	−80	−140	−190	µA
				GPIOB/EVB	−13	−25	−35	
		With pulldown	V_{DDIO} = 3.3 V, V_{IN} = 0 V				±2	
I_{IH}	Input current (high level)	With pullup	V_{DDIO} = 3.3 V, $V_{IN} = V_{DD}$				±2	µA
		With pulldown¶	V_{DDIO} = 3.3 V, $V_{IN} = V_{DD}$		28	50	80	
I_{OZ}	Output current, high-impedance state (off-state)		$V_O = V_{DDIO}$ or 0 V				±2	µA
C_i	Input capacitance					2		pF
C_o	Output capacitance					3		pF

† Applicable to C281x devices
‡ Applicable to F281x devices
§ The following pins have no internal PU/PD: GPIOE0, GPIOE1, GPIOF0, GPIOF1, GPIOF2, GPIOF3, GPIOF12, GPIOG4, and GPIOG5.
¶ The following pins have an internal pulldown: XMP/$\overline{MC}$, TESTSEL, and $\overline{TRST}$.

6.4 Current Consumption by Power-Supply Pins Over Recommended Operating Conditions During Low-Power Modes at 150-MHz SYSCLKOUT (TMS320F281x)

MODE	TEST CONDITIONS	I_{DD}		I_{DDIO}		I_{DD3VFL}		I_{DDA}†	
		TYP	MAX‡	TYP	MAX‡	TYP	MAX‡	TYP	MAX‡
Operational	All peripheral clocks are enabled. All PWM pins are toggled at 100 kHz. Data is continuously transmitted out of the SCIA, SCIB, and CAN ports. The hardware multiplier is exercised. Code is running out of flash with 5 wait-states.	195 mA	230 mA	15 mA	30 mA	40 mA	45 mA	40 mA	50 mA
IDLE	− Flash is powered down − XCLKOUT is turned off − All peripheral clocks are on, except ADC	125 mA	150 mA	5 mA	10 mA	2 µA	4 µA	1 µA	20 µA
STANDBY	− Flash is powered down − Peripheral clocks are turned off − Pins without an internal PU/PD are tied high/low	5 mA	10 mA	5 µA	20 µA	2 µA	4 µA	1 µA	20 µA
HALT	− Flash is powered down − Peripheral clocks are turned off − Pins without an internal PU/PD are tied high/low − Input clock is disabled	70 µA		5 µA	20 µA	2 µA	4 µA	1 µA	20 µA

† I_{DDA} includes current into V_{DDA1}, V_{DDA2}, V_{DD1}, $AV_{DDREFBG}$, and V_{DDAIO} pins.
‡ MAX numbers are at 125°C, and MAX voltage (V_{DD} = 2.0 V; V_{DDIO}, V_{DD3VFL}, V_{DDA} = 3.6 V).

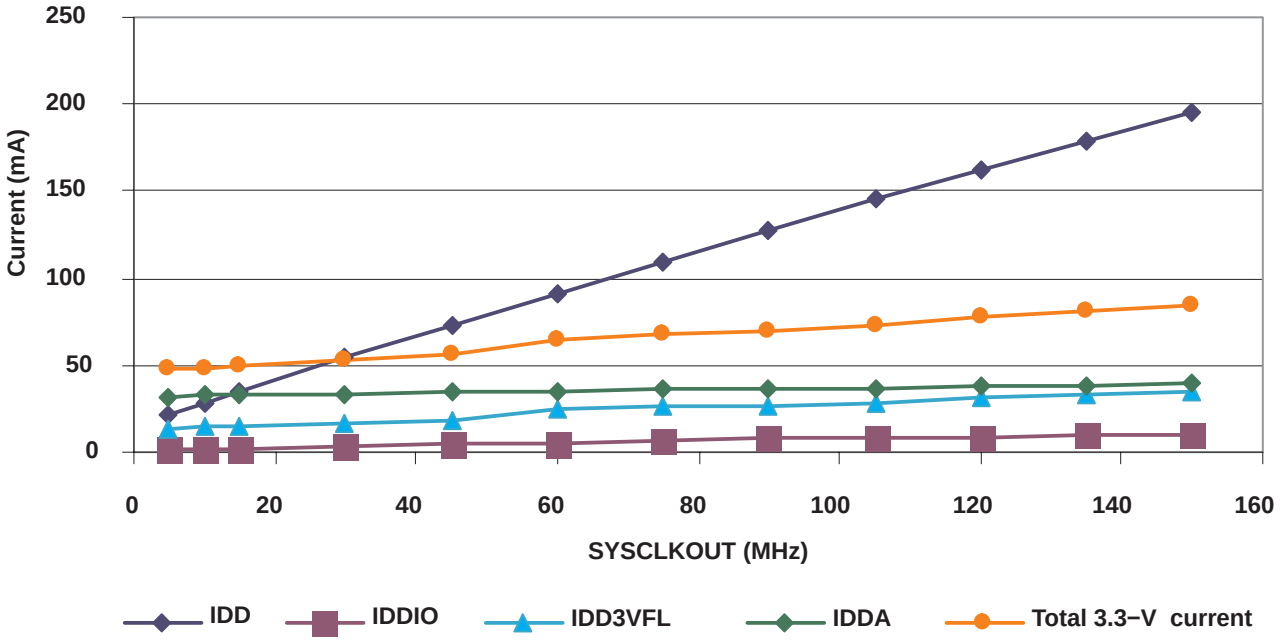
NOTE:
HALT and STANDBY modes cannot be used when the PLL is disabled.

6.5 Current Consumption by Power-Supply Pins Over Recommended Operating Conditions During Low-Power Modes at 150-MHz SYSCLKOUT (TMS320C281x)

MODE	TEST CONDITIONS	I_{DD}		I_{DDIO}		I_{DDA}†	
		TYP	MAX‡	TYP	MAX‡	TYP	MAX‡
Operational	All peripheral clocks are enabled. All PWM pins are toggled at 100 kHz. Data is continuously transmitted out of the SCIA, SCIB, and CAN ports. The hardware multiplier is exercised. Code is running out of ROM with 5 wait-states.	210 mA	260 mA	20 mA	30mA	40 mA	50 mA
IDLE	− XCLKOUT is turned off − All peripheral clocks are on, except ADC	140 mA	155 mA	20 mA	30 mA	5 µA	10 µA
STANDBY	− Peripheral clocks are turned off − Pins without an internal PU/PD are tied high/low	5 mA	10mA	1 mA	3 mA	5 µA	10 µA
HALT	− Peripheral clocks are turned off − Pins without an internal PU/PD are tied high/low − Input clock is disabled	70 µA		5 µA	10 µA	1 µA	

† I_{DDA} includes current into V_{DDA1}, V_{DDA2}, V_{DD1}, $AV_{DDREFBG}$, and V_{DDAIO} pins.

6.6 Current Consumption Graphs



NOTES: A. Test conditions are as defined in Table 6−5 for operational currents.
B. I_{DD} represents the total current drawn from the 1.8-V rail (V_{DD}). It includes a trivial amount of current (<1 mA) drawn by V_{DD1}.
C. IDDA represents the current drawn by VDDA1 and VDDA2 rails.
D. Total 3.3-V current is the sum of I_{DDIO}, I_{DD3VFL}, and I_{DDA}. It includes a trivial amount of current (<1 mA) drawn by VDDAIO.
.

Figure 6−1. F2812/F2811/F2810 Typical Current Consumption Over Frequency

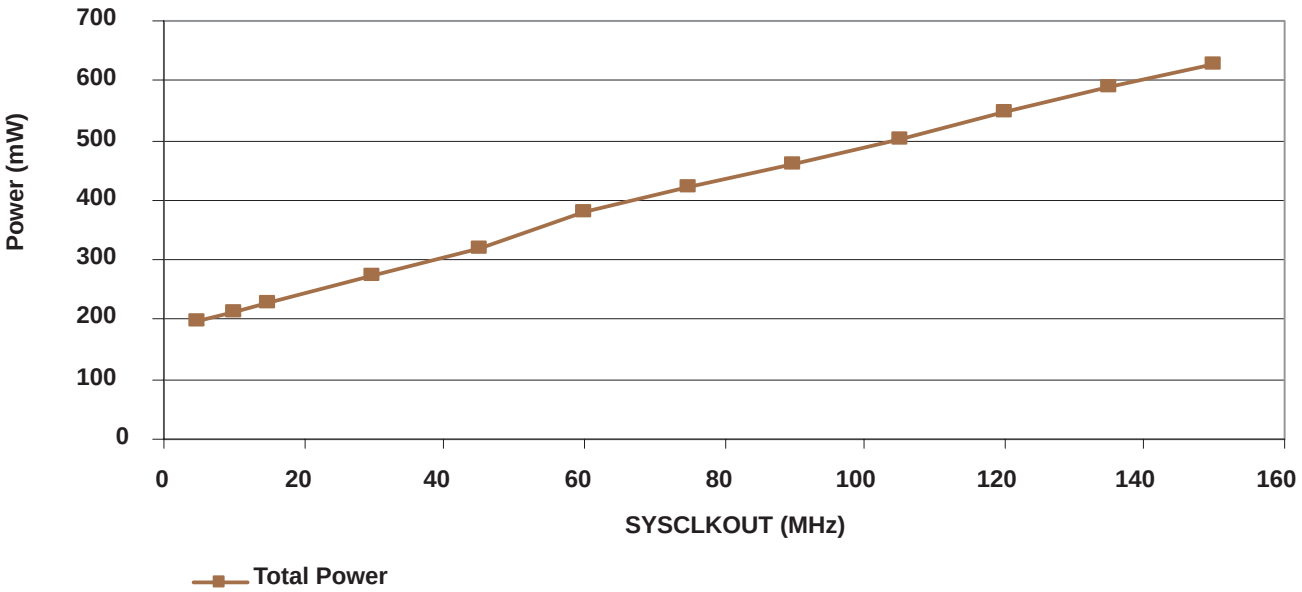
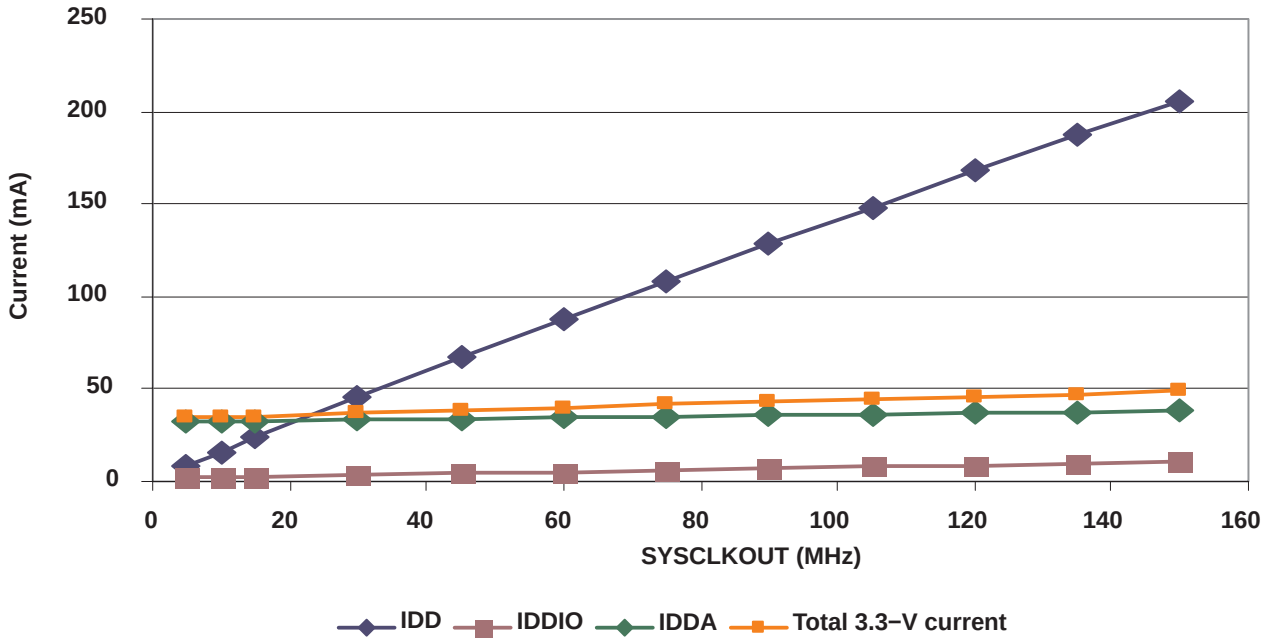


Figure 6−2. F2812/F2811/F2810 Typical Power Consumption Over Frequency



NOTES: A. Test conditions are as defined in Table 6−5 for operational currents.
B. I_{DD} represents the total current drawn from the 1.8-V rail (V_{DD}). It includes a trivial amount of current (<1 mA) drawn by V_{DD1}.
C. IDDA represents the current drawn by VDDA1 and VDDA2 rails.
D. Total 3.3-V current is the sum of I_{DDIO} and I_{DDA}. It includes a trivial amount of current (<1 mA) drawn by VDDAIO.

Figure 6−3. C2812/C2811/C2810 Typical Current Consumption Over Frequency

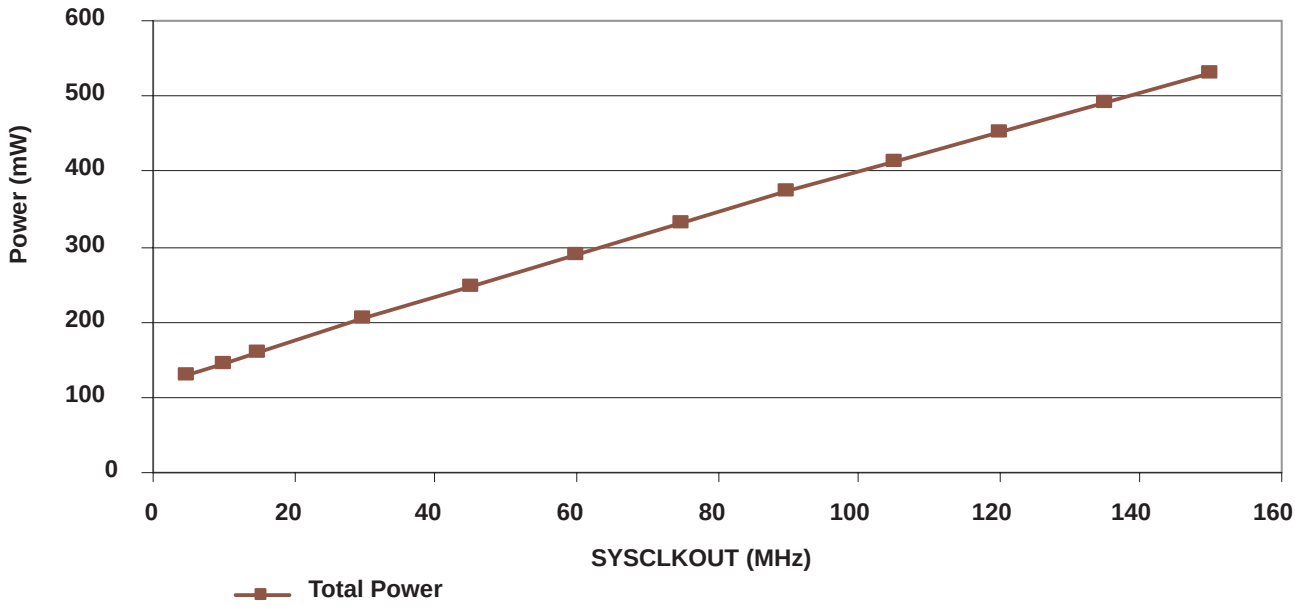


Figure 6−4. C2812/C2811/C2810 Typical Power Consumption Over Frequency

6.7 Reducing Current Consumption

28x DSPs incorporate a unique method to reduce the device current consumption. A reduction in current consumption can be achieved by turning off the clock to any peripheral module which is not used in a given application. Table 6−1 indicates the typical reduction in current consumption achieved by turning off the clocks to various peripherals.

Table 6−1. Typical Current Consumption by Various Peripherals (at 150 MHz)†

PERIPHERAL MODULE	I_{DD} CURRENT REDUCTION (mA)
eCAN	12
EVA	6
EVB	6
ADC	8‡
SCI	4
SPI	5
McBSP	13

† All peripheral clocks are disabled upon reset. Writing to/reading from peripheral registers is possible only after the peripheral clocks are turned on.

‡ This number represents the current drawn by the digital portion of the ADC module. Turning off the clock to the ADC module results in the elimination of the current drawn by the analog portion of the ADC (I_{CCA}) as well.

6.8 Power Sequencing Requirements

TMS320F2812/F2811/F2810 silicon requires dual voltages (1.8-V or 1.9-V and 3.3-V) to power up the CPU, Flash, ROM, ADC, and the I/Os. To ensure the correct reset state for all modules during power up, there are some requirements to be met while powering up/powering down the device. The current F2812 silicon reference schematics (Spectrum Digital Incorporated eZdsp. board) suggests two options for the power sequencing circuit.

Power sequencing is not needed for C281x devices. In other words, 3.3-V and 1.8-V (or 1.9-V) can ramp together. C281x can also be used on boards that have F281x power sequencing implemented; however, if the 1.8-V (or 1.9-V) rail lags the 3.3-V rail, the GPIO pins are undefined until the 1.8-V rail reaches at least 1 V.

- Option 1:

  In this approach, an external power sequencing circuit enables V_{DDIO} first, then V_{DD} and V_{DD1} (1.8 V or 1.9 V). After 1.8 V (or 1.9 V) ramps, the 3.3 V for Flash (V_{DD3VFL}) and ADC ($V_{DDA1}/V_{DDA2}/AV_{DDREFBG}$) modules are ramped up. While option 1 is still valid, TI has simplified the requirement. Option 2 is the recommended approach.

- Option 2:

  Enable power to all 3.3-V supply pins (V_{DDIO}, V_{DD3VFL}, $V_{DDA1}/V_{DDA2}/V_{DDAIO}/AV_{DDREFBG}$) and then ramp 1.8 V (or 1.9 V) (V_{DD}/V_{DD1}) supply pins.

  1.8 V or 1.9 V (V_{DD}/V_{DD1}) should not reach 0.3 V until V_{DDIO} has reached 2.5 V. This ensures the reset signal from the I/O pin has propagated through the I/O buffer to provide power-on reset to all the modules inside the device. See Figure 6−10 for power-on reset timing.

- Power-Down Sequencing:

  During power-down, the device reset should be asserted low (8 µs, minimum) before the V_{DD} supply reaches 1.5 V. This will help to keep on-chip flash logic in reset prior to the V_{DDIO}/V_{DD} power supplies ramping down. It is recommended that the device reset control from “Low-Dropout (LDO)” regulators or

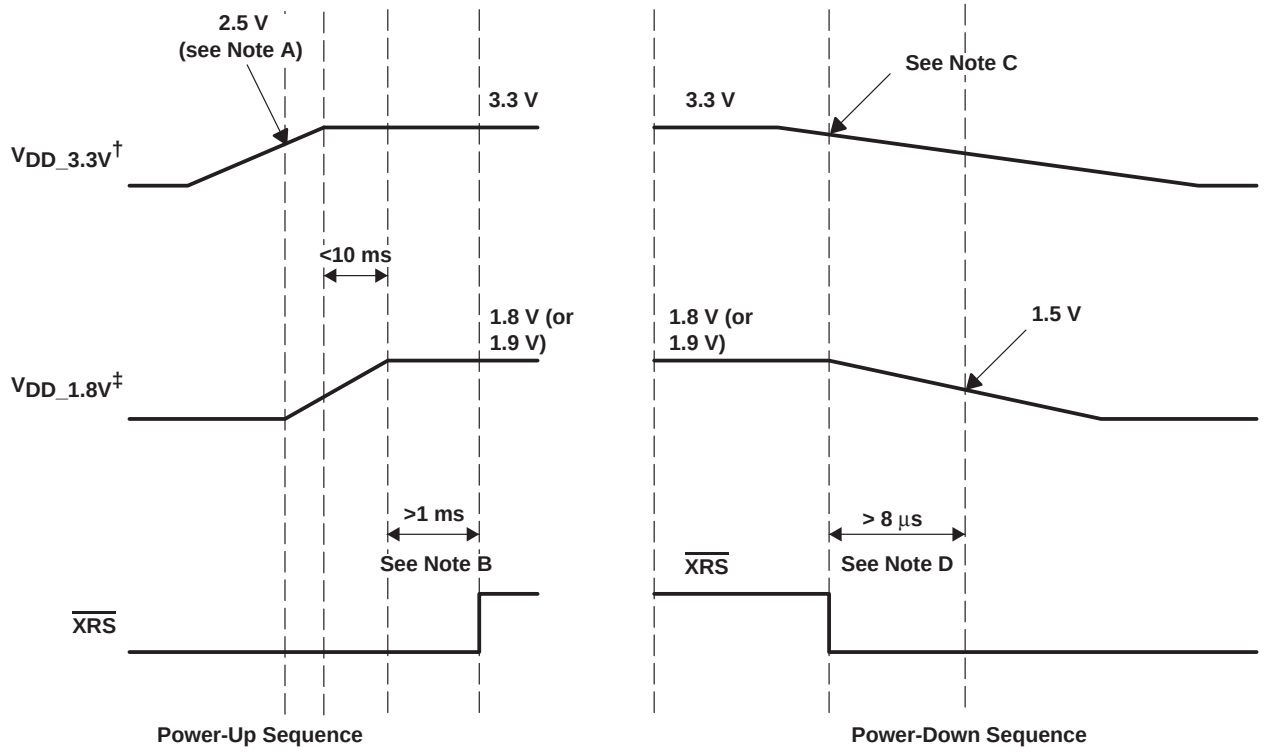
eZdsp is a trademark of Spectrum Digital Incorporated.

voltage supervisors be used to meet this constraint. LDO regulators that facilitate power-sequencing (with the aid of additional external components) may be used to meet the power sequencing requirement. See www.spectrumdigital.com for F2812 eZdsp™ schematics and updates.

Table 6−2. Recommended "Low-Dropout Regulators"

SUPPLIER	PART NUMBER
Texas Instruments	TPS767D301

NOTE:
The GPIO pins are undefined until V_{DD} = 1 V and V_{DDIO} = 2.5 V.



† $V_{DD\_3.3V}$ − V_{DDIO}, V_{DD3VFL}, V_{DDAIO}, V_{DDA1}, V_{DDA2}, $AV_{DDREFBG}$
‡ $V_{DD\_1.8V}$ − V_{DD}, V_{DD1}

NOTES: A. 1.8-V (or 1.9 V) supply should ramp after the 3.3-V supply reaches at least 2.5 V.
B. Reset ($\overline{XRS}$) should remain low until supplies and clocks are stable. See Figure 6−10, Power-on Reset in Microcomputer Mode ($XMP/\overline{MC}$ = 0), for minimum requirements.
C. Voltage supervisor or LDO reset control will trip reset ($\overline{XRS}$) first when the 3.3-V supply is off regulation. Typically, this occurs a few milliseconds before the 1.8-V (or 1.9 V) supply reaches 1.5 V.
D. Keeping reset low ($\overline{XRS}$) at least 8 µs prior to the 1.8-V (or 1.9 V) supply reaching 1.5 V will keep the flash module in complete reset before the supplies ramp down.
E. Since the state of GPIO pins is undefined until the 1.8-V (or 1.9 V) supply reaches at least 1 V, this supply should be ramped as quickly as possible (after the 3.3-V supply reaches at least 2.5 V).
F. Other than the power supply pins, no pin should be driven before the 3.3-V rail has been fully powered up.

Figure 6−5. F2812/F2811/F2810 Typical Power-Up and Power-Down Sequence − Option 2

6.9 Signal Transition Levels

Note that some of the signals use different reference voltages, see the recommended operating conditions table. Output levels are driven to a minimum logic-high level of 2.4 V and to a maximum logic-low level of 0.4 V.

Figure 6−6 shows output levels.

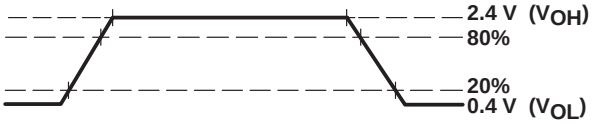


Figure 6−6. Output Levels

Output transition times are specified as follows:

- For a *high-to-low transition*, the level at which the output is said to be no longer high is below 80% of the total voltage range and lower and the level at which the output is said to be low is 20% of the total voltage range and lower.
- For a *low-to-high transition*, the level at which the output is said to be no longer low is 20% of the total voltage range and higher and the level at which the output is said to be high is 80% of the total voltage range and higher.

Figure 6−7 shows the input levels.

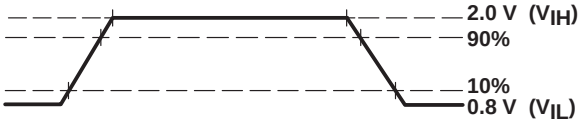


Figure 6−7. Input Levels

Input transition times are specified as follows:

- For a *high-to-low transition* on an input signal, the level at which the input is said to be no longer high is 90% of the total voltage range and lower and the level at which the input is said to be low is 10% of the total voltage range and lower.
- For a *low-to-high transition* on an input signal, the level at which the input is said to be no longer low is 10% of the total voltage range and higher and the level at which the input is said to be high is 90% of the total voltage range and higher.

NOTE: See the individual timing diagrams for levels used for testing timing parameters.

6.10 Timing Parameter Symbology

Timing parameter symbols used are created in accordance with JEDEC Standard 100. To shorten the symbols, some of the pin names and other related terminology have been abbreviated as follows:

Lowercase subscripts and their meanings:	
a	access time
c	cycle time (period)
d	delay time
f	fall time
h	hold time
r	rise time
su	setup time
t	transition time
v	valid time
w	pulse duration (width)

Letters and symbols and their meanings:	
H	High
L	Low
V	Valid
X	Unknown, changing, or don't care level
Z	High impedance

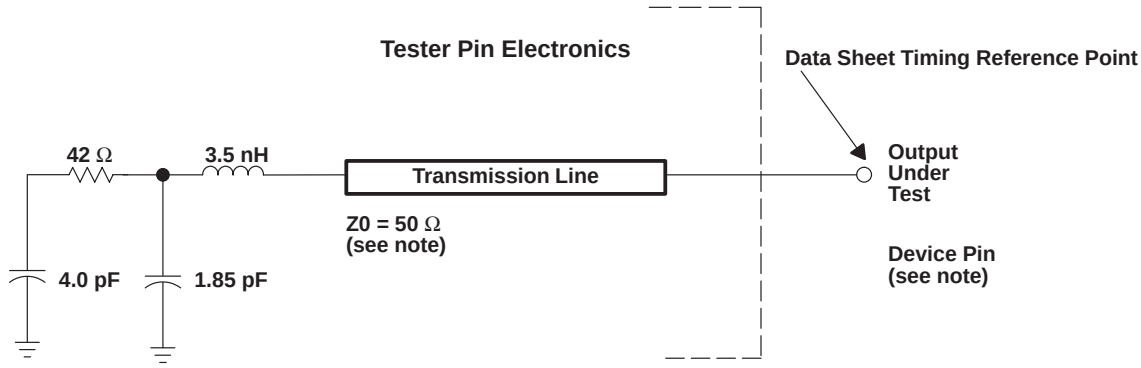
6.11 General Notes on Timing Parameters

All output signals from the 28x devices (including XCLKOUT) are derived from an internal clock such that all output transitions for a given half-cycle occur with a minimum of skewing relative to each other.

The signal combinations shown in the following timing diagrams may not necessarily represent actual cycles. For actual cycle examples, see the appropriate cycle description section of this document.

6.12 Test Load Circuit

This test load circuit is used to measure all switching characteristics provided in this document.



NOTE: The data sheet provides timing at the device pin. For output timing analysis, the tester pin electronics and its transmission line effects must be taken into account. A transmission line with a delay of 2 ns or longer can be used to produce the desired transmission line effect. The transmission line is intended as a load only. It is not necessary to add or subtract the transmission line delay (2 ns or longer) from the data sheet timing.

Input requirements in this data sheet are tested with an input slew rate of < 4 Volts per nanosecond (4 V/ns) at the device pin.

Figure 6−8. 3.3-V Test Load Circuit

6.13 Device Clock Table

This section provides the timing requirements and switching characteristics for the various clock options available on the F281x and C281x DSPs. Table 6−3 lists the cycle times of various clocks.

Table 6−3. TMS320F281x and TMS320C281x Clock Table and Nomenclature

		MIN	NOM	MAX	UNIT
On-chip oscillator clock	$t_{c(OSC)}$, Cycle time	28.6		50	ns
	Frequency	20		35	MHz
XCLKIN	$t_{c(CI)}$, Cycle time	6.67		250	ns
	Frequency	4		150	MHz
SYSCLKOUT	$t_{c(SCO)}$, Cycle time	6.67		500	ns
	Frequency	2		150	MHz
XCLKOUT	$t_{c(XCO)}$, Cycle time	6.67		2000	ns
	Frequency	0.5		150	MHz
HSPCLK	$t_{c(HCO)}$, Cycle time	6.67	13.3‡		ns
	Frequency		75‡	150	MHz
LSPCLK	$t_{c(LCO)}$, Cycle time	13.3	26.6‡		ns
	Frequency		37.5‡	75	MHz
ADC clock	$t_{c(ADCCLK)}$, Cycle time†	40			ns
	Frequency			25	MHz
SPI clock	$t_{c(SPC)}$, Cycle time	50			ns
	Frequency			20	MHz
McBSP	$t_{c(CKG)}$, Cycle time	50			ns
	Frequency			20	MHz
XTIMCLK	$t_{c(XTIM)}$, Cycle time	6.67			ns
	Frequency			150	MHz

† The maximum value for ADCCLK frequency is 25 MHz. For SYSCLKOUT values of 25 MHz or lower, ADCCLK has to be SYSCLKOUT/2 or lower. ADCCLK = SYSCLKOUT is not a valid mode for any value of SYSCLKOUT.
‡ This is the default reset value if SYSCLKOUT = 150 MHz.

6.14 Clock Requirements and Characteristics

6.14.1 Input Clock Requirements

The clock provided at the XCLKIN pin generates the internal CPU clock cycle.

Table 6−4. Input Clock Frequency

PARAMETER			MIN	TYP	MAX	UNIT
f_X	Input clock frequency	Resonator	20		35	MHz
		Crystal	20		35	
		XCLKIN	4		150	
f_l	Limp mode clock frequency			2		MHz

Table 6−5. XCLKIN Timing Requirements − PLL Bypassed or Enabled

NO.				MIN	MAX	UNIT
C8	$t_{c(CI)}$	Cycle time, XCLKIN		6.67	250	ns
C9	$t_{f(CI)}$	Fall time, XCLKIN	Up to 30 MHz		6	ns
			30 MHz to 150 MHz		2	
C10	$t_{r(CI)}$	Rise time, XCLKIN	Up to 30 MHz		6	ns
			30 MHz to 150 MHz		2	
C11	$t_{w(CIL)}$	Pulse duration, X1/XCLKIN low as a percentage of $t_{c(CI)}$		40	60	%
C12	$t_{w(CIH)}$	Pulse duration, X1/XCLKIN high as a percentage of $t_{c(CI)}$		40	60	%

Table 6−6. XCLKIN Timing Requirements − PLL Disabled

NO.				MIN	MAX	UNIT
C8	$t_{c(CI)}$	Cycle time, XCLKIN		6.67	250	ns
C9	$t_{f(CI)}$	Fall time, XCLKIN	Up to 30 MHz		6	ns
			30 MHz to 150 MHz		2	
C10	$t_{r(CI)}$	Rise time, XCLKIN	Up to 30 MHz		6	ns
			30 MHz to 150 MHz		2	
C11	$t_{w(CIL)}$	Pulse duration, X1/XCLKIN low as a percentage of $t_{c(CI)}$	XCLKIN ≤ 120 MHz	40	60	%
			120 < XCLKIN ≤ 150 MHz	45	55	
C12	$t_{w(CIH)}$	Pulse duration, X1/XCLKIN high as a percentage of $t_{c(CI)}$	XCLKIN ≤ 120 MHz	40	60	%
			120 < XCLKIN ≤ 150 MHz	45	55	

Table 6−7. Possible PLL Configuration Modes

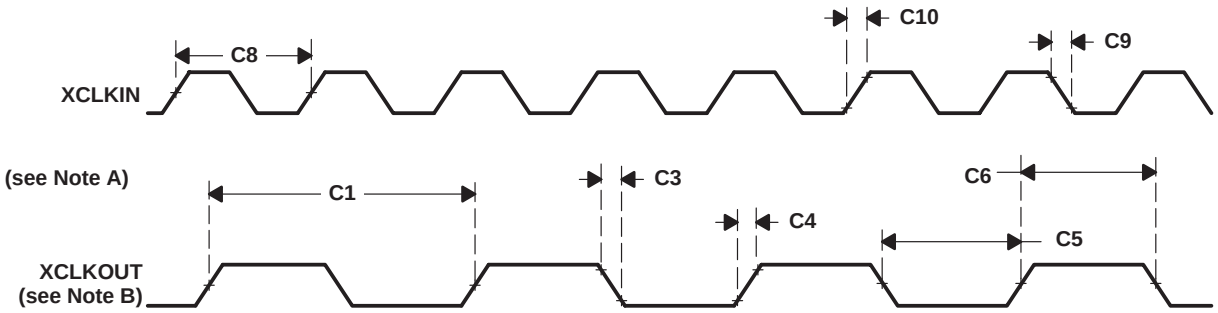
PLL MODE	REMARKS	SYSCLKOUT
PLL Disabled	Invoked by tying $\overline{\text{XPLLDIS}}$ pin low upon reset. PLL block is completely disabled. Clock input to the CPU (CLKIN) is directly derived from the clock signal present at the X1/XCLKIN pin.	XCLKIN
PLL Bypassed	Default PLL configuration upon power-up, if PLL is not disabled. The PLL itself is bypassed. However, the /2 module in the PLL block divides the clock input at the X1/XCLKIN pin by two before feeding it to the CPU.	XCLKIN/2
PLL Enabled	Achieved by writing a non-zero value “n” into PLLCR register. The /2 module in the PLL block now divides the output of the PLL by two before feeding it to the CPU.	(XCLKIN * n) / 2

6.14.2 Output Clock Characteristics

Table 6−8. XCLKOUT Switching Characteristics (PLL Bypassed or Enabled)†‡

No.	PARAMETER		MIN	TYP	MAX	UNIT
C1	$t_{c(XCO)}$	Cycle time, XCLKOUT	6.67§			ns
C3	$t_{f(XCO)}$	Fall time, XCLKOUT		2		ns
C4	$t_{r(XCO)}$	Rise time, XCLKOUT		2		ns
C5	$t_{w(XCOL)}$	Pulse duration, XCLKOUT low	H−2		H+2	ns
C6	$t_{w(XCOH)}$	Pulse duration, XCLKOUT high	H−2		H+2	ns
C7	t_p	PLL lock time¶			$131072t_{c(CI)}$	ns

† A load of 40 pF is assumed for these parameters.
‡ $H = 0.5t_{c(XCO)}$
§ The PLL must be used for maximum frequency operation.
¶ This parameter has changed from 4096 XCLKIN cycles in the earlier revisions of the silicon.



NOTES: A. The relationship of XCLKIN to XCLKOUT depends on the divide factor chosen. The waveform relationship shown in Figure 6−9 is intended to illustrate the timing parameters only and may differ based on configuration.
B. XCLKOUT configured to reflect SYSCLKOUT.

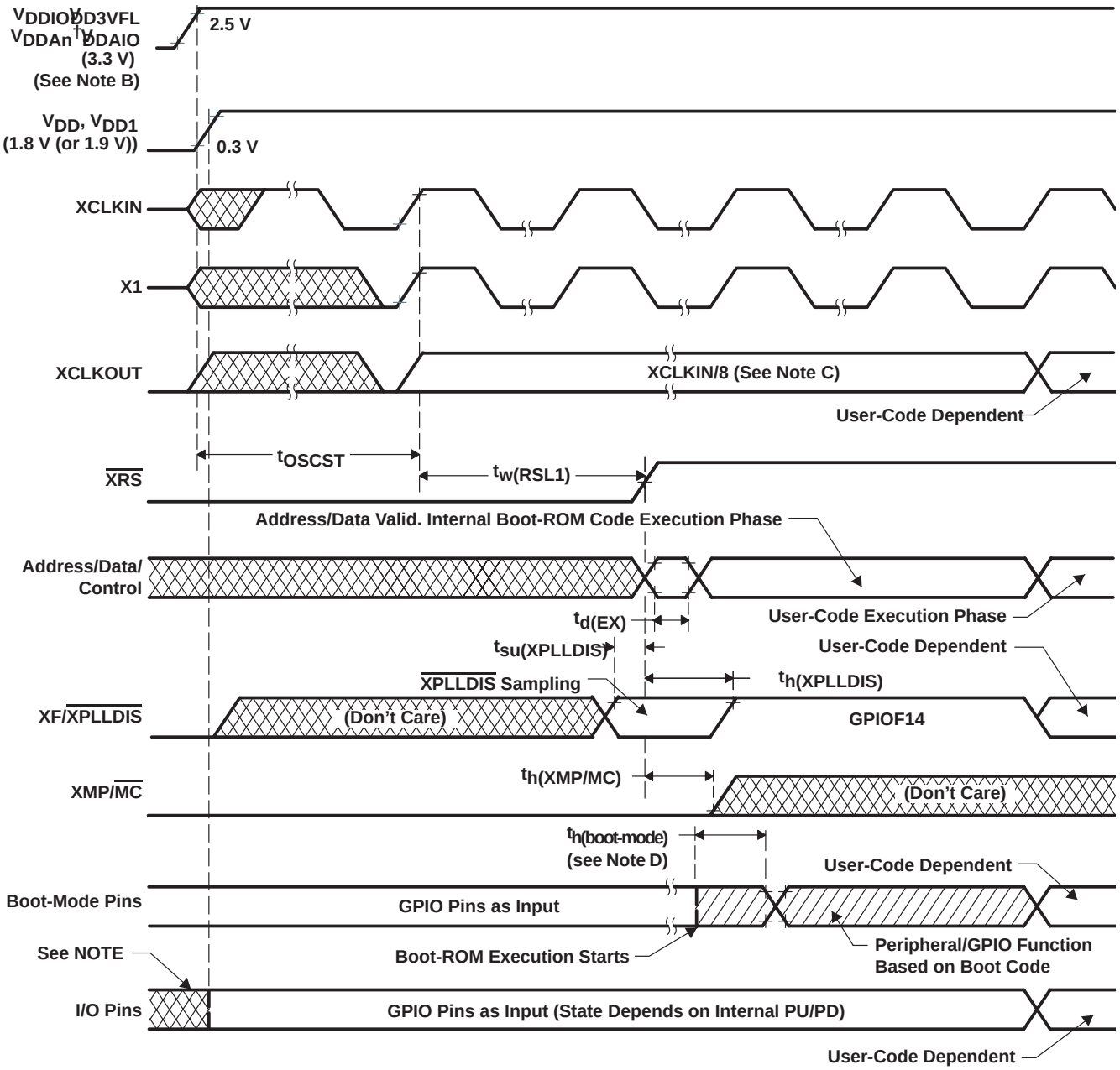
Figure 6−9. Clock Timing

6.15 Reset Timing

Table 6−9. Reset ($\overline{XRS}$) Timing Requirements†

			MIN	NOM	MAX	UNIT
$t_{w(RSL1)}$	Pulse duration, stable XCLKIN to $\overline{XRS}$ high		$8t_{c(CI)}$			cycles
$t_{w(RSL2)}$	Pulse duration, $\overline{XRS}$ low	Warm reset	$8t_{c(CI)}$			cycles
		WD-initiated reset		$512t_{c(CI)}$		
$t_{w(WDRS)}$	Pulse duration, reset pulse generated by watchdog			$512t_{c(CI)}$		cycles
$t_{d(EX)}$	Delay time, address/data valid after $\overline{XRS}$ high			$32t_{c(CI)}$		cycles
t_{OSCST}‡	Oscillator start-up time		1	10		ms
$t_{su(XPLLDIS)}$	Setup time for $\overline{XPLLDIS}$ pin		$16t_{c(CI)}$			cycles
$t_{h(XPLLDIS)}$	Hold time for $\overline{XPLLDIS}$ pin		$16t_{c(CI)}$			cycles
$t_{h(XMP/MC)}$	Hold time for XMP/$\overline{MC}$ pin		$16t_{c(CI)}$			cycles
$t_{h(boot-mode)}$	Hold time for boot-mode pins		$2520t_{c(CI)}$§			cycles

† If external oscillator/clock source are used, reset time has to be low at least for 1 ms after V_{DD} reaches 1.5 V.
‡ Dependent on crystal/resonator and board design.
§ The boot ROM reads the password locations. Therefore, this timing requirement includes the wakeup time for flash. See the *TMS320x281x Boot ROM Reference Guide* (literature number SPRU095) and *TMS320x281x System Control and Interrupts Reference Guide* (literature number SPRU078) for further information.



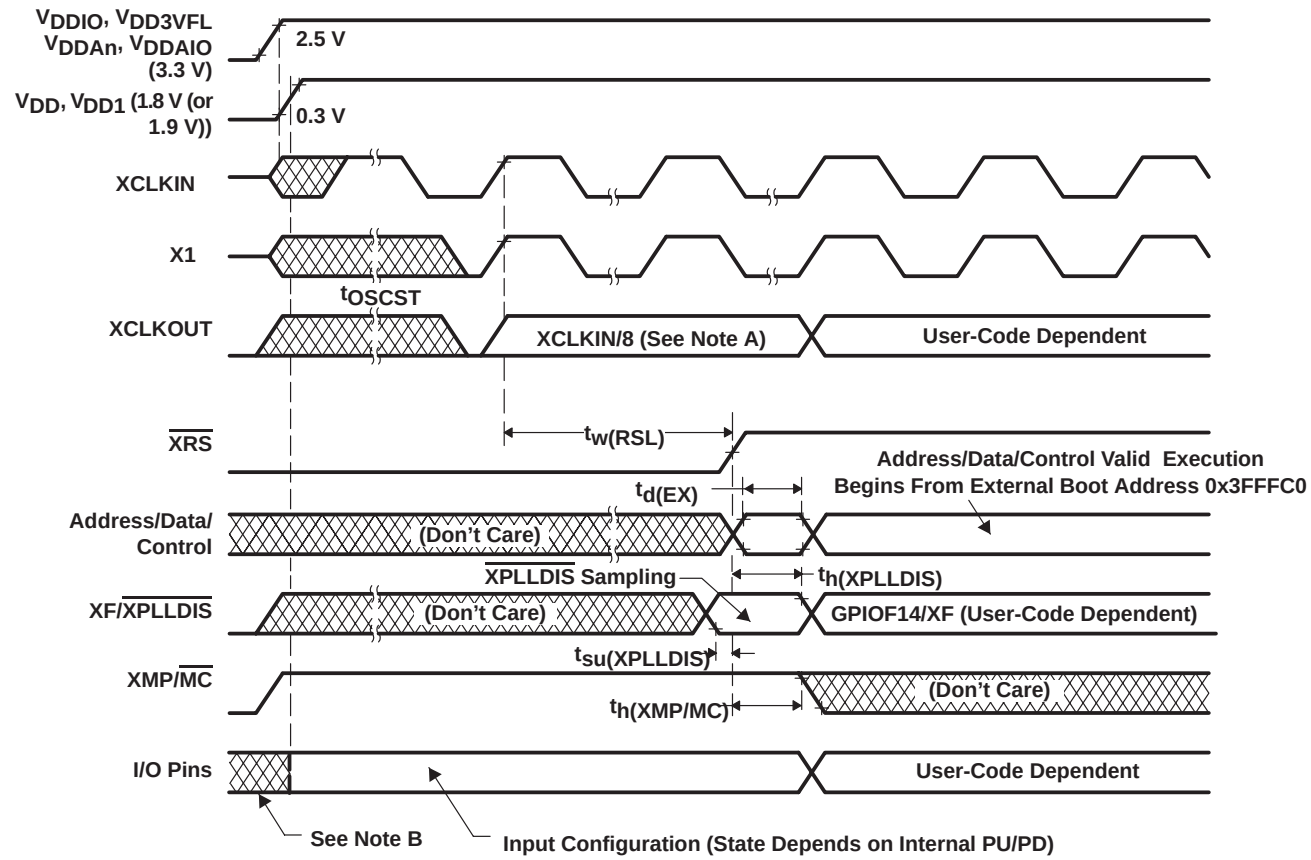
NOTES: A. The state of the GPIO pins is undefined (i.e., they could be input or output) until the 1.8-V (or 1.9-V) supply reaches at least 1 V and 3.3-V supply reaches 2.5 V.

B. V_{DDAn} − V_{DDA1}/V_{DDA2} and $AV_{DDREFBG}$

C. Upon power up, SYSCLKOUT is XCLKIN/2 if the PLL is enabled. Since both the XTIMCLK and CLKMODE bits in the XINTCNF2 register come up with a reset state of 1, SYSCLKOUT is further divided by 4 before it appears at XCLKOUT. This explains why XCLKOUT = XCLKIN/8 during this phase.

D. After reset, the Boot ROM code executes instructions for 1260 SYSCLKOUT cycles (SYSCLKOUT = XCLKIN/2) and then samples BOOT Mode pins. Based on the status of the Boot Mode pin, the boot code branches to destination memory or boot code function in ROM. The BOOT Mode pins should be held high/low for at least 2520 XCLKIN cycles from boot ROM execution time for proper selection of Boot modes.
If Boot ROM code executes after power-on conditions (in debugger environment), the Boot code execution time is based on the current SYSCLKOUT speed. The SYSCLKOUT will be based on user environment and could be with or without PLL enabled.

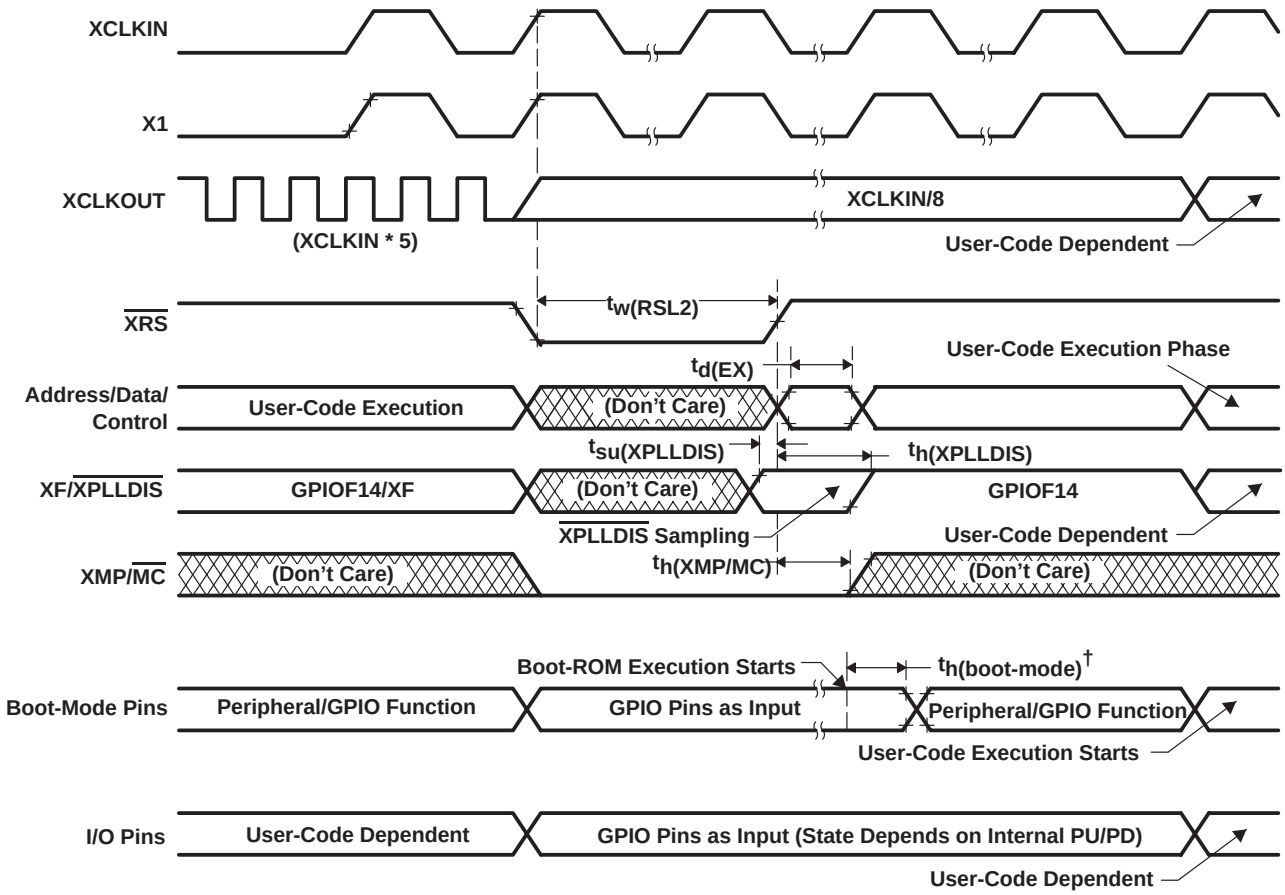
Figure 6−10. Power-on Reset in Microcomputer Mode (XMP/$\overline{MC}$ = 0) (See Note A)



NOTES: A. Upon power up, SYSCLKOUT is XCLKIN/2 if the PLL is enabled. Since both the XTIMCLK and CLKMODE bits in the XINTCNF2 register come up with a reset state of 1, SYSCLKOUT is further divided by 4 before it appears at XCLKOUT. This explains why XCLKOUT = XCLKIN/8 during this phase.

B. The state of the GPIO pins is undefined (i.e., they could be input or output) until the 1.8-V (or 1.9-V) supply reaches at least 1 V and 3.3-V supply reaches 2.5 V..

Figure 6−11. Power-on Reset in Microprocessor Mode (XMP/$\overline{MC}$ = 1)



† After reset, the Boot ROM code executes instructions for 1260 SYSCLKOUT cycles (SYSCLKOUT = XCLKIN/2) and then samples BOOT Mode pins. Based on the status of the Boot Mode pin, the boot code branches to destination memory or boot code function in ROM. The BOOT Mode pins should be held high/low for at least 2520 XCLKIN cycles from boot ROM execution time for proper selection of Boot modes.
If Boot ROM code executes after power-on conditions (in debugger environment), the Boot code execution time is based on the current SYSCLKOUT speed. The SYSCLKOUT will be based on user environment and could be with or without PLL enabled.

Figure 6−12. Warm Reset in Microcomputer Mode

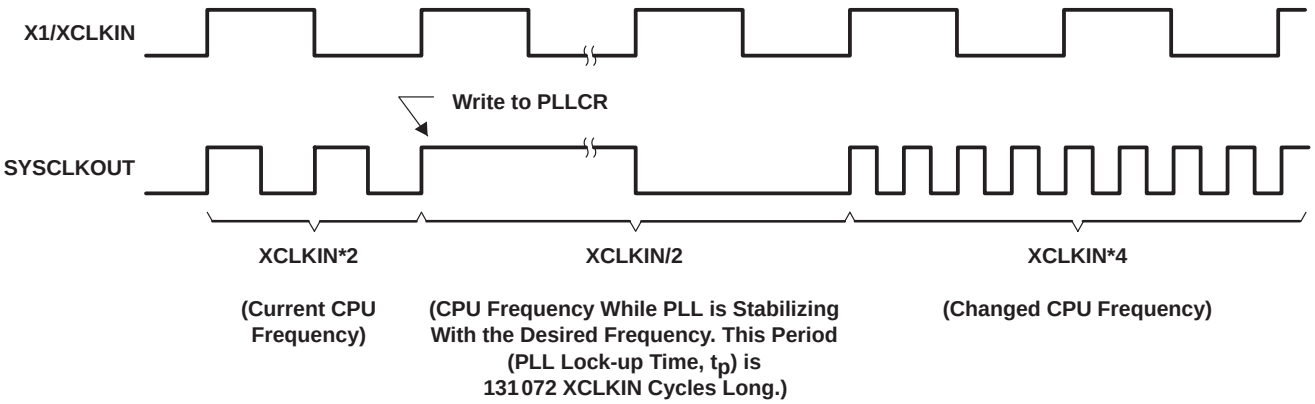


Figure 6−13. Effect of Writing Into PLLCR Register

6.16 Low-Power Mode Wakeup Timing

Table 6−10. IDLE Mode Timing Requirements

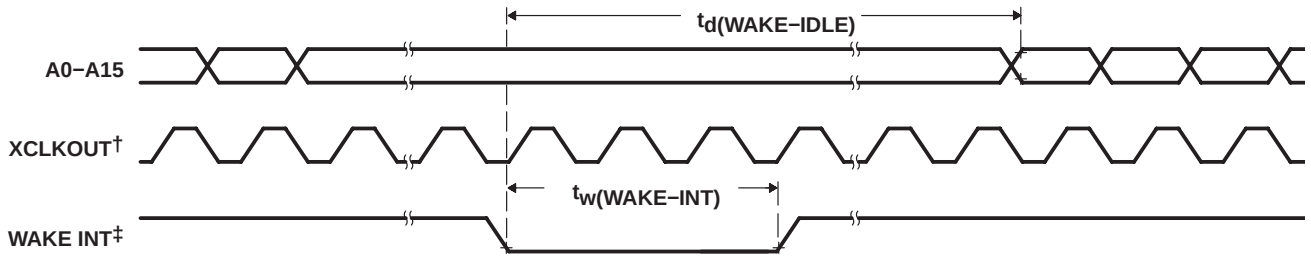
		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{w(WAKE-INT)}$	Pulse duration, external wake-up signal	Without input qualifier	$2 * t_{c(SCO)}$			Cycles
		With input qualifier	$1 * t_{c(SCO)} + IQT^{\dagger}$			Cycles

Table 6−11. IDLE Mode Switching Characteristics

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{d(WAKE-IDLE)}$	Delay time, external wake signal to program execution resume‡					
	− Wake-up from Flash − Flash module in active state	Without input qualifier	$8 * t_{c(SCO)}$			Cycles
	− Wake-up from Flash − Flash module in active state	With input qualifier	$8 * t_{c(SCO)} + IQT^{\dagger}$			Cycles
	− Wake-up from Flash − Flash module in sleep state	Without input qualifier	$1050 * t_{c(SCO)}$			Cycles
	− Wake-up from Flash − Flash module in sleep state	With input qualifier	$1050 * t_{c(SCO)} + IQT^{\dagger}$			Cycles
	− Wake-up from SARAM	Without input qualifier	$8 * t_{c(SCO)}$			Cycles
	− Wake-up from SARAM	With input qualifier	$8 * t_{c(SCO)} + IQT^{\dagger}$			Cycles

† Input Qualification Time (IQT) = [5 x QUALPRD x 2] * $t_{c(SCO)}$

‡ This is the time taken to begin execution of the instruction that immediately follows the IDLE instruction. Execution of an ISR (triggered by the wake-up) signal involves additional latency.



† XCLKOUT = SYSCLKOUT

‡ WAKE INT can be any enabled interrupt, $\overline{\text{WDINT}}$, XNMI, or $\overline{\text{XRS}}$.

Figure 6−14. IDLE Entry and Exit Timing

Table 6−12. STANDBY Mode Timing Requirements

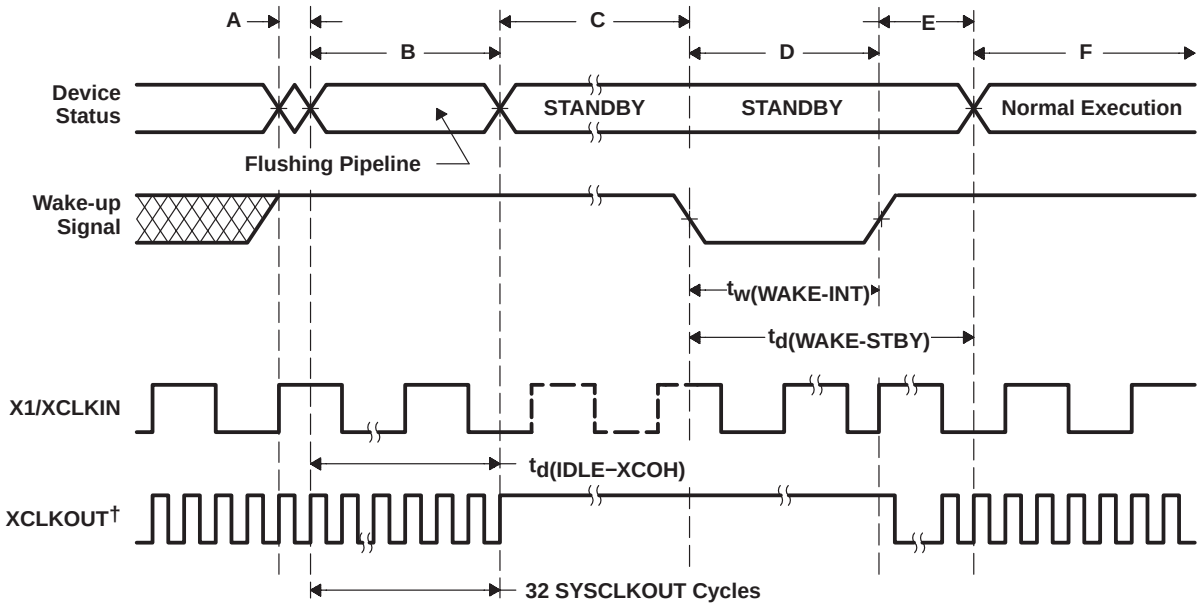
		TEST CONDITIONS	MIN	NOM	MAX	UNIT
$t_{w(WAKE-INT)}$	Pulse duration, external wake-up signal	Without input qualifier	$12 * t_{c(CI)}$			Cycles
		With input qualifier	$(2 + QUALSTDBY)^{\dagger} * t_{c(CI)}$			Cycles

† QUALSTDBY is a 6-bit field in the LPMCR0 register.

Table 6−13. STANDBY Mode Switching Characteristics

PARAMETER		TEST CONDITIONS	MIN	TYP	MAX	UNIT
$t_{d(IDLE-XCOH)}$	Delay time, IDLE instruction executed to XCLKOUT high		32 * $t_{c(SCO)}$	12 * $t_{c(CI)}$		Cycles
$t_{d(WAKE-STBY)}$	Delay time, external wake signal to program execution resume†					
	− Wake-up from Flash − Flash module in active state	Without input qualifier	12 * $t_{c(CI)}$			Cycles
	− Wake-up from Flash − Flash module in active state	With input qualifier	12 * $t_{c(CI)}$ + $t_{w(WAKE-INT)}$			Cycles
	− Wake-up from Flash − Flash module in sleep state	Without input qualifier	1125 * $t_{c(SCO)}$			Cycles
	− Wake-up from Flash − Flash module in sleep state	With input qualifier	1125 * $t_{c(SCO)}$ + $t_{w(WAKE-INT)}$			Cycles
	− Wake-up from SARAM	Without input qualifier	12 * $t_{c(CI)}$			Cycles
	− Wake-up from SARAM	With input qualifier	12 * $t_{c(CI)}$ + $t_{w(WAKE-INT)}$			Cycles

† This is the time taken to begin execution of the instruction that immediately follows the IDLE instruction. Execution of an ISR (triggered by the wake-up) signal involves additional latency.



NOTES: A. IDLE instruction is executed to put the device into STANDBY mode.
B. The PLL block responds to the STANDBY signal. SYSCLKOUT is held for approximately 32 cycles before being turned off. This 32-cycle delay enables the CPU pipe and any other pending operations to flush properly.
C. The device is now in STANDBY mode.
D. The external wake-up signal is driven active (negative edge triggered shown as an example).
E. After a latency period, the STANDBY mode is exited.
F. Normal operation resumes. The device will respond to the interrupt (if enabled).

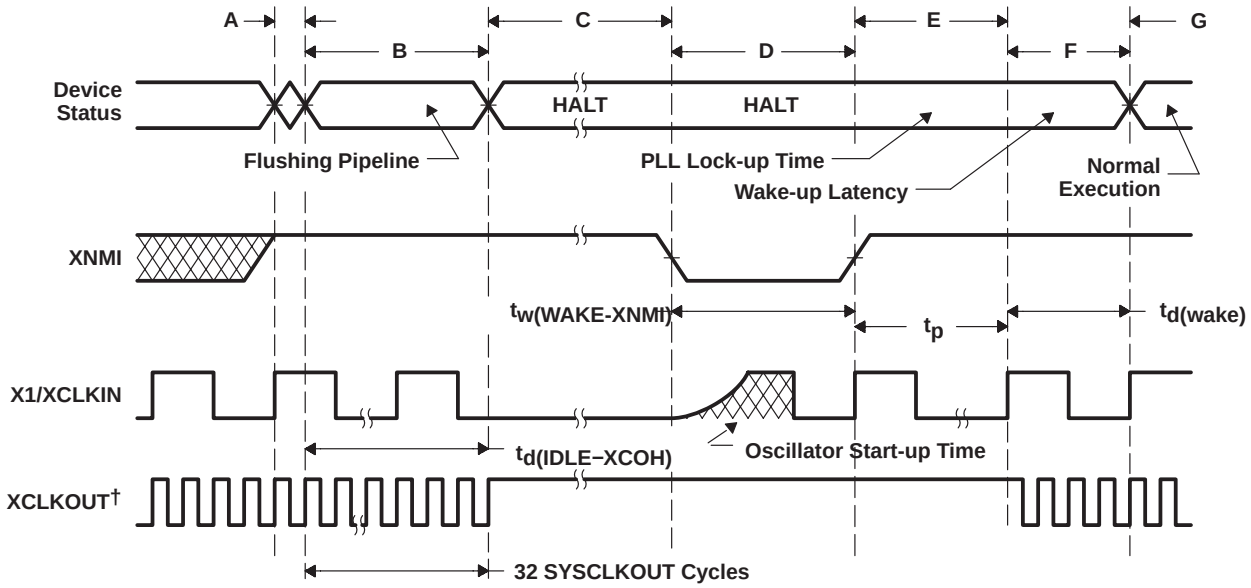
Figure 6−15. STANDBY Entry and Exit Timing

Table 6−14. HALT Mode Timing Requirements

		MIN	NOM	MAX	UNIT
$t_{w(WAKE-XNMI)}$	Pulse duration, XNMI wakeup signal	2 * $t_{c(CI)}$			Cycles
$t_{w(WAKE-XRS)}$	Pulse duration, $\overline{XRS}$ wakeup signal	8 * $t_{c(CI)}$			Cycles

Table 6−15. HALT Mode Switching Characteristics

PARAMETER		MIN	TYP	MAX	UNIT
$t_{d(IDLE-XCOH)}$	Delay time, IDLE instruction executed to XCLKOUT high	32 * $t_{c(SCO)}$	45 * $t_{c(SCO)}$		Cycles
t_p	PLL lock-up time			131072 * $t_{c(CI)}$	Cycles
$t_{d(wake)}$	Delay time, PLL lock to program execution resume				
	− Wake-up from flash − Flash module in sleep state	1125*tc(SCO)			Cycles
	− Wake-up from SARAM	35*tc(SCO)			Cycles



† XCLKOUT = SYSCLKOUT

NOTES: A. IDLE instruction is executed to put the device into HALT mode.

B. The PLL block responds to the HALT signal. SYSCLKOUT is held for another 32 cycles before the oscillator is turned off and the CLKIN to the core is stopped. This 32-cycle delay enables the CPU pipe and any other pending operations to flush properly.

C. Clocks to the device are turned off and the internal oscillator and PLL are shut down. The device is now in HALT mode and consumes absolute minimum power.

D. When XNMI is driven active (negative edge triggered shown , as an example), the oscillator is turned on; but the PLL is not activated.

E. When XNMI is deactivated, it initiates the PLL lock sequence, which takes 131,072 X1/XCLKIN cycles.

F. When CLKIN to the core is enabled, the device will respond to the interrupt (if enabled), after a latency. The HALT mode is now exited.

G. Normal operation resumes.

Figure 6−16. HALT Wakeup Using XNMI

6.17 Event Manager Interface

6.17.1 PWM Timing

PWM refers to all PWM outputs on EVA and EVB.

Table 6−16. PWM Switching Characteristics†‡

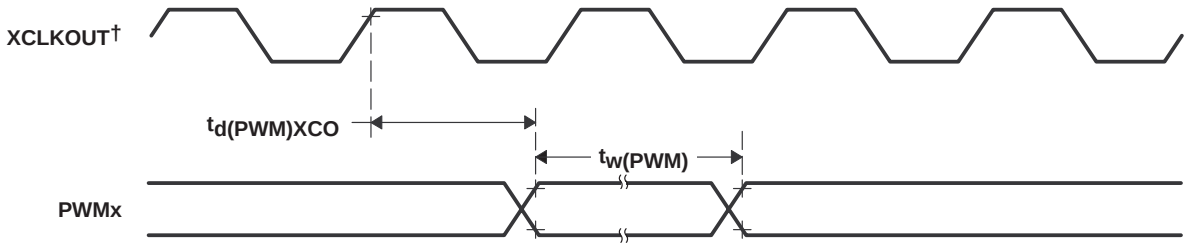
PARAMETER		TEST CONDITIONS	MIN	MAX	UNIT
$t_{w(PWM)}$§	Pulse duration, PWMx output high/low		25		ns
$t_{d(PWM)XCO}$	Delay time, XCLKOUT high to PWMx output switching	XCLKOUT = SYSCLKOUT/4		10	ns

† See the GPIO output timing for fall/rise times for PWM pins.
‡ PWM pin toggling frequency is limited by the GPIO output buffer switching frequency (20 MHz).
§ PWM outputs may be 100%, 0%, or increments of $t_{c(HCO)}$ with respect to the PWM period.

Table 6−17. Timer and Capture Unit Timing Requirements¶#

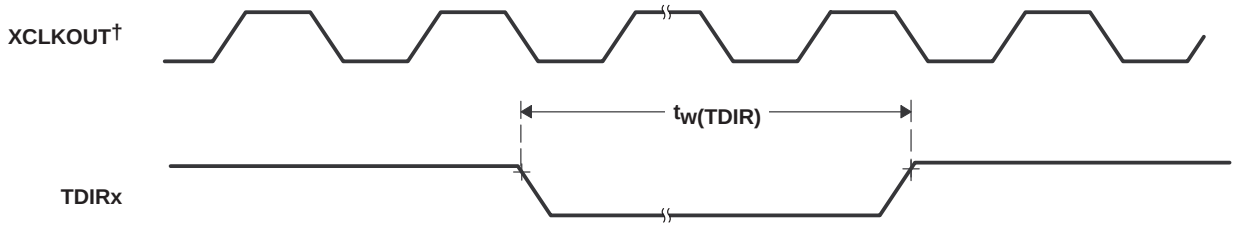
			MIN	MAX	UNIT
$t_{w(TDIR)}$	Pulse duration, TDIRx low/high	Without input qualifier	2 * $t_{c(SCO)}$		cycles
		With input qualifier	1 * $t_{c(SCO)}$ + IQT‖		
$t_{w(CAP)}$	Pulse duration, CAPx input low/high	Without input qualifier	2 * $t_{c(SCO)}$		cycles
		With input qualifier	1 * $t_{c(SCO)}$ + IQT‖		
$t_{w(TCLKINL)}$	Pulse duration, TCLKINx low as a percentage of TCLKINx cycle time		40	60	%
$t_{w(TCLKINH)}$	Pulse duration, TCLKINx high as a percentage of TCLKINx cycle time		40	60	%
$t_{c(TCLKIN)}$	Cycle time, TCLKINx		4 * $t_{c(HCO)}$		ns

¶ The QUALPRD bit field value can range from 0 (no qualification) through 0xFF (510 SYSCLKOUT cycles). The qualification sampling period is 2n SYSCLKOUT cycles, where “n” is the value stored in the QUALPRD bit field. As an example, when QUALPRD = 1, the qualification sampling period is 1 x 2 = 2 SYSCLKOUT cycles (i.e., the input is sampled every 2 SYSCLKOUT cycles). Six such samples will be taken over five sampling windows, each window being 2n SYSCLKOUT cycles. For QUALPRD = 1, the minimum width that is needed is 5 x 2 = 10 SYSCLKOUT cycles. However, since the external signal is driven asynchronously, a 11-SYSCLKOUT-wide pulse ensures reliable recognition.
# Maximum input frequency to the QEP = min[HSPCLK/2, 20 MHz]
‖ Input Qualification Time (IQT) = [5 x QUALPRD x 2] * $t_{c(SCO)}$



† XCLKOUT = SYSCLKOUT

Figure 6−17. PWM Output Timing



† XCLKOUT = SYSCLKOUT

Figure 6−18. TDIRx Timing

Table 6−18. External ADC Start-of-Conversion − EVA − Switching Characteristics†

PARAMETER		MIN	MAX	UNIT
$t_{d(XCOH-EVASOCL)}$	Delay time, XCLKOUT high to $\overline{EVASOC}$ low		1 * $t_{c(SCO)}$	cycle
$t_{w(EVASOCL)}$	Pulse duration, $\overline{EVASOC}$ low	32 * $t_{c(HCO)}$		ns

† XCLKOUT = SYSCLKOUT

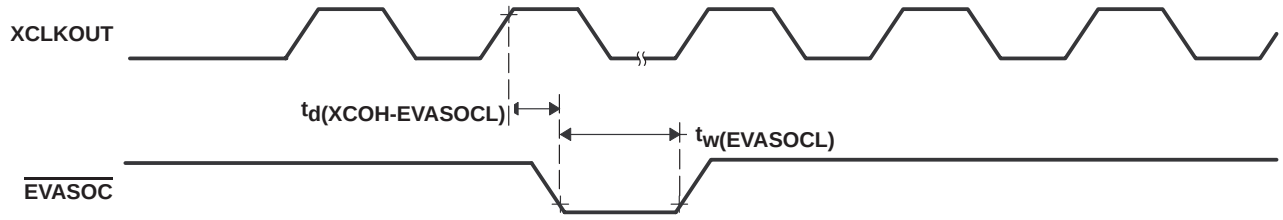


Figure 6−19. $\overline{EVASOC}$ Timing

Table 6−19. External ADC Start-of-Conversion − EVB − Switching Characteristics†

PARAMETER		MIN	MAX	UNIT
$t_{d(XCOH-EVBSOCL)}$	Delay time, XCLKOUT high to $\overline{EVBSOC}$ low		1 * $t_{c(SCO)}$	cycle
$t_{w(EVBSOCL)}$	Pulse duration, $\overline{EVBSOC}$ low	32 * $t_{c(HCO)}$		ns

† XCLKOUT = SYSCLKOUT

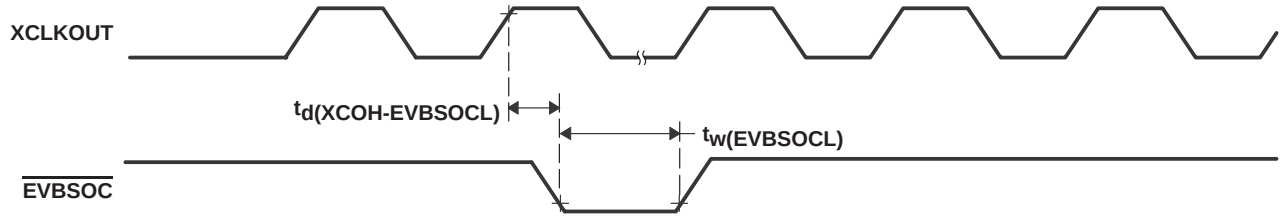


Figure 6−20. $\overline{EVBSOC}$ Timing

6.17.2 Interrupt Timing

Table 6−20. Interrupt Switching Characteristics

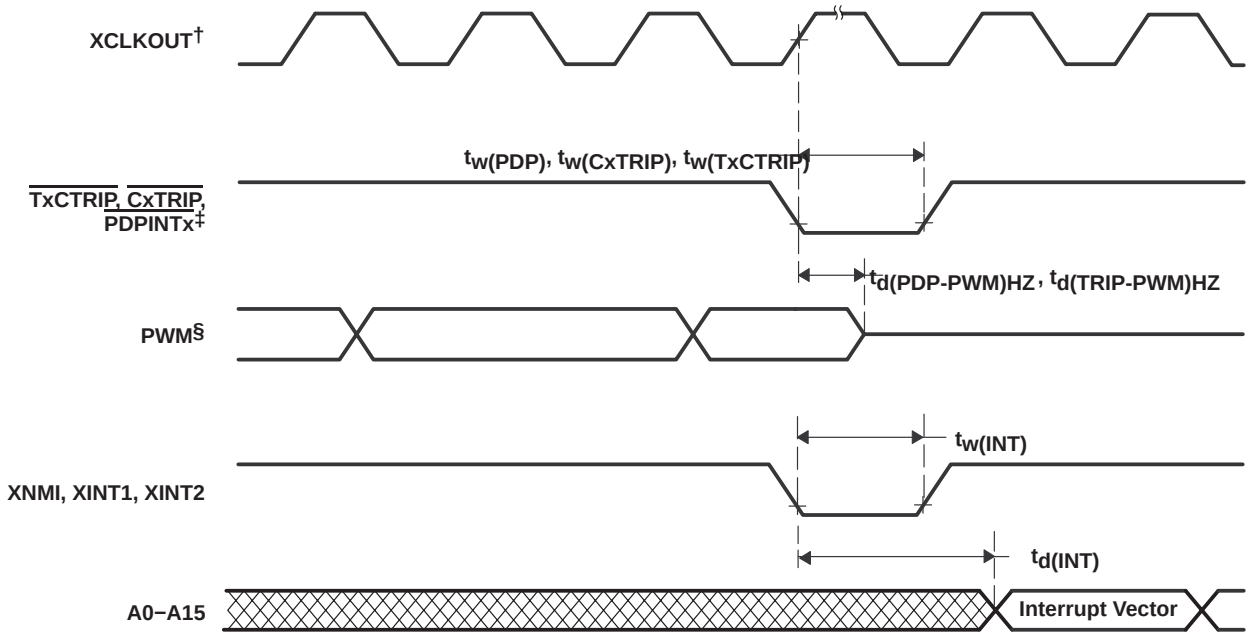
PARAMETER			MIN	MAX	UNIT
$t_{d(PDP-PWM)HZ}$	Delay time, $\overline{PDPINTx}$ low to PWM high-impedance state	Without input qualifier		12	ns
		With input qualifier		$1 * t_{c(SCO)} + IQT + 12$†	
$t_{d(TRIP-PWM)HZ}$	Delay time, $\overline{CxTRIP}$/$\overline{TxCTRIP}$ signals low to PWM high-impedance state	Without input qualifier		$3 * t_{c(SCO)}$	ns
		With input qualifier		$[2 * t_{c(SCO)}] + IQT$†	
$t_{d(INT)}$	Delay time, INT low/high to interrupt-vector fetch		$t_{qual} + 12t_{c(XCO)}$		ns

† Input Qualification Time (IQT) = [5 x QUALPRD x 2] * $t_{c(SCO)}$

Table 6−21. Interrupt Timing Requirements

			MIN MAX	UNIT
$t_{w(INT)}$	Pulse duration, INT input low/high	with no qualifier	$2 * t_{c(SCO)}$	cycles
		with qualifier	$1 * t_{c(SCO)} + IQT$†	
$t_{w(PDP)}$	Pulse duration, $\overline{PDPINTx}$ input low	with no qualifier	$2 * t_{c(SCO)}$	cycles
		with qualifier	$1 * t_{c(SCO)} + IQT$†	
$t_{w(CxTRIP)}$	Pulse duration, $\overline{CxTRIP}$ input low	with no qualifier	$2 * t_{c(SCO)}$	cycles
		with qualifier	$1 * t_{c(SCO)} + IQT$†	
$t_{w(TxCTRIP)}$	Pulse duration, $\overline{TxCTRIP}$ input low	with no qualifier	$2 * t_{c(SCO)}$	cycles
		with qualifier	$1 * t_{c(SCO)} + IQT$†	

† Input Qualification Time (IQT) = [5 x QUALPRD x 2] * $t_{c(SCO)}$



† XCLKOUT = SYSCLKOUT
‡ TxCTRIP − T1CTRIP, T2CTRIP, T3CTRIP, T4CTRIP
CxTRIP − C1TRIP, C2TRIP, C3TRIP, C4TRIP, C5TRIP, or C6TRIP
PDPINTx − PDPINTA or PDPINTB
§ PWM refers to **all** the PWM pins in the device (i.e., PWMn and TnPWM pins or PWM pin pair relevant to each CxTRIP pin). The state of the PWM pins after PDPINTx is taken high depends on the state of the FCOMPOE bit.

Figure 6−21. External Interrupt Timing

6.18 General-Purpose Input/Output (GPIO) − Output Timing

Table 6−22. General-Purpose Output Switching Characteristics

PARAMETER			MIN	MAX	UNIT
$t_{d(XCOH-GPO)}$	Delay time, XCLKOUT high to GPIO low/high	All GPIOs		1 * $t_{c(SCO)}$	cycle
$t_{r(GPO)}$	Rise time, GPIO switching low to high	All GPIOs		10	ns
$t_{f(GPO)}$	Fall time, GPIO switching high to low	All GPIOs		10	ns
f_{GPO}	Toggling frequency, GPO pins			20	MHz

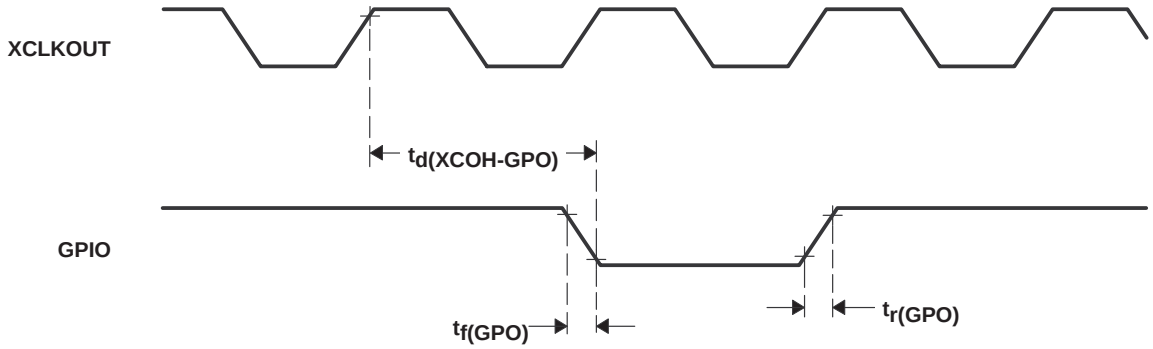
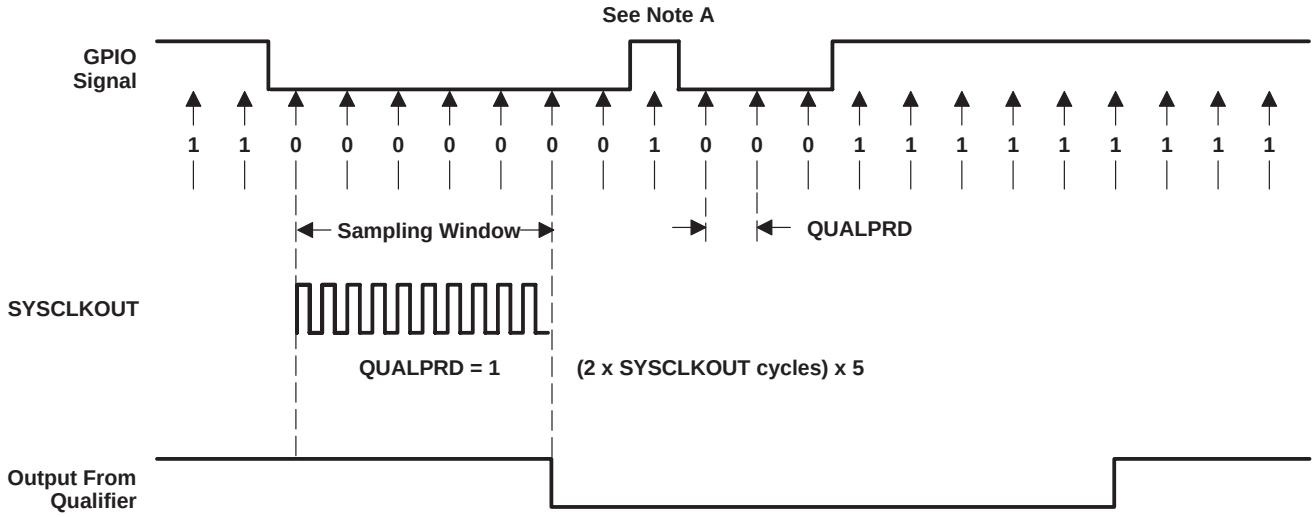


Figure 6−22. General-Purpose Output Timing

6.19 General-Purpose Input/Output (GPIO) − Input Timing



NOTES: A. This glitch will be ignored by the input qualifier. The QUALPRD bit field specifies the qualification sampling period. It can vary from 00 to 0xFF. Input qualification is not applicable when QUALPRD = 00. For any other value “n”, the qualification sampling period in 2n SYSCLKOUT cycles (i.e., at every 2n SYSCLKOUT cycle, the GPIO pin will be sampled). Six consecutive samples must be of the same value for a given input to be recognized.

B. For the qualifier to detect the change, the input should be stable for 10 SYSCLKOUT cycles or greater. In other words, the inputs should be stable for (5 x QUALPRD x 2) SYSCLKOUT cycles. This would ensure six sampling windows for detection to occur. Since external signals are driven asynchronously, an 11-SYSCLKOUT-wide pulse ensures reliable recognition.

Figure 6−23. GPIO Input Qualifier − Example Diagram for QUALPRD = 1

Table 6−23. General-Purpose Input Timing Requirements

				MIN MAX	UNIT
$t_{w(GPI)}$	Pulse duration, GPIO low/high	All GPIOs	With no qualifier	$2 * t_{c(SCO)}$	cycles
			With qualifier	$1 * t_{c(SCO)} + IQT^{\dagger}$	

† Input Qualification Time (IQT) = [5 x QUALPRD x 2] * $t_{c(SCO)}$

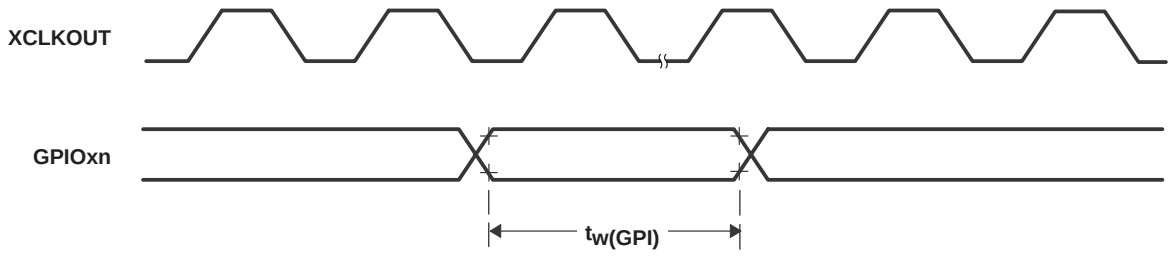


Figure 6−24. General-Purpose Input Timing

NOTE: The pulse width requirement for general-purpose input is applicable for the $\overline{\text{XBIO}}$ and ADCSOC pins as well.

6.20 SPI Master Mode Timing

Table 6–24. SPI Master Mode External Timing (Clock Phase = 0)†‡

NO.			SPI WHEN (SPIBRR + 1) IS EVEN OR SPIBRR = 0 OR 2		SPI WHEN (SPIBRR + 1) IS ODD AND SPIBRR > 3		UNIT
			MIN	MAX	MIN	MAX	
1	$t_{c(SPC)M}$	Cycle time, SPICLK	$4t_{c(LCO)}$	$128t_{c(LCO)}$	$5t_{c(LCO)}$	$127t_{c(LCO)}$	ns
2§	$t_{w(SPCH)M}$	Pulse duration, SPICLK high (clock polarity = 0)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}$	ns
	$t_{w(SPCL)M}$	Pulse duration, SPICLK low (clock polarity = 1)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}$	
3§	$t_{w(SPCL)M}$	Pulse duration, SPICLK low (clock polarity = 0)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}$	ns
	$t_{w(SPCH)M}$	Pulse duration, SPICLK high (clock polarity = 1)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}$	
4§	$t_{d(SPCH-SIMO)M}$	Delay time, SPICLK high to SPISIMO valid (clock polarity = 0)	−10	10	−10	10	ns
	$t_{d(SPCL-SIMO)M}$	Delay time, SPICLK low to SPISIMO valid (clock polarity = 1)	−10	10	−10	10	
5§	$t_{v(SPCL-SIMO)M}$	Valid time, SPISIMO data valid after SPICLK low (clock polarity = 0)	$0.5t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}+0.5t_{c(LCO)}-10$		ns
	$t_{v(SPCH-SIMO)M}$	Valid time, SPISIMO data valid after SPICLK high (clock polarity = 1)	$0.5t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}+0.5t_{c(LCO)}-10$		
8§	$t_{su(SOMI-SPCL)M}$	Setup time, SPISOMI before SPICLK low (clock polarity = 0)	0		0		ns
	$t_{su(SOMI-SPCH)M}$	Setup time, SPISOMI before SPICLK high (clock polarity = 1)	0		0		
9§	$t_{v(SPCL-SOMI)M}$	Valid time, SPISOMI data valid after SPICLK low (clock polarity = 0)	$0.25t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-0.5t_{c(LCO)}-10$		ns
	$t_{v(SPCH-SOMI)M}$	Valid time, SPISOMI data valid after SPICLK high (clock polarity = 1)	$0.25t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-0.5t_{c(LCO)}-10$		

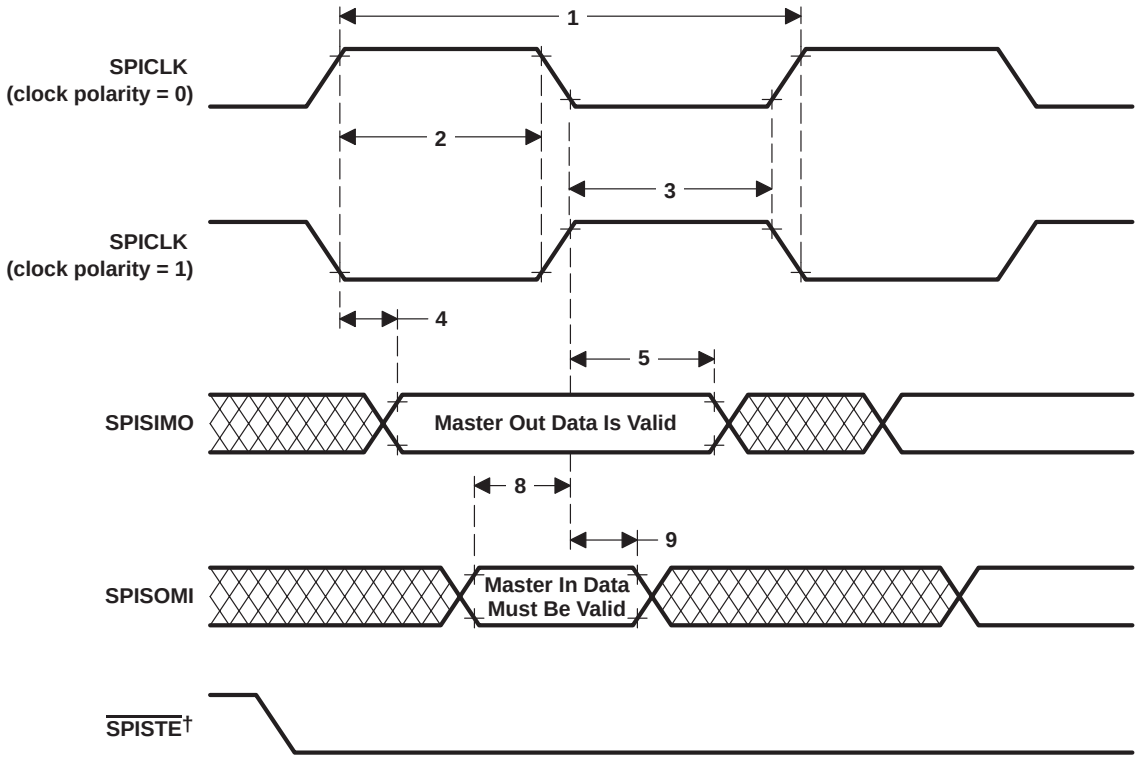
† The MASTER/SLAVE bit (SPICTL.2) is set and the CLOCK PHASE bit (SPICTL.3) is cleared.

‡ $t_{c(SPC)}$ = SPI clock cycle time = $\frac{\text{LSPCLK}}{4}$ or $\frac{\text{LSPCLK}}{(\text{SPIBRR} + 1)}$

$t_{c(LCO)}$ = LSPCLK cycle time

§ The active edge of the SPICLK signal referenced is controlled by the CLOCK POLARITY bit (SPICCR.6).

NOTE: Internal clock prescalers must be adjusted such that the SPI clock speed is not greater than the I/O buffer speed limit (20 MHz).



† In the master mode, $\overline{\text{SPISTE}}$ goes active $0.5t_{C(SPC)}$ before valid SPI clock edge. On the trailing end of the word, the SPISTE will go inactive $0.5t_{C(SPC)}$ after the receiving edge (SPICLK) of the last data bit.

Figure 6−25. SPI Master Mode External Timing (Clock Phase = 0)

Table 6–25. SPI Master Mode External Timing (Clock Phase = 1)†‡

NO.			SPI WHEN (SPIBRR + 1) IS EVEN OR SPIBRR = 0 OR 2		SPI WHEN (SPIBRR + 1) IS ODD AND SPIBRR > 3		UNIT
			MIN	MAX	MIN	MAX	
1	$t_{c(SPC)M}$	Cycle time, SPICLK	$4t_{c(LCO)}$	$128t_{c(LCO)}$	$5t_{c(LCO)}$	$127t_{c(LCO)}$	ns
2§	$t_{w(SPCH)M}$	Pulse duration, SPICLK high (clock polarity = 0)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}$	ns
	$t_{w(SPCL)M}$	Pulse duration, SPICLK low (clock polarity = 1)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}-0.5t_{c(LCO)}$	
3§	$t_{w(SPCL)M}$	Pulse duration, SPICLK low (clock polarity = 0)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}$	ns
	$t_{w(SPCH)M}$	Pulse duration, SPICLK high (clock polarity = 1)	$0.5t_{c(SPC)M}-10$	$0.5t_{c(SPC)M}$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}-10$	$0.5t_{c(SPC)M}+0.5t_{c(LCO)}$	
6§	$t_{su(SIMO-SPCH)M}$	Setup time, SPISIMO data valid before SPICLK high (clock polarity = 0)	$0.5t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-10$		ns
	$t_{su(SIMO-SPCL)M}$	Setup time, SPISIMO data valid before SPICLK low (clock polarity = 1)	$0.5t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-10$		
7§	$t_{v(SPCH-SIMO)M}$	Valid time, SPISIMO data valid after SPICLK high (clock polarity = 0)	$0.5t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-10$		ns
	$t_{v(SPCL-SIMO)M}$	Valid time, SPISIMO data valid after SPICLK low (clock polarity = 1)	$0.5t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-10$		
10§	$t_{su(SOMI-SPCH)M}$	Setup time, SPISOMI before SPICLK high (clock polarity = 0)	0		0		ns
	$t_{su(SOMI-SPCL)M}$	Setup time, SPISOMI before SPICLK low (clock polarity = 1)	0		0		
11§	$t_{v(SPCH-SOMI)M}$	Valid time, SPISOMI data valid after SPICLK high (clock polarity = 0)	$0.25t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-10$		ns
	$t_{v(SPCL-SOMI)M}$	Valid time, SPISOMI data valid after SPICLK low (clock polarity = 1)	$0.25t_{c(SPC)M}-10$		$0.5t_{c(SPC)M}-10$		

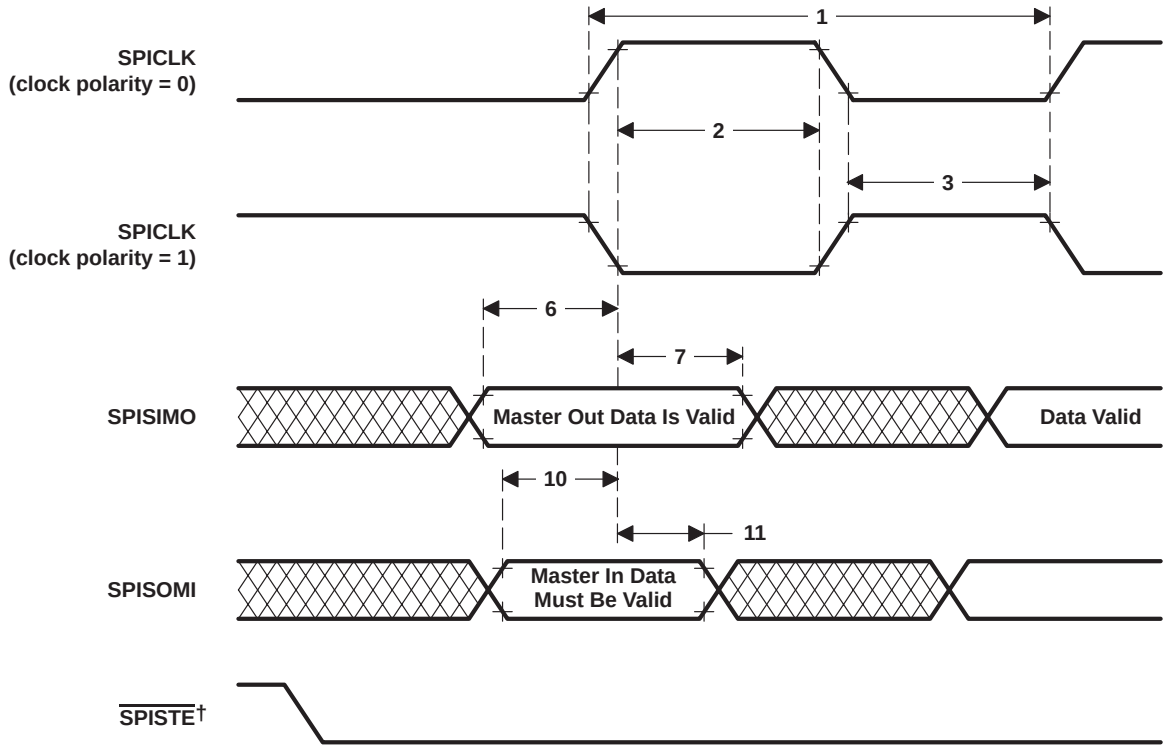
† The MASTER/SLAVE bit (SPICTL.2) is set and the CLOCK PHASE bit (SPICTL.3) is set.

‡ $t_{c(SPC)}$ = SPI clock cycle time = $\frac{LSPCLK}{4}$ or $\frac{LSPCLK}{(SPIBRR + 1)}$

$t_{c(LCO)}$ = LSPCLK cycle time

§ The active edge of the SPICLK signal referenced is controlled by the CLOCK POLARITY bit (SPICCR.6).

NOTE: Internal clock prescalers must be adjusted such that the SPI clock speed is not greater than the I/O buffer speed limit (20 MHz).



† In the master mode, $\overline{\text{SPISTE}}$ goes active $0.5t_{c(SPC)}$ before valid SPI clock edge. On the trailing end of the word, the SPISTE will go inactive $0.5t_{c(SPC)}$ after the receiving edge (SPICLK) of the last data bit.

Figure 6−26. SPI Master External Timing (Clock Phase = 1)

6.21 SPI Slave Mode Timing

Table 6−26. SPI Slave Mode External Timing (Clock Phase = 0)†‡

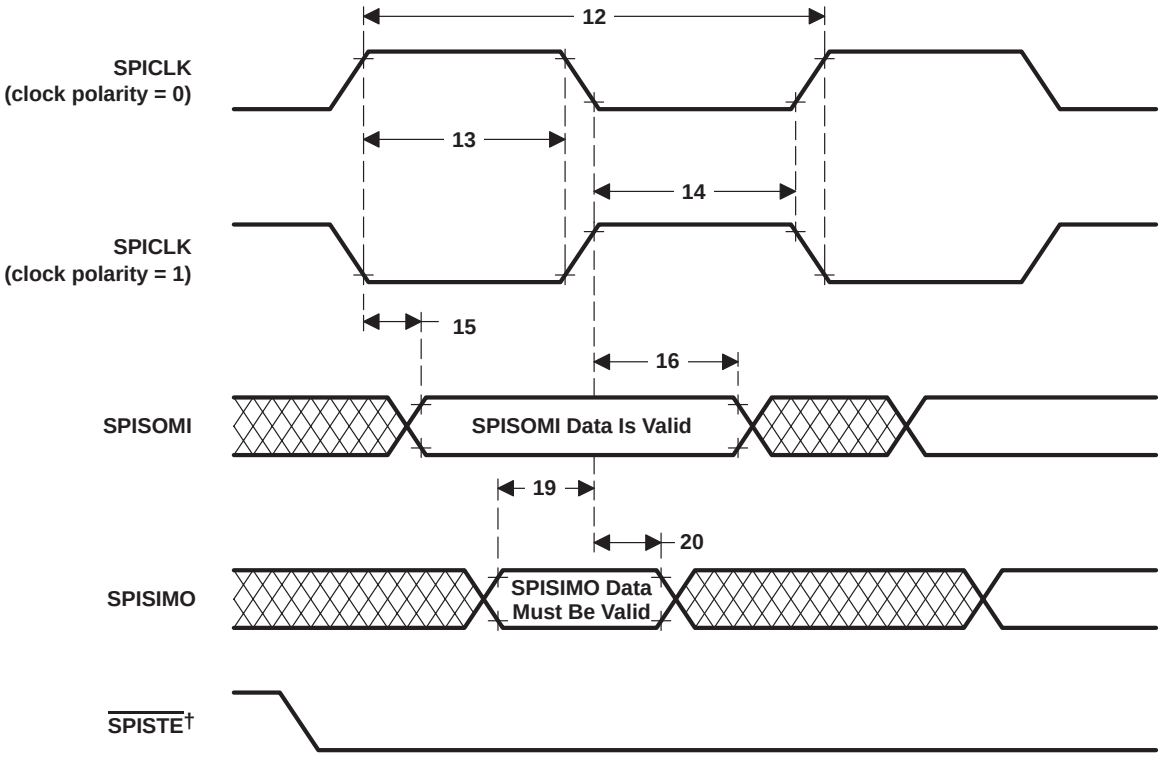
NO.			MIN	MAX	UNIT
12	$t_{c(SPC)S}$	Cycle time, SPICLK	$4t_{c(LCO)}$‡		ns
13§	$t_{w(SPCH)S}$	Pulse duration, SPICLK high (clock polarity = 0)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	ns
	$t_{w(SPCL)S}$	Pulse duration, SPICLK low (clock polarity = 1)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	
14§	$t_{w(SPCL)S}$	Pulse duration, SPICLK low (clock polarity = 0)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	ns
	$t_{w(SPCH)S}$	Pulse duration, SPICLK high (clock polarity = 1)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	
15§	$t_{d(SPCH-SOMI)S}$	Delay time, SPICLK high to SPISOMI valid (clock polarity = 0)	$0.375t_{c(SPC)S}-10$		ns
	$t_{d(SPCL-SOMI)S}$	Delay time, SPICLK low to SPISOMI valid (clock polarity = 1)	$0.375t_{c(SPC)S}-10$		
16§	$t_{v(SPCL-SOMI)S}$	Valid time, SPISOMI data valid after SPICLK low (clock polarity =0)	$0.75t_{c(SPC)S}$		ns
	$t_{v(SPCH-SOMI)S}$	Valid time, SPISOMI data valid after SPICLK high (clock polarity =1)	$0.75t_{c(SPC)S}$		
19§	$t_{su(SIMO-SPCL)S}$	Setup time, SPISIMO before SPICLK low (clock polarity = 0)	0		ns
	$t_{su(SIMO-SPCH)S}$	Setup time, SPISIMO before SPICLK high (clock polarity = 1)	0		
20§	$t_{v(SPCL-SIMO)S}$	Valid time, SPISIMO data valid after SPICLK low (clock polarity = 0)	$0.5t_{c(SPC)S}$		ns
	$t_{v(SPCH-SIMO)S}$	Valid time, SPISIMO data valid after SPICLK high (clock polarity = 1)	$0.5t_{c(SPC)S}$		

† The MASTER / SLAVE bit (SPICTL.2) is cleared and the CLOCK PHASE bit (SPICTL.3) is cleared.

‡ $t_{c(SPC)}$ = SPI clock cycle time = $\frac{LSPCLK}{4}$ or $\frac{LSPCLK}{(SPIBRR + 1)}$

$t_{c(LCO)}$ = LSPCLK cycle time

§ The active edge of the SPICLK signal referenced is controlled by the CLOCK POLARITY bit (SPICCR.6).



† In the slave mode, the $\overline{\text{SPISTE}}$ signal should be asserted low at least $0.5t_{c(SPC)}$ before the valid SPI clock edge and remain low for at least $0.5t_{c(SPC)}$ after the receiving edge (SPICLK) of the last data bit.

Figure 6−27. SPI Slave Mode External Timing (Clock Phase = 0)

Table 6−27. SPI Slave Mode External Timing (Clock Phase = 1)†‡

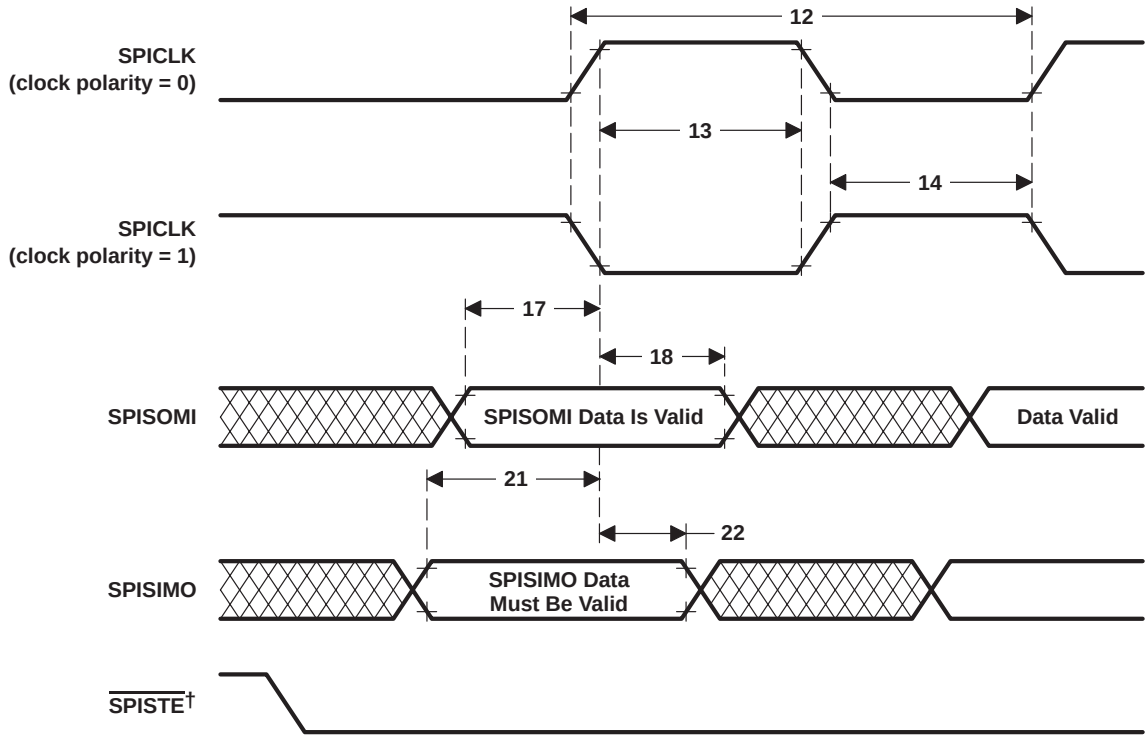
NO.			MIN	MAX	UNIT
12	$t_{c(SPC)S}$	Cycle time, SPICLK	$8t_{c(LCO)}$		ns
13§	$t_{w(SPCH)S}$	Pulse duration, SPICLK high (clock polarity = 0)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	ns
	$t_{w(SPCL)S}$	Pulse duration, SPICLK low (clock polarity = 1)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	
14§	$t_{w(SPCL)S}$	Pulse duration, SPICLK low (clock polarity = 0)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	ns
	$t_{w(SPCH)S}$	Pulse duration, SPICLK high (clock polarity = 1)	$0.5t_{c(SPC)S}-10$	$0.5t_{c(SPC)S}$	
17§	$t_{su(SOMI-SPCH)S}$	Setup time, SPISOMI before SPICLK high (clock polarity = 0)	$0.125t_{c(SPC)S}$		ns
	$t_{su(SOMI-SPCL)S}$	Setup time, SPISOMI before SPICLK low (clock polarity = 1)	$0.125t_{c(SPC)S}$		
18§	$t_{v(SPCH-SOMI)S}$	Valid time, SPISOMI data valid after SPICLK high (clock polarity =0)	$0.75t_{c(SPC)S}$		ns
	$t_{v(SPCL-SOMI)S}$	Valid time, SPISOMI data valid after SPICLK low (clock polarity =1)	$0.75t_{c(SPC)S}$		
21§	$t_{su(SIMO-SPCH)S}$	Setup time, SPISIMO before SPICLK high (clock polarity = 0)	0		ns
	$t_{su(SIMO-SPCL)S}$	Setup time, SPISIMO before SPICLK low (clock polarity = 1)	0		
22§	$t_{v(SPCH-SIMO)S}$	Valid time, SPISIMO data valid after SPICLK high (clock polarity = 0)	$0.5t_{c(SPC)S}$		ns
	$t_{v(SPCL-SIMO)S}$	Valid time, SPISIMO data valid after SPICLK low (clock polarity = 1)	$0.5t_{c(SPC)S}$		

† The MASTER / SLAVE bit (SPICTL.2) is cleared and the CLOCK PHASE bit (SPICTL.3) is set.

‡ $t_{c(SPC)}$ = SPI clock cycle time = $\frac{LSPCLK}{4}$ or $\frac{LSPCLK}{(SPIBRR + 1)}$

$t_{c(LCO)}$ = LSPCLK cycle time

§ The active edge of the SPICLK signal referenced is controlled by the CLOCK POLARITY bit (SPICCR.6).



† In the slave mode, the $\overline{SPISTE}$ signal should be asserted low at least $0.5t_{c(SPC)}$ before the valid SPI clock edge and remain low for at least $0.5t_{c(SPC)}$ after the receiving edge (SPICLK) of the last data bit.

Figure 6−28. SPI Slave Mode External Timing (Clock Phase = 1)

6.22 External Interface (XINTF) Timing

Each XINTF access consists of three parts: Lead, Active, and Trail. The user configures the Lead/Active/Trail wait states in the XTIMING registers. There is one XTIMING register for each XINTF zone. Table 6−28 shows the relationship between the parameters configured in the XTIMING register and the duration of the pulse in terms of XTIMCLK cycles.

Table 6−28. Relationship Between Parameters Configured in XTIMING and Duration of Pulse†‡

DESCRIPTION		DURATION (ns)	
		X2TIMING = 0	X2TIMING = 1
LR	Lead period, read access	XRDLEAD x $t_{c(XTIM)}$	(XRDLEAD x 2) x $t_{c(XTIM)}$
AR	Active period, read access	(XRDACTIVE + WS + 1) x $t_{c(XTIM)}$	(XRDACTIVE x 2 + WS + 1) x $t_{c(XTIM)}$
TR	Trail period, read access	XRDTRAIL x $t_{c(XTIM)}$	(XRDTRAIL x 2) x $t_{c(XTIM)}$
LW	Lead period, write access	XWRLEAD x $t_{c(XTIM)}$	(XWRLEAD x 2) x $t_{c(XTIM)}$
AW	Active period, write access	(XWRACTIVE + WS + 1) x $t_{c(XTIM)}$	(XWRACTIVE x 2 + WS + 1) x $t_{c(XTIM)}$
TW	Trail period, write access	XWRTRAIL x $t_{c(XTIM)}$	(XWRTRAIL x 2) x $t_{c(XTIM)}$

† $t_{c(XTIM)}$ − Cycle time, XTIMCLK
‡ WS refers to the number of wait states inserted by hardware when using XREADY. If the zone is configured to ignore XREADY (USEREADY = 0), then WS = 0.

Minimum wait state requirements must be met when configuring each zone's XTIMING register. These requirements are in addition to any timing requirements as specified by that device's data sheet. No internal device hardware is included to detect illegal settings.

- If the XREADY signal is ignored (USEREADY = 0), then:

  1. Lead: $LR \geq t_{c(XTIM)}$
     $LW \geq t_{c(XTIM)}$

  These requirements result in the following XTIMING register configuration restrictions§:

XRDLEAD	XRDACTIVE	XRDTRAIL	XWRLEAD	XWRACTIVE	XWRTRAIL	X2TIMING
≥ 1	≥ 0	≥ 0	≥ 1	≥ 0	≥ 0	0, 1

§ No hardware to detect illegal XTIMING configurations

Examples of valid and invalid timing when not sampling XREADY§:

	XRDLEAD	XRDACTIVE	XRDTRAIL	XWRLEAD	XWRACTIVE	XWRTRAIL	X2TIMING
Invalid	0	0	0	0	0	0	0, 1
Valid	1	0	0	1	0	0	0, 1

§ No hardware to detect illegal XTIMING configurations

- If the XREADY signal is sampled in the Synchronous mode (USEREADY = 1, READYMODE = 0), then:

  1. Lead: $LR \geq t_{c(XTIM)}$
     $LW \geq t_{c(XTIM)}$
  2. Active: $AR \geq 2 \times t_{c(XTIM)}$
     $AW \geq 2 \times t_{c(XTIM)}$

     NOTE: Restriction does not include external hardware wait states

  These requirements result in the following XTIMING register configuration restrictions†:

XRDLEAD	XRDACTIVE	XRDTRAIL	XWRLEAD	XWRACTIVE	XWRTRAIL	X2TIMING
≥ 1	≥ 1	≥ 0	≥ 1	≥ 1	≥ 0	0, 1

† No hardware to detect illegal XTIMING configurations

Examples of valid and invalid timing when using Synchronous XREADY†:

	XRDLEAD	XRDACTIVE	XRDTRAIL	XWRLEAD	XWRACTIVE	XWRTRAIL	X2TIMING
Invalid	0	0	0	0	0	0	0, 1
Invalid	1	0	0	1	0	0	0, 1
Valid	1	1	0	1	1	0	0, 1

† No hardware to detect illegal XTIMING configurations

- If the XREADY signal is sampled in the Asynchronous mode (USEREADY = 1, READYMODE = 1), then:

  1. Lead: $LR \geq t_{c(XTIM)}$
     $LW \geq t_{c(XTIM)}$
  2. Active: $AR \geq 2 \times t_{c(XTIM)}$
     $AW \geq 2 \times t_{c(XTIM)}$

     NOTE: Restriction does not include external hardware wait states
  3. Lead + Active: $LR + AR \geq 4 \times t_{c(XTIM)}$
     $LW + AW \geq 4 \times t_{c(XTIM)}$

     NOTE: Restriction does not include external hardware wait states

  These requirements result in the following XTIMING register configuration restrictions†:

XRDLEAD	XRDACTIVE	XRDTRAIL	XWRLEAD	XWRACTIVE	XWRTRAIL	X2TIMING
≥ 1	≥ 2	0	≥ 1	≥ 2	0	0, 1

† No hardware to detect illegal XTIMING configurations

or†

XRDLEAD	XRDACTIVE	XRDTRAIL	XWRLEAD	XWRACTIVE	XWRTRAIL	X2TIMING
≥ 2	≥ 1	0	≥ 2	≥ 1	0	0, 1

† No hardware to detect illegal XTIMING configurations

Examples of valid and invalid timing when using Asynchronous XREADY†:

	XRDLEAD	XRDACTIVE	XRDTRAIL	XWRLEAD	XWRACTIVE	XWRTRAIL	X2TIMING
Invalid	0	0	0	0	0	0	0, 1
Invalid	1	0	0	1	0	0	0, 1
Invalid	1	1	0	1	1	0	0
Valid	1	1	0	1	1	0	1
Valid	1	2	0	1	2	0	0, 1
Valid	2	1	0	2	1	0	0, 1

† No hardware to detect illegal XTIMING configurations

Unless otherwise specified, all XINTF timing is applicable for the clock configurations shown in Table 6−29.

Table 6−29. XINTF Clock Configurations

MODE	SYSCLKOUT	XTIMCLK	XCLKOUT
1 Example:	 150 MHz	SYSCLKOUT 150 MHz	SYSCLKOUT 150 MHz
2 Example:	 150 MHz	SYSCLKOUT 150 MHz	1/2 SYSCLKOUT 75 MHz
3 Example:	 150 MHz	1/2 SYSCLKOUT 75 MHz	1/2 SYSCLKOUT 75 MHz
4 Example:	 150 MHz	1/2 SYSCLKOUT 75 MHz	1/4 SYSCLKOUT 37.5 MHz

The relationship between SYSCLKOUT and XTIMCLK is shown in Figure 6−29.

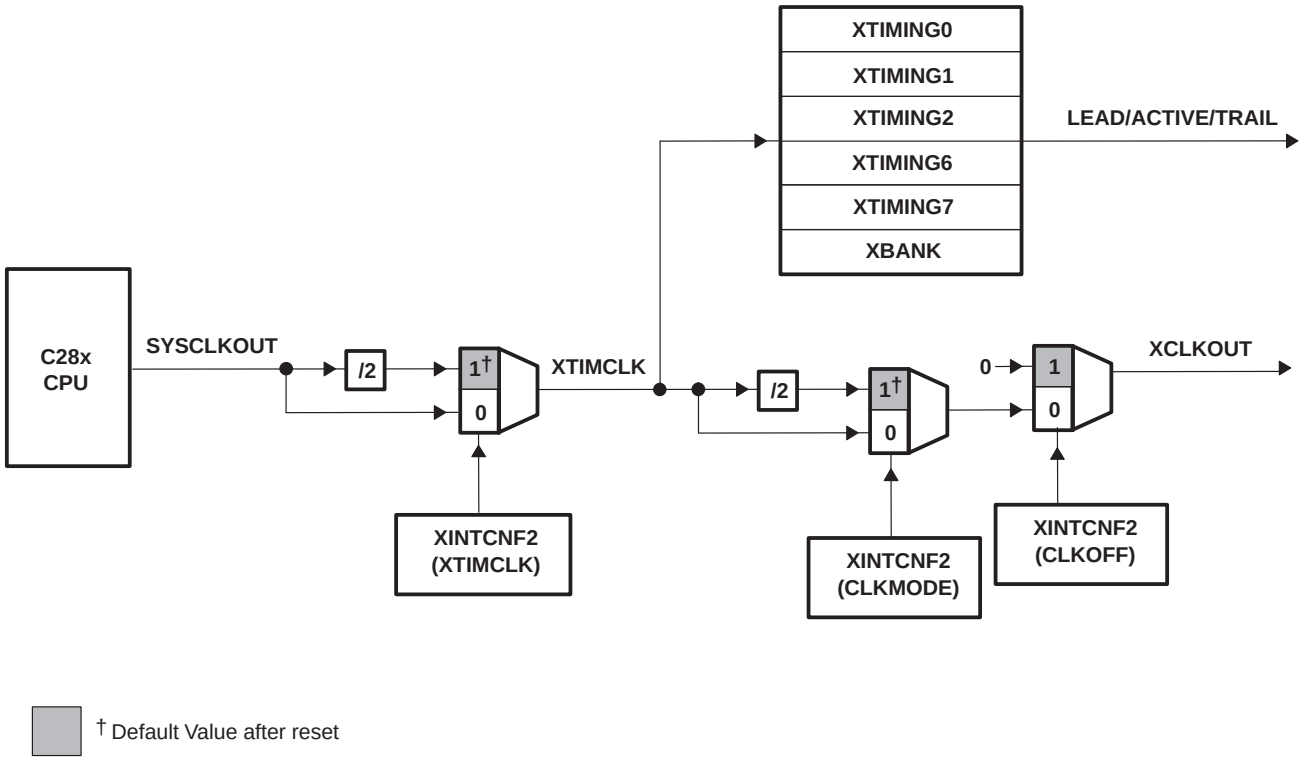


Figure 6−29. Relationship Between XTIMCLK and SYSCLKOUT

6.23 XINTF Signal Alignment to XCLKOUT

For each XINTF access, the number of lead, active, and trail cycles is based on the internal clock XTIMCLK. Strobes such as $\overline{\text{XRD}}$, $\overline{\text{XWE}}$, and zone chip-select (XZCS) change state in relationship to the rising edge of XTIMCLK. The external clock, XCLKOUT, can be configured to be either equal to or one-half the frequency of XTIMCLK.

For the case where XCLKOUT = XTIMCLK, all of the XINTF strobes will change state with respect to the rising edge of XCLKOUT. For the case where XCLKOUT = one-half XTIMCLK, some strobes will change state either on the rising edge of XCLKOUT or the falling edge of XCLKOUT. In the XINTF timing tables, the notation XCOHL is used to indicate that the parameter is with respect to either case; XCLKOUT rising edge (high) or XCLKOUT falling edge (low). If the parameter is always with respect to the rising edge of XCLKOUT, the notation XCOH is used.

For the case where XCLKOUT = one-half XTIMCLK, the XCLKOUT edge with which the change will be aligned can be determined based on the number of XTIMCLK cycles from the start of the access to the point at which the signal changes. If this number of XTIMCLK cycles is even, the alignment will be with respect to the rising edge of XCLKOUT. If this number is odd, then the signal will change with respect to the falling edge of XCLKOUT. Examples include the following:

- Strobes that change at the beginning of an access always align to the rising edge of XCLKOUT. This is because all XINTF accesses begin with respect to the rising edge of XCLKOUT.

Examples:	XZCSL	Zone chip-select active low
	XRNWL	XR/$\overline{W}$ active low

- Strobes that change at the beginning of the active period will align to the rising edge of XCLKOUT if the total number of lead XTIMCLK cycles for the access is even. If the number of lead XTIMCLK cycles is odd, then the alignment will be with respect to the falling edge of XCLKOUT.

Examples:	XRDL	$\overline{XRD}$ active low
	XWEL	$\overline{XWE}$ active low

- Strobes that change at the beginning of the trail period will align to the rising edge of XCLKOUT if the total number of lead + active XTIMCLK cycles (including hardware waitstates) for the access is even. If the number of lead + active XTIMCLK cycles (including hardware waitstates) is odd, then the alignment will be with respect to the falling edge of XCLKOUT.

Examples:	XRDH	$\overline{XRD}$ inactive high
	XWEH	$\overline{XWE}$ inactive high

- Strobes that change at the end of the access will align to the rising edge of XCLKOUT if the total number of lead + active + trail XTIMCLK cycles (including hardware waitstates) is even. If the number of lead + active + trail XTIMCLK cycles (including hardware waitstates) is odd, then the alignment will be with respect to the falling edge of XCLKOUT.

Examples:	XZCSH	Zone chip-select inactive high
	XRNWH	XR/$\overline{W}$ inactive high

6.24 External Interface Read Timing

Table 6−30. External Memory Interface Read Switching Characteristics

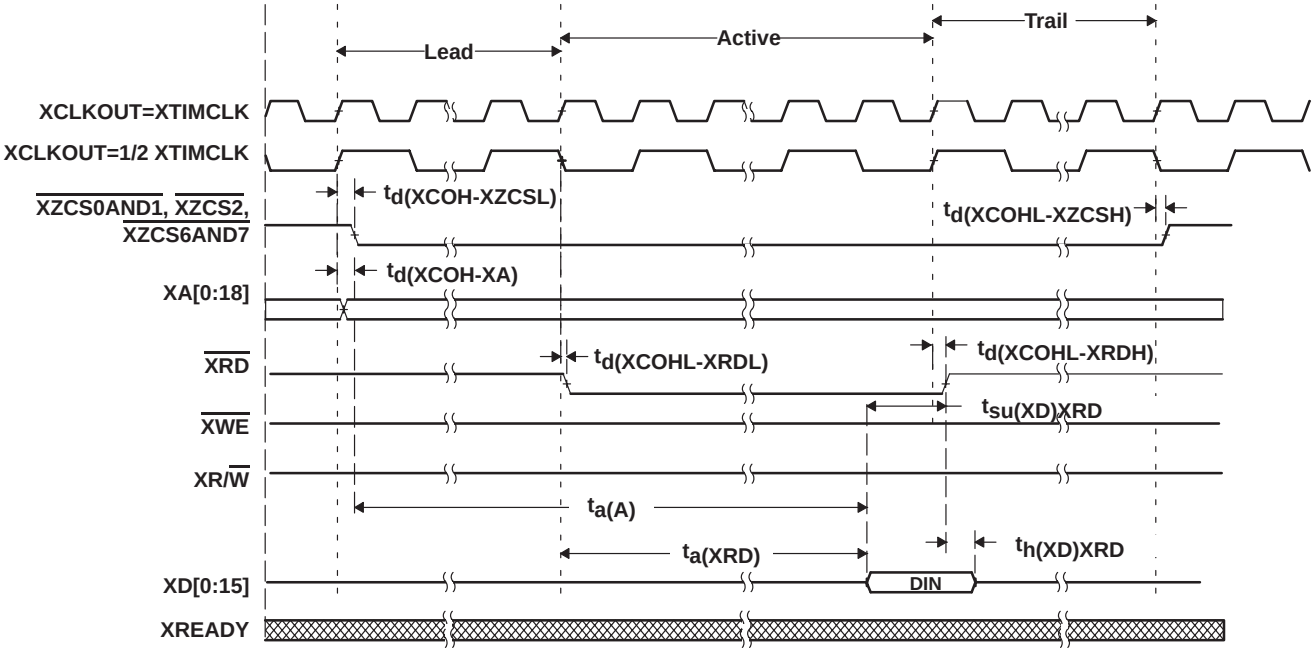
PARAMETER		MIN	MAX	UNIT
$t_{d(XCOH-XZCSL)}$	Delay time, XCLKOUT high to zone chip-select active low		1	ns
$t_{d(XCOHL-XZCSH)}$	Delay time, XCLKOUT high/low to zone chip-select inactive high	−2	3	ns
$t_{d(XCOH-XA)}$	Delay time, XCLKOUT high to address valid		2	ns
$t_{d(XCOHL-XRDL)}$	Delay time, XCLKOUT high/low to $\overline{XRD}$ active low		1	ns
$t_{d(XCOHL-XRDH}$	Delay time, XCLKOUT high/low to $\overline{XRD}$ inactive high	−2	1	ns
$t_{h(XA)XZCSH}$	Hold time, address valid after zone chip-select inactive high	†		ns
$t_{h(XA)XRD}$	Hold time, address valid after $\overline{XRD}$ inactive high	†		ns

† During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.

Table 6−31. External Memory Interface Read Timing Requirements

		MIN	MAX	UNIT
$t_{a(A)}$	Access time, read data from address valid		(LR + AR) − 14‡	ns
$t_{a(XRD)}$	Access time, read data valid from $\overline{XRD}$ active low		AR − 12‡	ns
$t_{su(XD)XRD}$	Setup time, read data valid before $\overline{XRD}$ strobe inactive high	12		ns
$t_{h(XD)XRD}$	Hold time, read data valid after $\overline{XRD}$ inactive high	0		ns

‡ LR = Lead period, read access. AR = Active period, read access. See Table 6−28.



NOTES: A. All XINTF accesses (lead period) begin on the rising edge of XCLKOUT. When necessary, the device will insert an alignment cycle before an access to meet this requirement.
B. During alignment cycles, all signals will transition to their inactive state.
C. For USEREADY = 0, the external XREADY input signal is ignored.
D. XA[0:18] will hold the last address put on the bus during inactive cycles, including alignment cycles.

Figure 6−30. Example Read Access

XTIMING register parameters used for this example:

XRDLEAD	XRDACTIVE	XRDTRAIL	USEREADY	X2TIMING	XWRLEAD	XWRACTIVE	XWRTRAIL	READYMODE
≥1	≥0	≥0	0	0	N/A†	N/A†	N/A†	N/A†

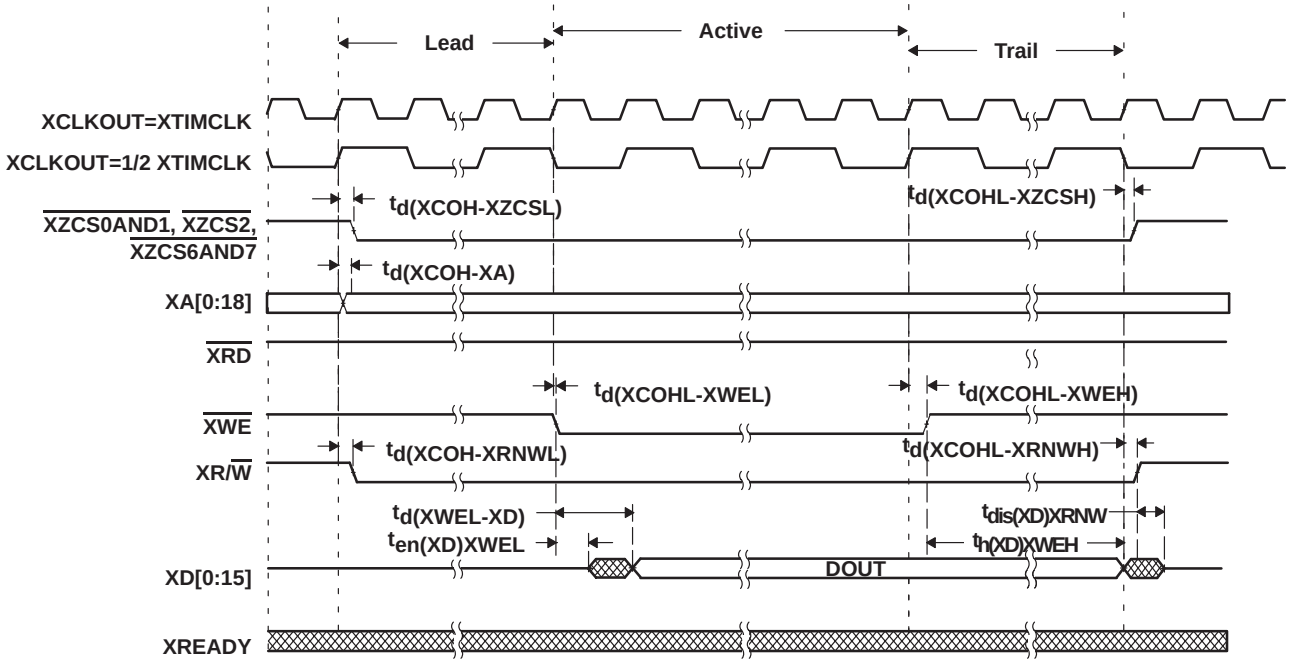
† N/A = “Don’t care” for this example

6.25 External Interface Write Timing

Table 6−32. External Memory Interface Write Switching Characteristics

PARAMETER		MIN	MAX	UNIT
$t_{d(XCOH-XZCSL)}$	Delay time, XCLKOUT high to zone chip-select active low		1	ns
$t_{d(XCOHL-XZCSH)}$	Delay time, XCLKOUT high or low to zone chip-select inactive high	−2	3	ns
$t_{d(XCOH-XA)}$	Delay time, XCLKOUT high to address valid		2	ns
$t_{d(XCOHL-XWEL)}$	Delay time, XCLKOUT high/low to $\overline{XWE}$ low		2	ns
$t_{d(XCOHL-XWEH)}$	Delay time, XCLKOUT high/low to $\overline{XWE}$ high		2	ns
$t_{d(XCOH-XRNWL)}$	Delay time, XCLKOUT high to XR/$\overline{W}$ low		1	ns
$t_{d(XCOHL-XRNWH)}$	Delay time, XCLKOUT high/low to XR/$\overline{W}$ high	−2	1	ns
$t_{en(XD)XWEL}$	Enable time, data bus driven from $\overline{XWE}$ low	0		ns
$t_{d(XWEL-XD)}$	Delay time, data valid after $\overline{XWE}$ active low		4	ns
$t_{h(XA)XZCSH}$	Hold time, address valid after zone chip-select inactive high	†		ns
$t_{h(XD)XWE}$	Hold time, write data valid after $\overline{XWE}$ inactive high	TW−2‡		ns
$t_{dis(XD)XRNW}$	Data bus disabled after XR/$\overline{W}$ inactive high	4		ns

† During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.
‡ TW = Trail period, write access. See Table 6−28.



NOTES: A. All XINTF accesses (lead period) begin on the rising edge of XCLKOUT. When necessary, the device will insert an alignment cycle before an access to meet this requirement.
B. During alignment cycles, all signals will transition to their inactive state.
C. For USEREADY = 0, the external XREADY input signal is ignored.
D. XA[0:18] will hold the last address put on the bus during inactive cycles, including alignment cycles.

Figure 6−31. Example Write Access

XTIMING register parameters used for this example:

XRDLEAD	XRDACTIVE	XRDTRAIL	USEREADY	X2TIMING	XWRLEAD	XWRACTIVE	XWRTRAIL	READYMODE
N/A†	N/A†	N/A†	0	0	≥1	≥0	≥0	N/A†

† N/A = “Don’t care” for this example

6.26 External Interface Ready-on-Read Timing With One External Wait State

Table 6−33. External Memory Interface Read Switching Characteristics (Ready-on-Read, 1 Wait State)

PARAMETER		MIN	MAX	UNIT
$t_{d(XCOH-XZCSL)}$	Delay time, XCLKOUT high to zone chip-select active low		1	ns
$t_{d(XCOHL-XZCSH)}$	Delay time, XCLKOUT high/low to zone chip-select inactive high	−2	3	ns
$t_{d(XCOH-XA)}$	Delay time, XCLKOUT high to address valid		2	ns
$t_{d(XCOHL-XRDL)}$	Delay time, XCLKOUT high/low to $\overline{XRD}$ active low		1	ns
$t_{d(XCOHL-XRDH}$	Delay time, XCLKOUT high/low to $\overline{XRD}$ inactive high	−2	1	ns
$t_{h(XA)XZCSH}$	Hold time, address valid after zone chip-select inactive high	†		ns
$t_{h(XA)XRD}$	Hold time, address valid after $\overline{XRD}$ inactive high	†		ns

† During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.

Table 6−34. External Memory Interface Read Timing Requirements (Ready-on-Read, 1 Wait State)

		MIN	MAX	UNIT
$t_{a(A)}$	Access time, read data from address valid		(LR + AR) − 14‡	ns
$t_{a(XRD)}$	Access time, read data valid from $\overline{XRD}$ active low		AR − 12‡	ns
$t_{su(XD)XRD}$	Setup time, read data valid before $\overline{XRD}$ strobe inactive high	12		ns
$t_{h(XD)XRD}$	Hold time, read data valid after $\overline{XRD}$ inactive high	0		ns

‡ LR = Lead period, read access. AR = Active period, read access. See Table 6−28.

Table 6−35. Synchronous XREADY Timing Requirements (Ready-on-Read, 1 Wait State)§

		MIN	MAX	UNIT
$t_{su(XRDYsynchL)XCOHL}$	Setup time, XREADY (Synch) low before XCLKOUT high/low	15		ns
$t_{h(XRDYsynchL)}$	Hold time, XREADY (Synch) low	12		ns
$t_{e(XRDYsynchH)}$	Earliest time XREADY (Synch) can go high before the sampling XCLKOUT edge		3	ns
$t_{su(XRDYsynchH)XCOHL}$	Setup time, XREADY (Synch) high before XCLKOUT high/low	15		ns
$t_{h(XRDYsynchH)XZCSH}$	Hold time, XREADY (Synch) held high after zone chip select high	0		ns

§ The first XREADY (Synch) sample occurs with respect to E in Figure 6−32:

$E = (XRDLEAD + XRDACTIVE)\ t_{c(XTIM)}$

When first sampled, if XREADY (Synch) is found to be high, then the access will complete. If XREADY (Synch) is found to be low, it will be sampled again each $t_{c(XTIM)}$ until it is found to be high.

For each sample (n) the setup time (D) with respect to the beginning of the access can be calculated as:

$D = (XRDLEAD + XRDACTIVE + n - 1)\ t_{c(XTIM)} - t_{su(XRDYsynchL)XCOHL}$

where n is the sample number: n = 1, 2, 3, and so forth.

Table 6−36. Asynchronous XREADY Timing Requirements (Ready-on-Read, 1 Wait State)¶

		MIN	MAX	UNIT
$t_{su(XRDYAsynchL)XCOHL}$	Setup time, XREADY (Async) low before XCLKOUT high/low	11		ns
$t_{h(XRDYAsynchL)}$	Hold time, XREADY (Async) low	8		ns
$t_{e(XRDYAsynchH)}$	Earliest time XREADY (Async) can go high before the sampling XCLKOUT edge		3	ns
$t_{su(XRDYAsynchH)XCOHL}$	Setup time, XREADY (Async) high before XCLKOUT high/low	11		ns
$t_{h(XRDYasynchH)XZCSH}$	Hold time, XREADY (Async) held high after zone chip select high	0		ns

¶ The first XREADY (Async) sample occurs with respect to E in Figure 6−33:

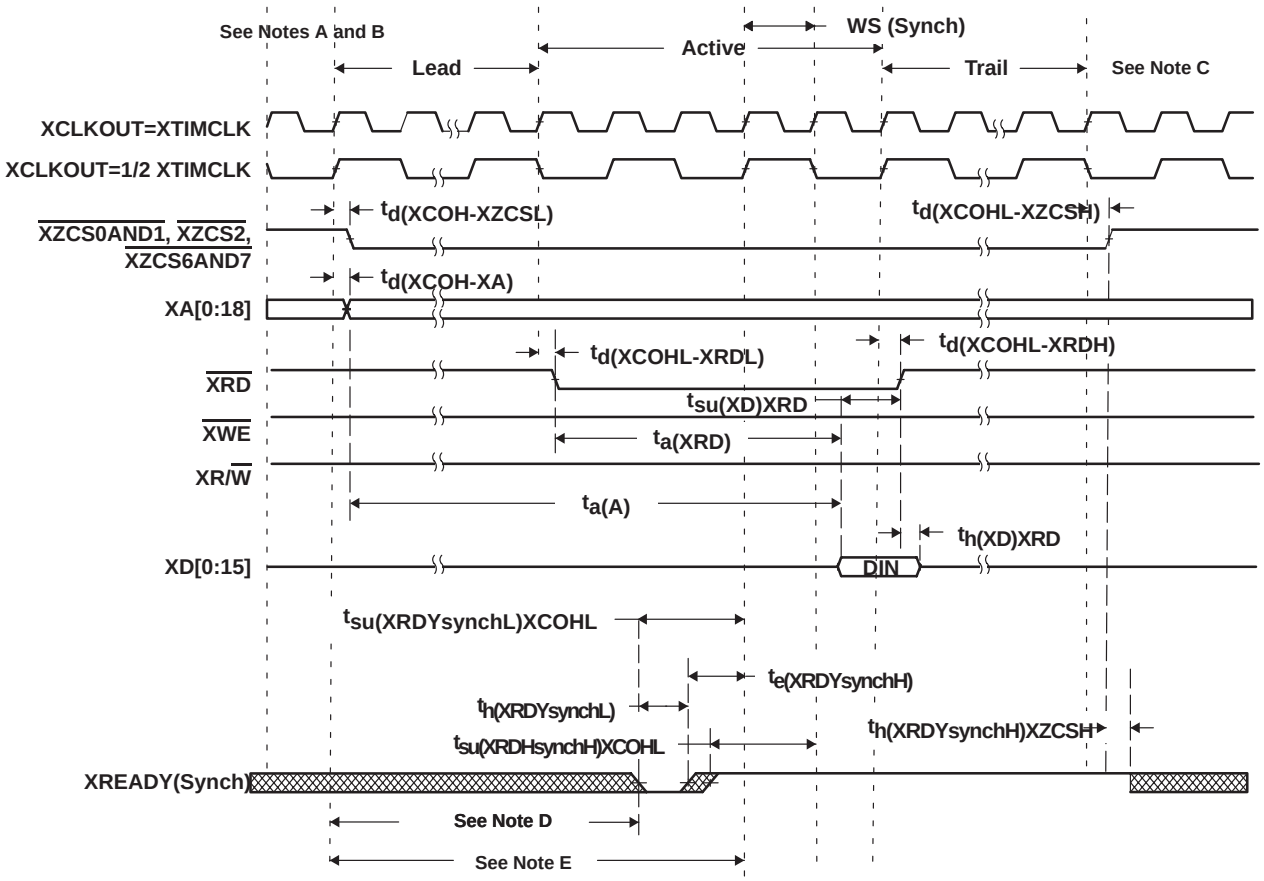
$E = (XRDLEAD + XRDACTIVE - 2)\ t_{c(XTIM)}$

When first sampled, if XREADY (Async) is found to be high, then the access will complete. If XREADY (Async) is found to be low, it will be sampled again each $t_{c(XTIM)}$ until it is found to be high.

For each sample, setup time from the beginning of the access can be calculated as:

$D = (XRDLEAD + XRDACTIVE - 3 + n)\ t_{c(XTIM)} - t_{su(XRDYasynchL)XCOHL}$

where n is the sample number: n = 1, 2, 3, and so forth.



Legend:

▨ = Don't care. Signal can be high or low during this time.

NOTES: A. All XINTF accesses (lead period) begin on the rising edge of XCLKOUT. When necessary, the device will insert an alignment cycle before an access to meet this requirement.

B. During alignment cycles, all signals will transition to their inactive state.

C. During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.

D. For each sample, setup time from the beginning of the access (D) can be calculated as:
$D = (XRDLEAD + XRDACTIVE + n - 1)\ t_{c(XTIM)} - t_{su(XRDYsynchL)XCOHL}$

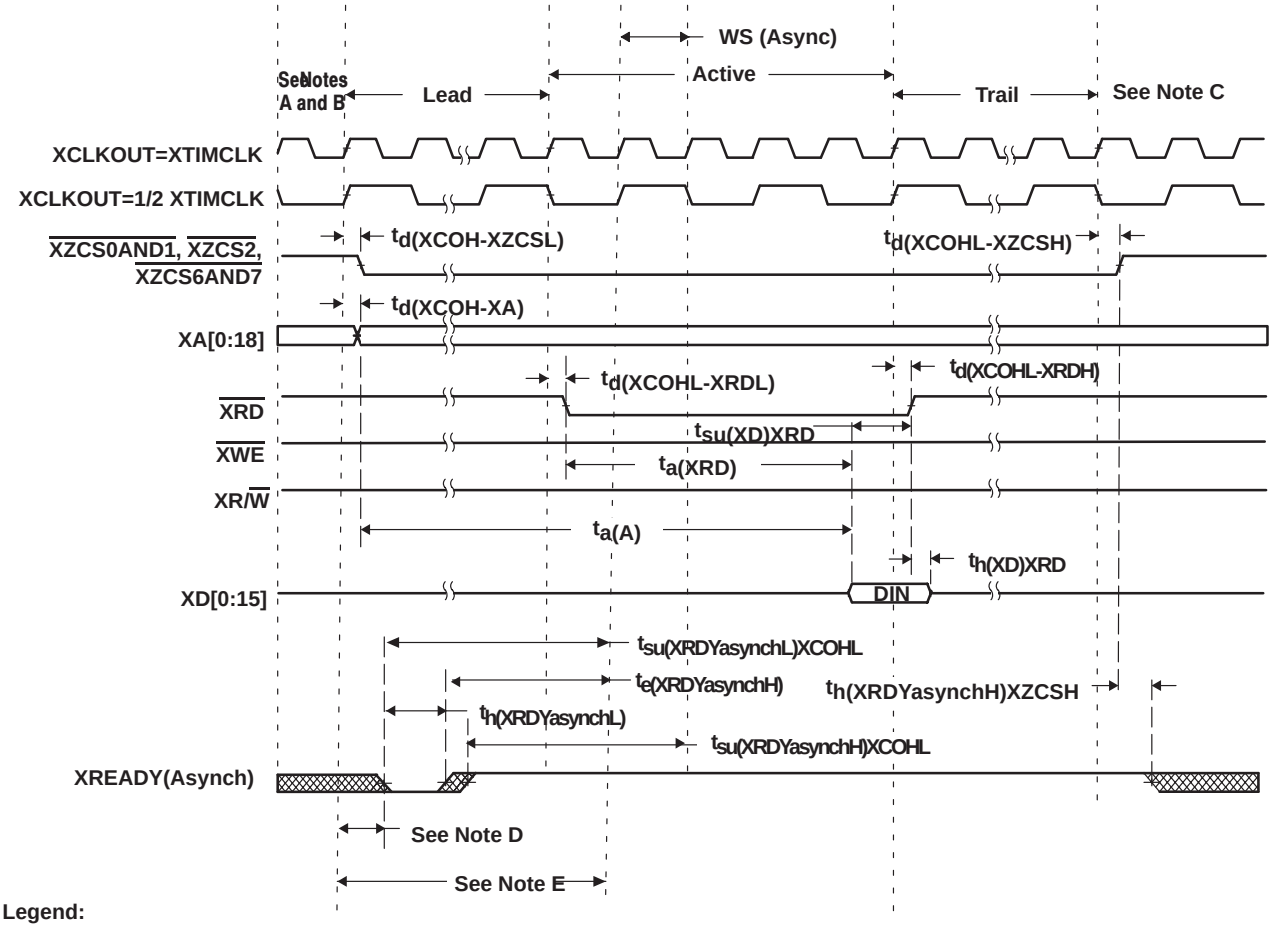
E. Reference for the first sample is with respect to this point
$E = (XRDLEAD + XRDACTIVE)\ t_{c(XTIM)}$
where n is the sample number: n = 1, 2, 3, and so forth.

Figure 6−32. Example Read With Synchronous XREADY Access

XTIMING register parameters used for this example:

XRDLEAD	XRDACTIVE	XRDTRAIL	USEREADY	X2TIMING	XWRLEAD	XWRACTIVE	XWRTRAIL	READYMODE
≥1	3	≥1	1	0	N/A†	N/A†	N/A†	0 = XREADY (Synch)

† N/A = "Don't care" for this example



Legend:

▨ = Don't care. Signal can be high or low during this time.

NOTES: A. All XINTF accesses (lead period) begin on the rising edge of XCLKOUT. When necessary, the device will insert an alignment cycle before an access to meet this requirement.

B. During alignment cycles, all signals will transition to their inactive state.

C. During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.

D. For each sample, setup time from the beginning of the access can be calculated as:
D = (XRDLEAD + XRDACTIVE −3 +n) $t_{c(XTIM)}$ − $t_{su(XRDYasynchL)XCOHL}$
where n is the sample number: n = 1, 2, 3, and so forth.

E. Reference for the first sample is with respect to this point:
E = (XRDLEAD + XRDACTIVE −2) $t_{c(XTIM)}$

Figure 6−33. Example Read With Asynchronous XREADY Access

XTIMING register parameters used for this example:

XRDLEAD	XRDACTIVE	XRDTRAIL	USEREADY	X2TIMING	XWRLEAD	XWRACTIVE	XWRTRAIL	READYMODE
≥1	3	≥1	1	0	N/A†	N/A†	N/A†	1 = XREADY (Async)

† N/A = "Don't care" for this example

6.27 External Interface Ready-on-Write Timing With One External Wait State

Table 6−37. External Memory Interface Write Switching Characteristics (Ready-on-Write, 1 Wait State)

PARAMETER		MIN	MAX	UNIT
$t_{d(XCOH-XZCSL)}$	Delay time, XCLKOUT high to zone chip-select active low		1	ns
$t_{d(XCOHL-XZCSH)}$	Delay time, XCLKOUT high or low to zone chip-select inactive high	−2	3	ns
$t_{d(XCOH-XA)}$	Delay time, XCLKOUT high to address valid		2	ns
$t_{d(XCOHL-XWEL)}$	Delay time, XCLKOUT high/low to $\overline{XWE}$ low		2	ns
$t_{d(XCOHL-XWEH)}$	Delay time, XCLKOUT high/low to $\overline{XWE}$ high		2	ns
$t_{d(XCOH-XRNWL)}$	Delay time, XCLKOUT high to XR/$\overline{W}$ low		1	ns
$t_{d(XCOHL-XRNWH)}$	Delay time, XCLKOUT high/low to XR/$\overline{W}$ high	−2	1	ns
$t_{en(XD)XWEL}$	Enable time, data bus driven from $\overline{XWE}$ low	0		ns
$t_{d(XWEL-XD)}$	Delay time, data valid after $\overline{XWE}$ active low		4	ns
$t_{h(XA)XZCSH}$	Hold time, address valid after zone chip-select inactive high	†		ns
$t_{h(XD)XWE}$	Hold time, write data valid after $\overline{XWE}$ inactive high	TW−2‡		ns
$t_{dis(XD)XRNW}$	Data bus disabled after XR/$\overline{W}$ inactive high	4		ns

† During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.
‡ TW = trail period, write access (see Table 6−28)

Table 6−38. Synchronous XREADY Timing Requirements (Ready-on-Write, 1 Wait State)§

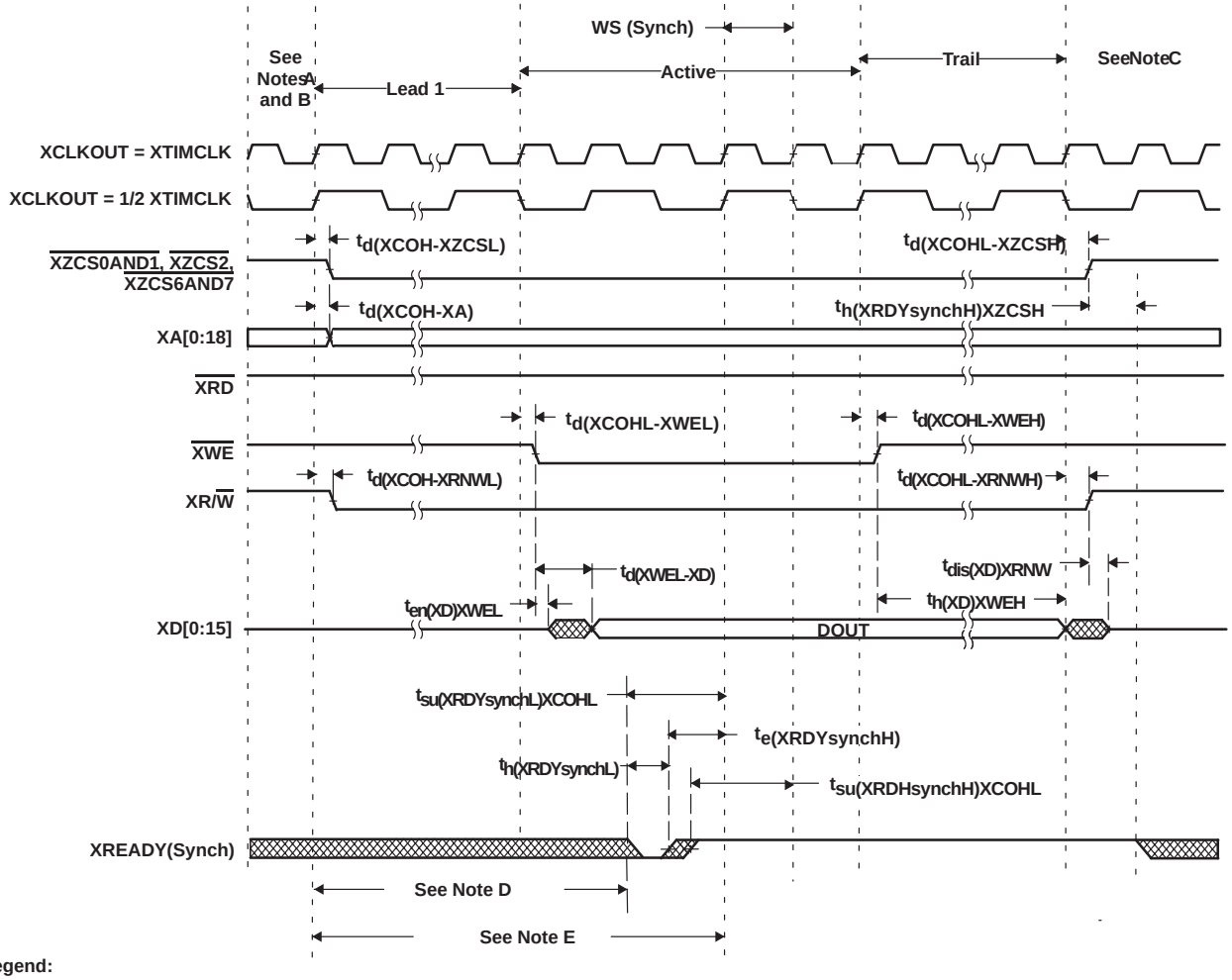
		MIN	MAX	UNIT
$t_{su(XRDYsynchL)XCOHL}$	Setup time, XREADY (Synch) low before XCLKOUT high/low	15		ns
$t_{h(XRDYsynchL)}$	Hold time, XREADY (Synch) low	12		ns
$t_{e(XRDYsynchH)}$	Earliest time XREADY (Synch) can go high before the sampling XCLKOUT edge		3	ns
$t_{su(XRDYsynchH)XCOHL}$	Setup time, XREADY (Synch) high before XCLKOUT high/low	15		ns
$t_{h(XRDYsynchH)XZCSH}$	Hold time, XREADY (Synch) held high after zone chip select high	0		ns

§ The first XREADY (Synch) sample occurs with respect to E in Figure 6−34:
E =(XWRLEAD + XWRACTIVE) $t_{c(XTIM)}$
When first sampled, if XREADY (Synch) is found to be high, then the access will complete. If XREADY (Synch) is found to be low, it will be sampled again each $t_{c(XTIM)}$ until it is found to be high.
For each sample, setup time from the beginning of the access can be calculated as:
D =(XWRLEAD + XWRACTIVE +n − 1) $t_{c(XTIM)}$ − $t_{su(XRDYsynchL)XCOHL}$
where n is the sample number: n = 1, 2, 3, and so forth.

Table 6−39. Asynchronous XREADY Timing Requirements (Ready-on-Write, 1 Wait State)¶

		MIN	MAX	UNIT
$t_{su(XRDYasynchL)XCOHL}$	Setup time, XREADY (Async) low before XCLKOUT high/low	11		ns
$t_{h(XRDYasynchL)}$	Hold time, XREADY (Async) low	8		ns
$t_{e(XRDYasynchH)}$	Earliest time XREADY (Async) can go high before the sampling XCLKOUT edge		3	ns
$t_{su(XRDYasynchH)XCOHL}$	Setup time, XREADY (Async) high before XCLKOUT high/low	11		ns
$t_{h(XRDYasynchH)XZCSH}$	Hold time, XREADY (Async) held high after zone chip select high	0		ns

¶ The first XREADY (Synch) sample occurs with respect to E in Figure 6−35:
E = (XWRLEAD + XWRACTIVE − 2) $t_{c(XTIM)}$
When first sampled, if XREADY (Async) is found to be high, then the access will complete. If XREADY (Async) is found to be low, it will be sampled again each $t_{c(XTIM)}$ until it is found to be high.
For each sample, setup time from the beginning of the access can be calculated as:
D = (XWRLEAD + XWRACTIVE −3 + n) $t_{c(XTIM)}$ − $t_{su(XRDYasynchL)XCOHL}$
where n is the sample number: n = 1, 2, 3, and so forth.



Legend:

= Don't care. Signal can be high or low during this time.

NOTES: A. All XINTF accesses (lead period) begin on the rising edge of XCLKOUT. When necessary, the device will insert an alignment cycle before an access to meet this requirement.

B. During alignment cycles, all signals will transition to their inactive state.

C. During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.

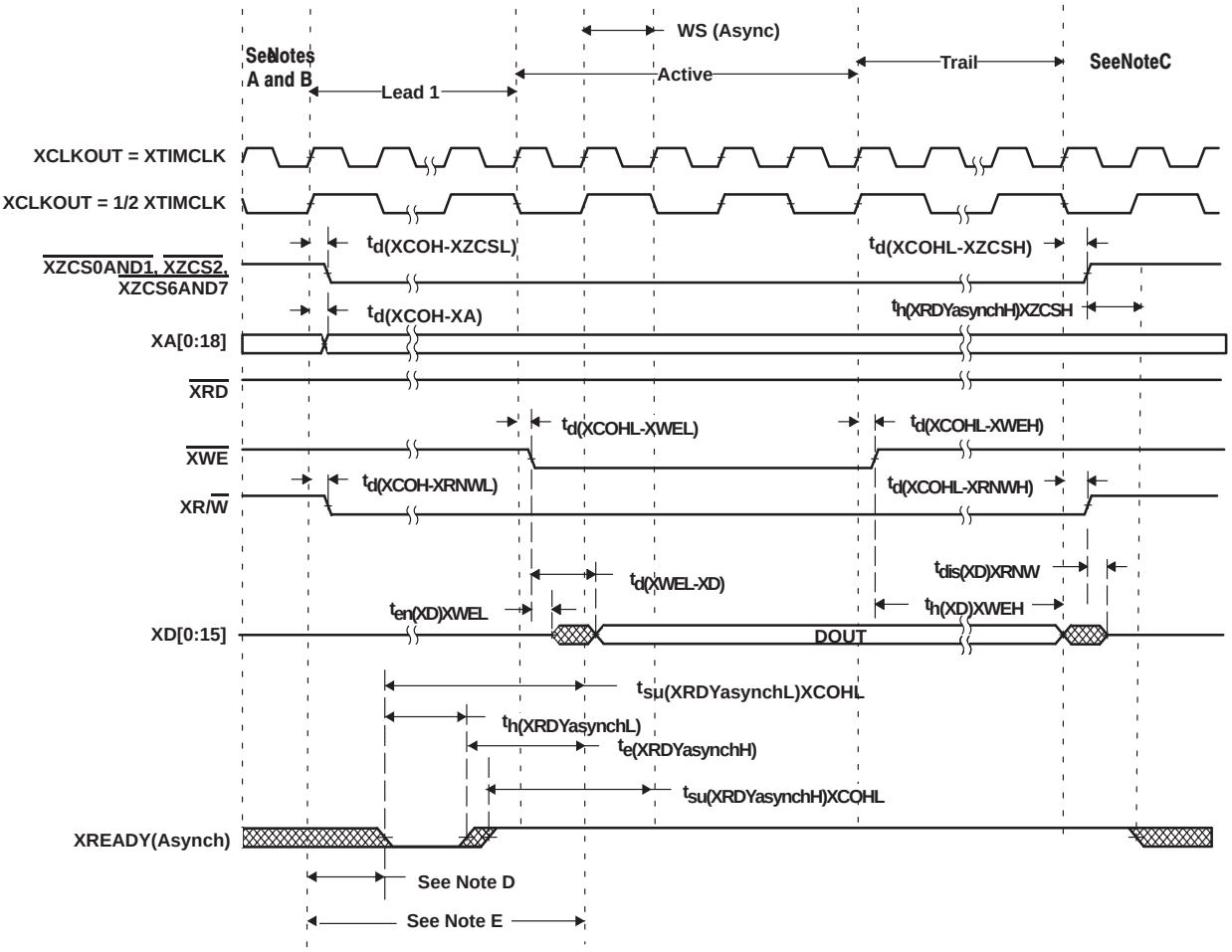
D. For each sample, setup time from the beginning of the access can be calculated as
$D = (XWRLEAD + XWRACTIVE + n - 1)\ t_{c(XTIM)} - t_{su(XRDYsynchL)XCOHL}$
where n is the sample number: n = 1, 2, 3 and so forth.

E. Reference for the first sample is with respect to this point
$E = (XWRLEAD + XWRACTIVE)\ t_{c(XTIM)}$

Figure 6−34. Write With Synchronous XREADY Access

XTIMING register parameters used for this example:

XRDLEAD	XRDACTIVE	XRDTRAIL	USEREADY	X2TIMING	XWRLEAD	XWRACTIVE	XWRTRAIL	READYMODE
N/A†	N/A†	N/A†	1	0	≥1	3	≥1	0 = XREADY (Synch)



Legend:

▨ = Don't care. Signal can be high or low during this time.

NOTES: A. All XINTF accesses (lead period) begin on the rising edge of XCLKOUT. When necessary, the device will insert an alignment cycle before an access to meet this requirement.

B. During alignment cycles, all signals will transition to their inactive state.

C. During inactive cycles, the XINTF address bus will always hold the last address put out on the bus. This includes alignment cycles.

D. For each sample, setup time from the beginning of the access can be calculated as:
$D = (XWRLEAD + XWRACTIVE - 3 + n)\ t_{c(XTIM)} - t_{su(XRDYasynchL)XCOHL}$
where n is the sample number: n = 1, 2, 3 and so forth.

E. Reference for the first sample is with respect to this point
$E = (XWRLEAD + XWRACTIVE - 2)\ t_{c(XTIM)}$

Figure 6−35. Write With Asynchronous XREADY Access

XTIMING register parameters used for this example:

XRDLEAD	XRDACTIVE	XRDTRAIL	USEREADY	X2TIMING	XWRLEAD	XWRACTIVE	XWRTRAIL	READYMODE
N/A†	N/A†	N/A†	1	0	≥1	3	≥1	1 = XREADY (Async)

† N/A = "Don't care" for this example

6.28 XHOLD and XHOLDA

If the HOLD mode bit is set while XHOLD and XHOLDA are both low (external bus accesses granted), the XHOLDA signal is forced high (at the end of the current cycle) and the external interface is taken out of high-impedance mode.

On a reset (XRS), the HOLD mode bit is set to 0. If the XHOLD signal is active low on a system reset, the bus and all signal strobes must be in high-impedance mode, and the XHOLDA signal is also driven active low.

When HOLD mode is enabled and XHOLDA is active low (external bus grant active), the CPU can still execute code from internal memory. If an access is made to the external interface, the CPU is stalled until the XHOLD signal is removed.

An external DMA request, when granted, places the following signals in a high-impedance mode:

XA[18:0]	XZCS0AND1
XD[15:0]	XZCS2
XWE, XRD	XZCS6AND7
XR/W	

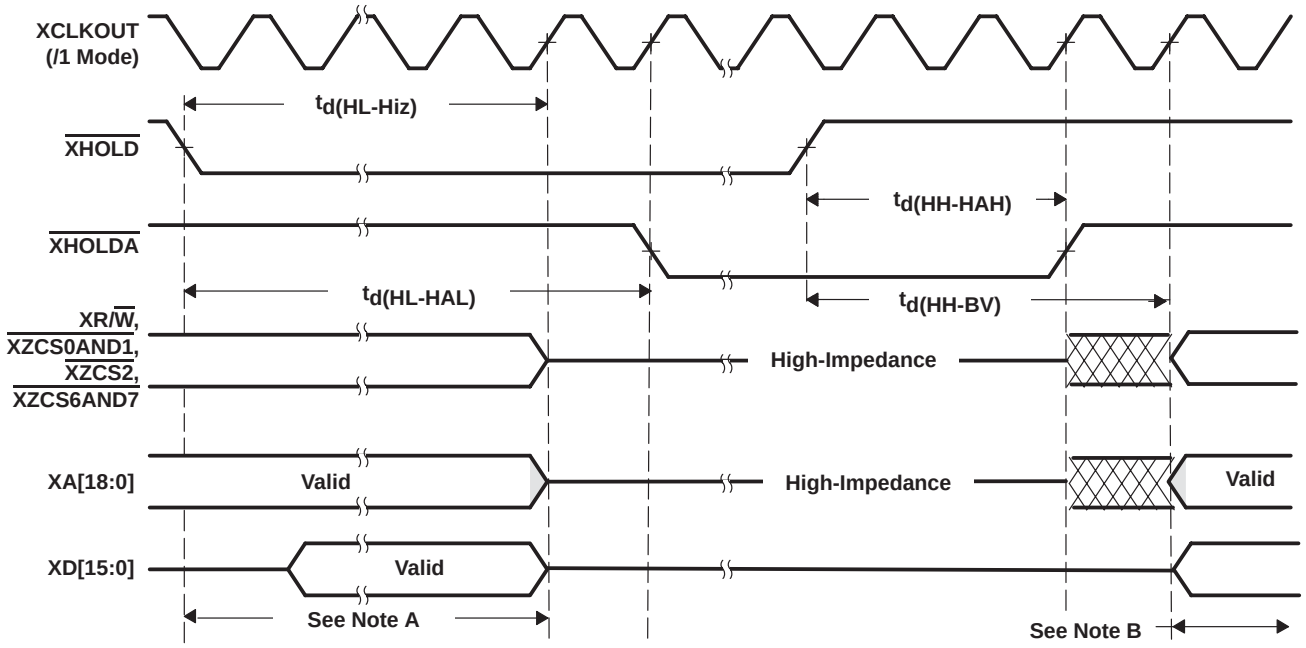
All other signals not listed in this group remain in their default or functional operational modes during these signal events. Detailed timing diagram will be released in a future revision of this data sheet.

6.29 $\overline{\text{XHOLD}}$/$\overline{\text{XHOLDA}}$ Timing

Table 6−40. $\overline{\text{XHOLD}}$/$\overline{\text{XHOLDA}}$ Timing Requirements (XCLKOUT = XTIMCLK)†‡

		MIN	MAX	UNIT
$t_{d(HL-HiZ)}$	Delay time, $\overline{\text{XHOLD}}$ low to Hi-Z on all Address, Data, and Control		$4t_{c(XTIM)}$	ns
$t_{d(HL-HAL)}$	Delay time, $\overline{\text{XHOLD}}$ low to $\overline{\text{XHOLDA}}$ low		$5t_{c(XTIM)}$	ns
$t_{d(HH-HAH)}$	Delay time, $\overline{\text{XHOLD}}$ high to $\overline{\text{XHOLDA}}$ high		$3t_{c(XTIM)}$	ns
$t_{d(HH-BV)}$	Delay time, $\overline{\text{XHOLD}}$ high to Bus valid		$4t_{c(XTIM)}$	ns

† When a low signal is detected on $\overline{\text{XHOLD}}$, all pending XINTF accesses will be completed before the bus is placed in a high-impedance state.
‡ The state of $\overline{\text{XHOLD}}$ is latched on the rising edge of XTIMCLK.



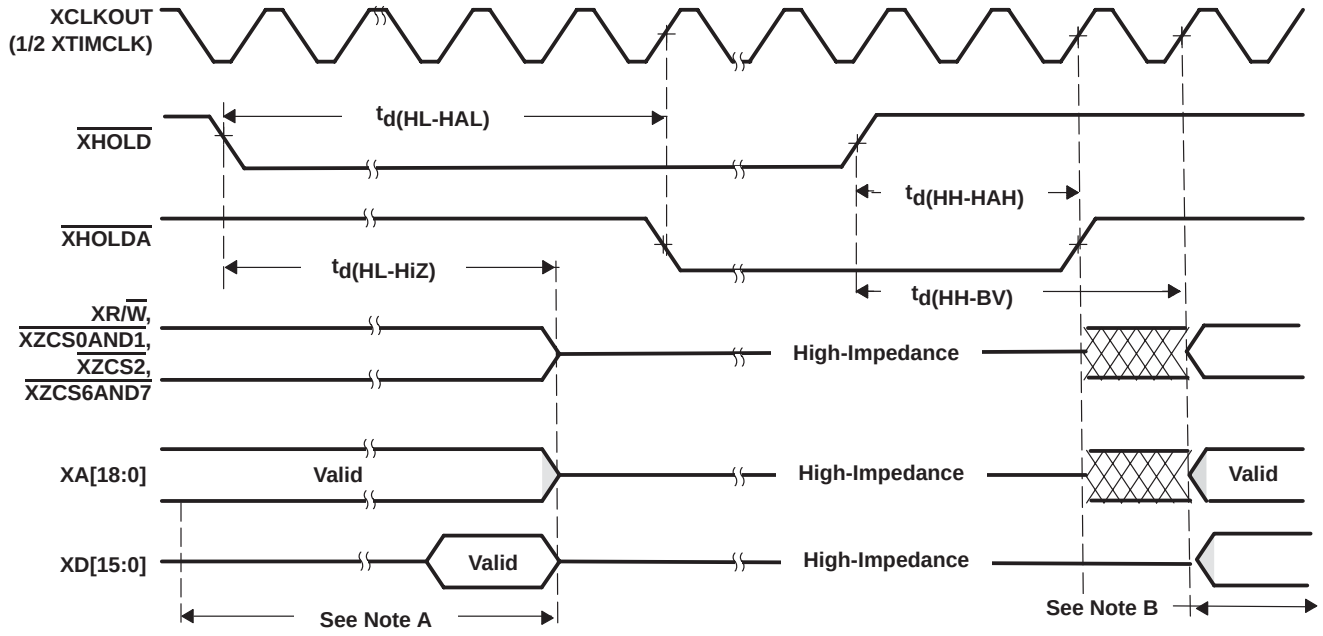
NOTES: A. All pending XINTF accesses are completed.
B. Normal XINTF operation resumes.

Figure 6−36. External Interface Hold Waveform

Table 6−41. $\overline{\text{XHOLD}}$/$\overline{\text{XHOLDA}}$ Timing Requirements (XCLKOUT = 1/2 XTIMCLK)†‡§

		MIN	MAX	UNIT
$t_{d(HL-HiZ)}$	Delay time, $\overline{\text{XHOLD}}$ low to Hi-Z on all Address, Data, and Control		$4t_{c(XTIM)}+t_{c(XCO)}$	ns
$t_{d(HL-HAL)}$	Delay time, $\overline{\text{XHOLD}}$ low to $\overline{\text{XHOLDA}}$ low		$4t_{c(XTIM}+2t_{c(XCO)}$	ns
$t_{d(HH-HAH)}$	Delay time, $\overline{\text{XHOLD}}$ high to $\overline{\text{XHOLDA}}$ high		$4t_{c(XTIM)}$	ns
$t_{d(HH-BV)}$	Delay time, $\overline{\text{XHOLD}}$ high to Bus valid		$6t_{c(XTIM)}$	ns

† When a low signal is detected on $\overline{\text{XHOLD}}$, all pending XINTF accesses will be completed before the bus is placed in a high-impedance state.
‡ The state of $\overline{\text{XHOLD}}$ is latched on the rising edge of XTIMCLK.
§ After the $\overline{\text{XHOLD}}$ is detected low or high, all bus transitions and $\overline{\text{XHOLDA}}$ transitions will occur with respect to the rising edge of XCLKOUT. Thus, for this mode where XCLKOUT = 1/2 XTIMCLK, the transitions can occur up to 1 XTIMCLK cycle earlier than the maximum value specified.



NOTES: A All pending XINTF accesses are completed.
B Normal XINTF operation resumes.

Figure 6−37. $\overline{\text{XHOLD}}$/$\overline{\text{XHOLDA}}$ Timing Requirements (XCLKOUT = 1/2 XTIMCLK)

6.30 On-Chip Analog-to-Digital Converter

6.30.1 ADC Absolute Maximum Ratings†

Supply voltage range, V_{SSA1}/V_{SSA2} to $V_{DDA1}/V_{DDA2}/AV_{DDREFBG}$ . . . . . −0.3 V to 4.6 V

V_{SS1} to V_{DD1} . . . . . −0.3 V to 2.5 V

Analog Input (ADCIN) Clamp Current, total (max) . . . . . ± 20 mA‡

† Unless otherwise noted, the list of absolute maximum ratings are specified over operating conditions. Stresses beyond those listed under Absolute Maximum Ratings may cause permanent damage to the device. These are stress ratings only. Exposure to absolute-maximum-rated conditions for extended periods may affect device reliability.

‡ The analog inputs have an internal clamping circuit that clamps the voltage to a diode drop above V_{DDA} or below V_{SS}. The continuous clamp current per pin is ± 2 mA.

6.30.2 ADC Electrical Characteristics Over Recommended Operating Conditions

Table 6−42. DC Specifications (See Note 1)

PARAMETER		MIN	TYP	MAX	UNIT
Resolution		12			Bits
ADC clock (See Note 2)		1			kHz
				25	MHz
ACCURACY					
INL (Integral nonlinearity) (See Note 3)	1−18.75 MHz ADC clock			±1.5	LSB
DNL (Differential nonlinearity) (See Note 3)	1−18.75 MHz ADC clock			±1	LSB
Offset error (See Note 4)		−80		80	LSB
Overall gain error with internal reference (See Note 5)	F281x	−200		200	LSB
	C281x	−80		80	LSB
Overall gain error with external reference (See Note 6)	If ADCREFP−ADCREFM = 1 V ±0.1%	−50		50	LSB
Channel-to-channel offset variation			±8		LSB
Channel-to-channel Gain variation			±8		LSB
ANALOG INPUT					
Analog input voltage (ADCINx to ADCLO) (See Note 7)		0		3	V
ADCLO		−5	0	5	mV
Input capacitance			10		pF
Input leakage current			3	±5	µA
INTERNAL VOLTAGE REFERENCE (See Note 5)					
Accuracy, $ADCV_{REFP}$		1.9	2	2.1	V
Accuracy, $ADCV_{REFM}$		0.95	1	1.05	V
Voltage difference, ADCREFP − ADCREFM			1		V
Temperature coefficient			50		PPM/°C
Reference noise			100		µV
EXTERNAL VOLTAGE REFERENCE (See Note 6)					
Accuracy, $ADCV_{REFP}$		1.9	2	2.1	V
Accuracy, $ADCV_{REFM}$		0.95	1	1.05	V
Input voltage difference, ADCREFP − ADCREFM		0.99	1	1.01	V

NOTES:
1. Tested at 12.5-MHz ADCCLK
2. If SYSCLKOUT ≤ 25 MHz, ADC clock ≤ SYSCLKOUT/2
3. The INL degrades for frequencies beyond 18.75 MHz − 25 MHz. Applications that require these sampling rates should use a 20K-resistor as bias resistor on the ADCRESEXT pin. This improves overall linearity and typical current drawn by the ADC will be a few mA more than 24.9 kΩ bias. The ADC module in C281x devices can operate at 24.9k bias on ADCRESEXT pin for the full range 1−25MHz
4. 1 LSB has the weighted value of 3.0/4096 = 0.732 mV.
5. A single internal band gap reference (±5% accuracy) sources both ADCREFP and ADCREFM signals, and hence, these voltages track together. The ADC converter uses the difference between these two as its reference. The total gain error will be the combination of the gain error shown here and the voltage reference accuracy (ADCREFP − ADCREFM). A software-based calibration procedure is recommended for better accuracy. See *F2812 ADC Calibration Application Note* (literature number SPRA989) and Section 5.2, Documentation Support, for relevant documents.
6. In this mode, the accuracy of external reference is critical for overall gain. The voltage difference (ADCREFP−ADCREFM) will determine the overall accuracy.
7. Voltages above V_{DDA} + 0.3 V or below V_{SS} − 0.3 V applied to an analog input pin may temporarily affect the conversion of another pin. To avoid this, the analog inputs should be kept within these limits.

Table 6−43. AC Specifications

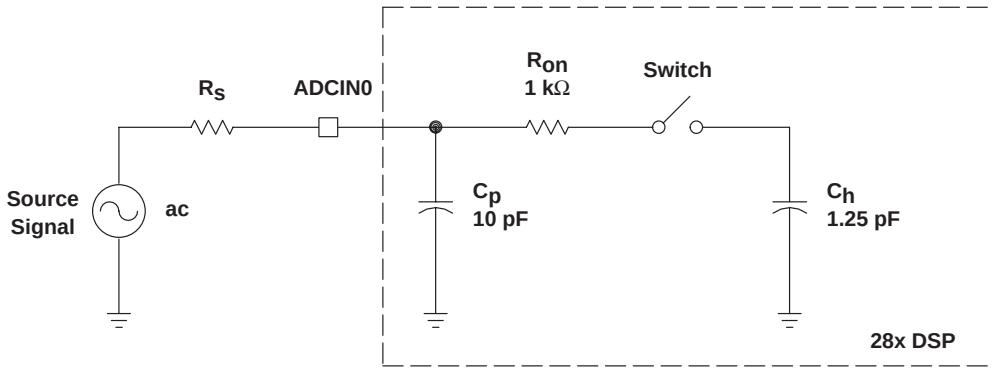
PARAMETER		MIN	TYP	MAX	UNIT
SINAD	Signal-to-noise ratio + distortion		62		dB
SNR	Signal-to-noise ratio		62		dB
THD (100 kHz)	Total harmonic distortion		−68		dB
ENOB (SNR)	Effective number of bits		10.1		Bits
SFDR	Spurious free dynamic range		69		dB

6.30.3 Current Consumption for Different ADC Configurations (at 25-MHz ADCCLK)‡

I_{DDA} (TYP)§	I_{DDAIO} (TYP)	I_{DD1} (TYP)	ADC OPERATING MODE/CONDITIONS
40 mA	1 µA	0.5 mA	Mode A (Operational Mode): − BG and REF enabled − PWD disabled
7 mA	0	5 µA	Mode B: − ADC clock enabled − BG and REF enabled − PWD enabled
1 µA	0	5 µA	Mode C: − ADC clock enabled − BG and REF disabled − PWD enabled
1 µA	0	0	Mode D: − ADC clock disabled − BG and REF disabled − PWD enabled

‡ Test Conditions: SYSCLKOUT = 150 MHz
ADC module clock = 25 MHz
ADC performing a continuous conversion of all 16 channels in Mode A

§ I_{DDA} − includes current into V_{DDA1} / V_{DDA2} and $AV_{DDREFBG}$



Typical Values of the Input Circuit Components:

Switch Resistance (R_{on}): 1 kΩ
Sampling Capacitor (C_h): 1.25 pF
Parasitic Capacitance (C_p): 10 pF
Source Resistance (R_S): 50 Ω

Figure 6−38. ADC Analog Input Impedance Model

6.30.4 ADC Power-Up Control Bit Timing

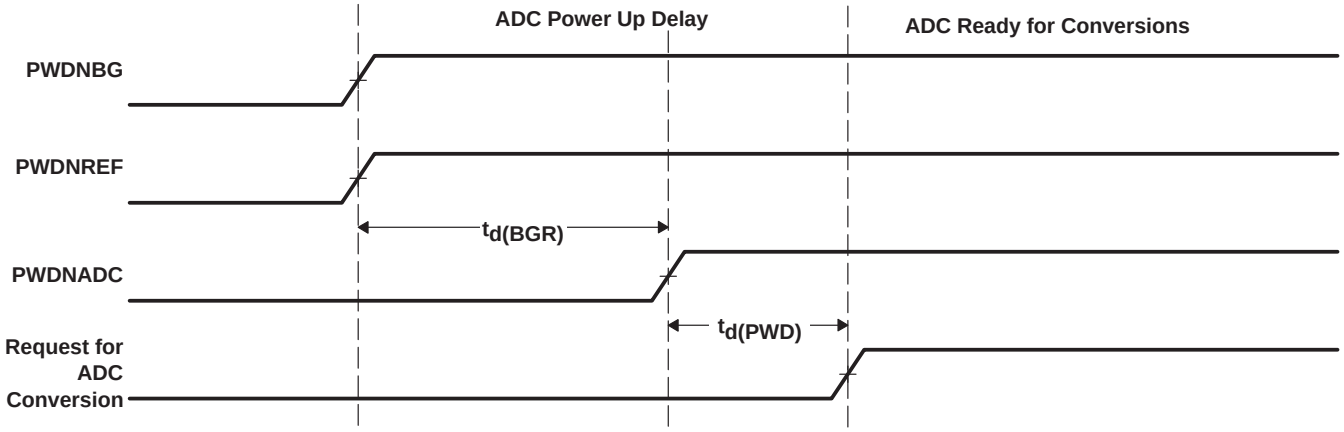


Figure 6−39. ADC Power-Up Control Bit Timing

Table 6−44. ADC Power-Up Delays†

		MIN	TYP	MAX	UNIT
$t_{d(BGR)}$	Delay time for band gap reference to be stable. Bits 6 and 5 of the ADCTRL3 register (PWDNBG and PWDNREF) are to be set to 1 before the ADCPWDN bit is enabled.	7	8	10	ms
$t_{d(PWD)}$	Delay time for power-down control to be stable. Bit 7 of the ADCTRL3 register (ADCPWDN) is to be set to 1 before any ADC conversions are initiated.	20	50		µs
				1	ms

† These delays are necessary and recommended to make the ADC analog reference circuit stable before conversions are initiated. If conversions are started without these delays, the ADC results will show a higher gain. For power down, all three bits can be cleared at the same time.

6.30.5 Detailed Description

6.30.5.1 Reference Voltage

The on-chip ADC has a built-in reference, which provides the reference voltages for the ADC. ADCVREFP is set to 2.0 V and ADCVREFM is set to 1.0 V.

6.30.5.2 Analog Inputs

The on-chip ADC consists of 16 analog inputs, which are sampled either one at a time or two channels at a time. These inputs are software-selectable.

6.30.5.3 Converter

The on-chip ADC uses a 12-bit four-stage pipeline architecture, which achieves a high sample rate with low power consumption.

6.30.5.4 Conversion Modes

The conversion can be performed in two different conversion modes:

- Sequential sampling mode (SMODE = 0)
- Simultaneous sampling mode (SMODE = 1)

6.30.6 Sequential Sampling Mode (Single-Channel) (SMODE = 0)

In sequential sampling mode, the ADC can continuously convert input signals on any of the channels (Ax to Bx). The ADC can start conversions on event triggers from the Event Managers (EVA/EVB), software trigger, or from an external ADCSOC signal. If the SMODE bit is 0, the ADC will do conversions on the selected channel on every Sample/Hold pulse. The conversion time and latency of the Result register update are explained below. The ADC interrupt flags are set a few SYSCLKOUT cycles after the Result register update. The selected channels will be sampled at every falling edge of the Sample/Hold pulse. The Sample/Hold pulse width can be programmed to be 1 ADC clock wide (minimum) or 16 ADC clocks wide (maximum).

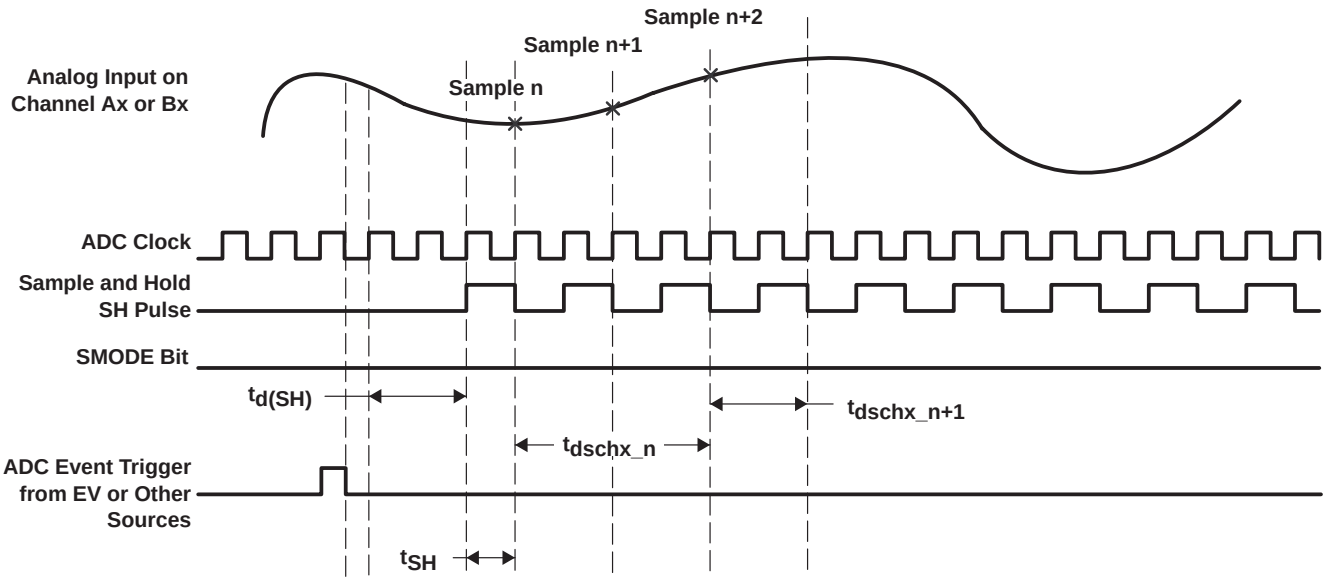


Figure 6−40. Sequential Sampling Mode (Single-Channel) Timing

Table 6−45. Sequential Sampling Mode Timing

		SAMPLE n	SAMPLE n + 1	AT 25-MHz ADC CLOCK, $t_{c(ADCCLK)}$ = 40 ns	REMARKS
$t_{d(SH)}$	Delay time from event trigger to sampling	$2.5t_{c(ADCCLK)}$			
t_{SH}	Sample/Hold width/ Acquisition width	(1 + Acqps) * $t_{c(ADCCLK)}$		40 ns with Acqps = 0	Acqps value = 0−15 ADCTRL1[8:11]
$t_{d(schx\_n)}$	Delay time for first result to appear in the Result register	$4t_{c(ADCCLK)}$		160 ns	
$t_{d(schx\_n+1)}$	Delay time for successive results to appear in the Result register		(2 + Acqps) * $t_{c(ADCCLK)}$	80 ns	

6.30.7 Simultaneous Sampling Mode (Dual-Channel) (SMODE = 1)

In simultaneous mode, the ADC can continuously convert input signals on any one pair of channels (A0/B0 to A7/B7). The ADC can start conversions on event triggers from the Event Managers (EVA/EVB), software trigger, or from an external ADCSOC signal. If the SMODE bit is 1, the ADC will do conversions on two selected channels on every Sample/Hold pulse. The conversion time and latency of the Result register update are explained below. The ADC interrupt flags are set a few SYSCLKOUT cycles after the Result register update. The selected channels will be sampled simultaneously at the falling edge of the Sample/Hold pulse. The Sample/Hold pulse width can be programmed to be 1 ADC clock wide (minimum) or 16 ADC clocks wide (maximum).

NOTE: In Simultaneous mode, the ADCIN channel pair select has to be A0/B0, A1/B1, ..., A7/B7, and *not* in other combinations (such as A1/B3, etc.).

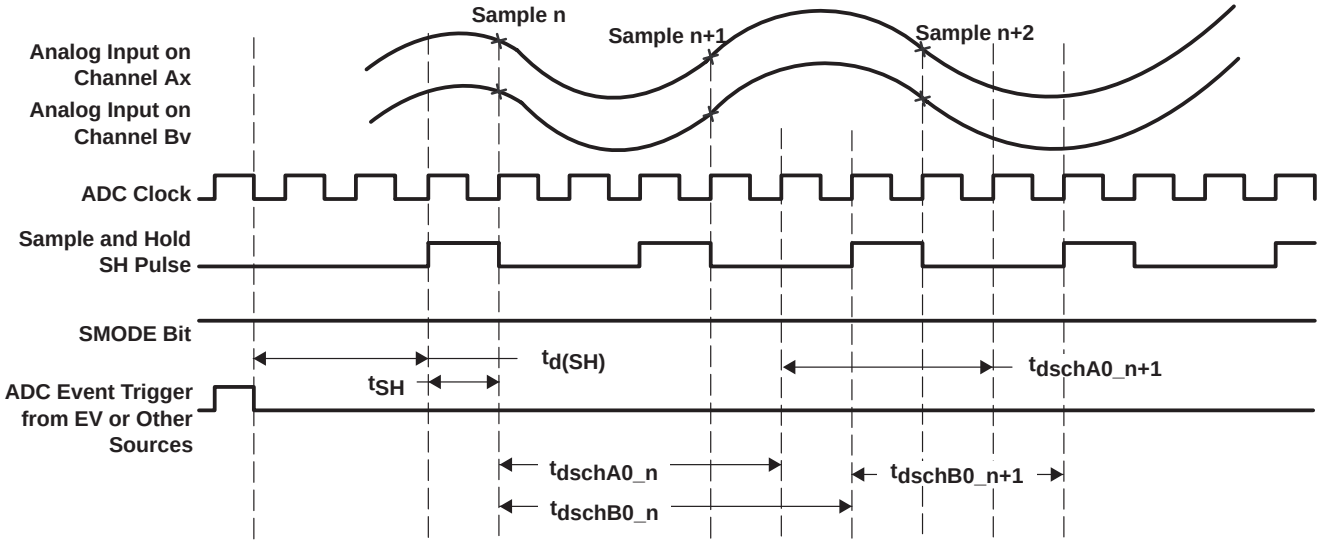


Figure 6−41. Simultaneous Sampling Mode Timing

Table 6−46. Simultaneous Sampling Mode Timing

		SAMPLE n	SAMPLE n + 1	AT 25-MHz ADC CLOCK, $t_{c(ADCCLK)}$ = 40 ns	REMARKS
$t_{d(SH)}$	Delay time from event trigger to sampling	$2.5t_{c(ADCCLK)}$			
t_{SH}	Sample/Hold width/ Acquisition Width	(1 + Acqps) * $t_{c(ADCCLK)}$		40 ns with Acqps = 0	Acqps value = 0−15 ADCTRL1[8:11]
$t_{d(schA0\_n)}$	Delay time for first result to appear in Result register	$4t_{c(ADCCLK)}$		160 ns	
$t_{d(schB0\_n)}$	Delay time for first result to appear in Result register	$5t_{c(ADCCLK)}$		200 ns	
$t_{d(schA0\_n+1)}$	Delay time for successive results to appear in Result register		(3 + Acqps) * $t_{c(ADCCLK)}$	120 ns	
$t_{d(schB0\_n+1)}$	Delay time for successive results to appear in Result register		(3 + Acqps) * $t_{c(ADCCLK)}$	120 ns	

6.30.8 Definitions of Specifications and Terminology

Integral Nonlinearity

Integral nonlinearity refers to the deviation of each individual code from a line drawn from zero through full scale. The point used as zero occurs 1/2 LSB before the first code transition. The full-scale point is defined as level 1/2 LSB beyond the last code transition. The deviation is measured from the center of each particular code to the true straight line between these two points.

Differential Nonlinearity

An ideal ADC exhibits code transitions that are exactly 1 LSB apart. DNL is the deviation from this ideal value. A differential nonlinearity error of less than ±1 LSB ensures no missing codes.

Zero Offset

The major carry transition should occur when the analog input is at zero volts. Zero error is defined as the deviation of the actual transition from that point.

Gain Error

The first code transition should occur at an analog value 1/2 LSB above negative full scale. The last transition should occur at an analog value 1 1/2 LSB below the nominal full scale. Gain error is the deviation of the actual difference between first and last code transitions and the ideal difference between first and last code transitions.

Signal-to-Noise Ratio + Distortion (SINAD)

SINAD is the ratio of the rms value of the measured input signal to the rms sum of all other spectral components below the Nyquist frequency, including harmonics but excluding dc. The value for SINAD is expressed in decibels.

Effective Number of Bits (ENOB)

For a sine wave, SINAD can be expressed in terms of the number of bits. Using the following formula,

$$N = \frac{(SINAD - 1.76)}{6.02}$$

it is possible to get a measure of performance expressed as N, the effective number of bits. Thus, effective number of bits for a device for sine wave inputs at a given input frequency can be calculated directly from its measured SINAD.

Total Harmonic Distortion (THD)

THD is the ratio of the rms sum of the first six harmonic components to the rms value of the measured input signal and is expressed as a percentage or in decibels.

Spurious Free Dynamic Range (SFDR)

SFDR is the difference in dB between the rms amplitude of the input signal and the peak spurious signal.

6.31 Multichannel Buffered Serial Port (McBSP) Timing

6.31.1 McBSP Transmit and Receive Timing

Table 6−47. McBSP Timing Requirements†‡

NO.				MIN	MAX	UNIT
	McBSP module clock (CLKG, CLKX, CLKR) range			1		kHz
					20§	MHz
	McBSP module cycle time (CLKG, CLKX, CLKR) range			50		ns
					1	ms
M11	$t_{c(CKRX)}$	Cycle time, CLKR/X	CLKR/X ext	2P		ns
M12	$t_{w(CKRX)}$	Pulse duration, CLKR/X high or CLKR/X low	CLKR/X ext	P−7		ns
M13	$t_{r(CKRX)}$	Rise time, CLKR/X	CLKR/X ext		7	ns
M14	$t_{f(CKRX)}$	Fall time, CLKR/X	CLKR/X ext		7	ns
M15	$t_{su(FRH-CKRL)}$	Setup time, external FSR high before CLKR low	CLKR int	18		ns
			CLKR ext	2		
M16	$t_{h(CKRL-FRH)}$	Hold time, external FSR high after CLKR low	CLKR int	0		ns
			CLKR ext	6		
M17	$t_{su(DRV-CKRL)}$	Setup time, DR valid before CLKR low	CLKR int	18		ns
			CLKR ext	2		
M18	$t_{h(CKRL-DRV)}$	Hold time, DR valid after CLKR low	CLKR int	0		ns
			CLKR ext	6		
M19	$t_{su(FXH-CKXL)}$	Setup time, external FSX high before CLKX low	CLKX int	18		ns
			CLKX ext	2		
M20	$t_{h(CKXL-FXH)}$	Hold time, external FSX high after CLKX low	CLKX int	0		ns
			CLKX ext	6		

† Polarity bits CLKRP = CLKXP = FSRP = FSXP = 0. If the polarity of any of the signals is inverted, then the timing references of that signal are also inverted.

‡ 2P = 1/CLKG in ns. CLKG is the output of sample rate generator mux. CLKG = $\frac{\text{CLKSRG}}{(1 + \text{CLKGDV})}$.

CLKSRG can be LSPCLK, CLKX, CLKR as source. CLKSRG ≤ (SYSCLKOUT/2). McBSP performance is limited by I/O buffer switching speed.

§ Internal clock prescalers must be adjusted such that the McBSP clock (CLKG, CLKX, CLKR) speeds are not greater than the I/O buffer speed limit (20 MHz).

Table 6−48. McBSP Switching Characteristics†‡

NO.	PARAMETER				MIN	MAX	UNIT
M1	$t_{c(CKRX)}$	Cycle time, CLKR/X		CLKR/X int	2P		ns
M2	$t_{w(CKRXH)}$	Pulse duration, CLKR/X high		CLKR/X int	D−5§	D+5§	ns
M3	$t_{w(CKRXL)}$	Pulse duration, CLKR/X low		CLKR/X int	C−5§	C+5§	ns
M4	$t_{d(CKRH-FRV)}$	Delay time, CLKR high to internal FSR valid		CLKR int	0	4	ns
				CLKR ext	3	27	ns
M5	$t_{d(CKXH-FXV)}$	Delay time, CLKX high to internal FSX valid		CLKX int	0	4	ns
				CLKX ext	3	27	
M6	$t_{dis(CKXH-DXHZ)}$	Disable time, CLKX high to DX high impedance following last data bit		CLKX int		8	ns
				CLKX ext		14	
M7	$t_{d(CKXH-DXV)}$	Delay time, CLKX high to DX valid. This applies to all bits except the first bit transmitted.		CLKX int		9	ns
				CLKX ext		28	
		Delay time, CLKX high to DX valid	DXENA = 0	CLKX int		8	
				CLKX ext		14	
		Only applies to first bit transmitted when in Data Delay 1 or 2 (XDATDLY=01b or 10b) modes	DXENA = 1	CLKX int		P + 8	
				CLKX ext		P + 14	
M8	$t_{en(CKXH-DX)}$	Enable time, CLKX high to DX driven	DXENA = 0	CLKX int	0		ns
				CLKX ext	6		
		Only applies to first bit transmitted when in Data Delay 1 or 2 (XDATDLY=01b or 10b) modes	DXENA = 1	CLKX int	P		
				CLKX ext	P + 6		
M9	$t_{d(FXH-DXV)}$	Delay time, FSX high to DX valid	DXENA = 0	FSX int		8	ns
				FSX ext		14	
		Only applies to first bit transmitted when in Data Delay 0 (XDATDLY=00b) mode.	DXENA = 1	FSX int		P + 8	
				FSX ext		P + 14	
M10	$t_{en(FXH-DX)}$	Enable time, FSX high to DX driven	DXENA = 0	FSX int	0		ns
				FSX ext	6		
		Only applies to first bit transmitted when in Data Delay 0 (XDATDLY=00b) mode	DXENA = 1	FSX int	P		
				FSX ext	P + 6		

† Polarity bits CLKRP = CLKXP = FSRP = FSXP = 0. If the polarity of any of the signals is inverted, then the timing references of that signal are also inverted.
‡ 2P = 1/CLKG in ns.
§ C=CLKRX low pulse width = P
D=CLKRX high pulse width = P

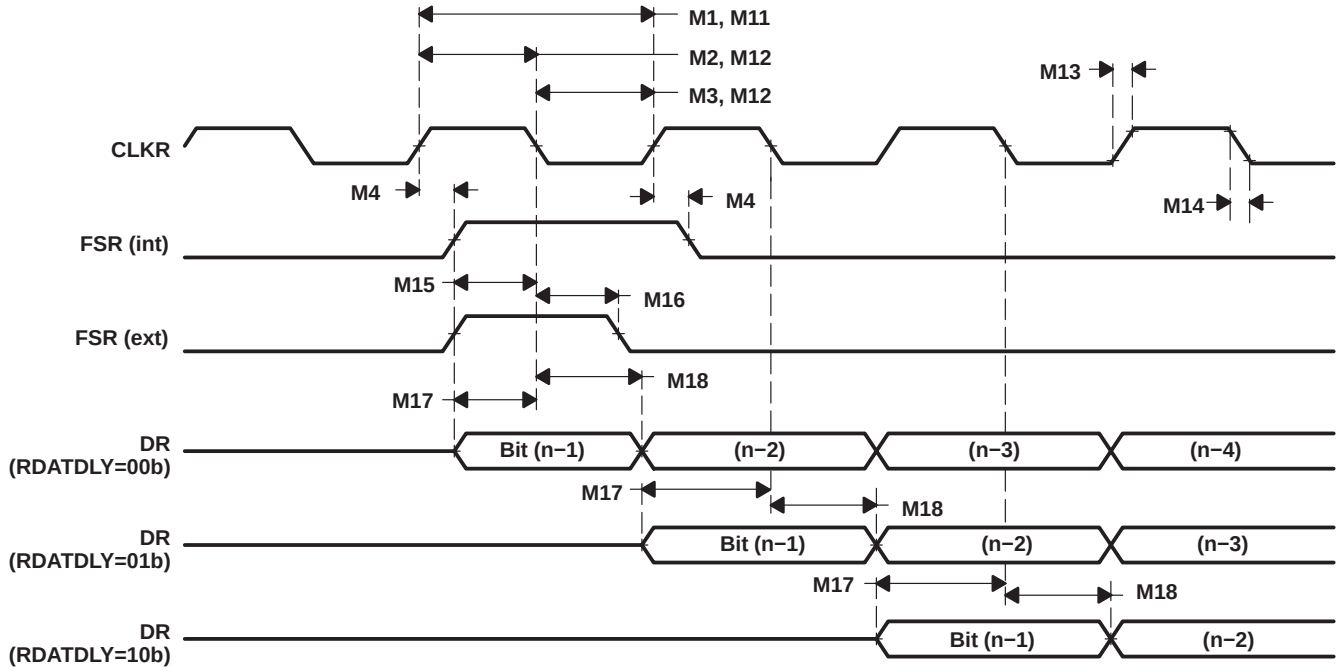


Figure 6−42. McBSP Receive Timing

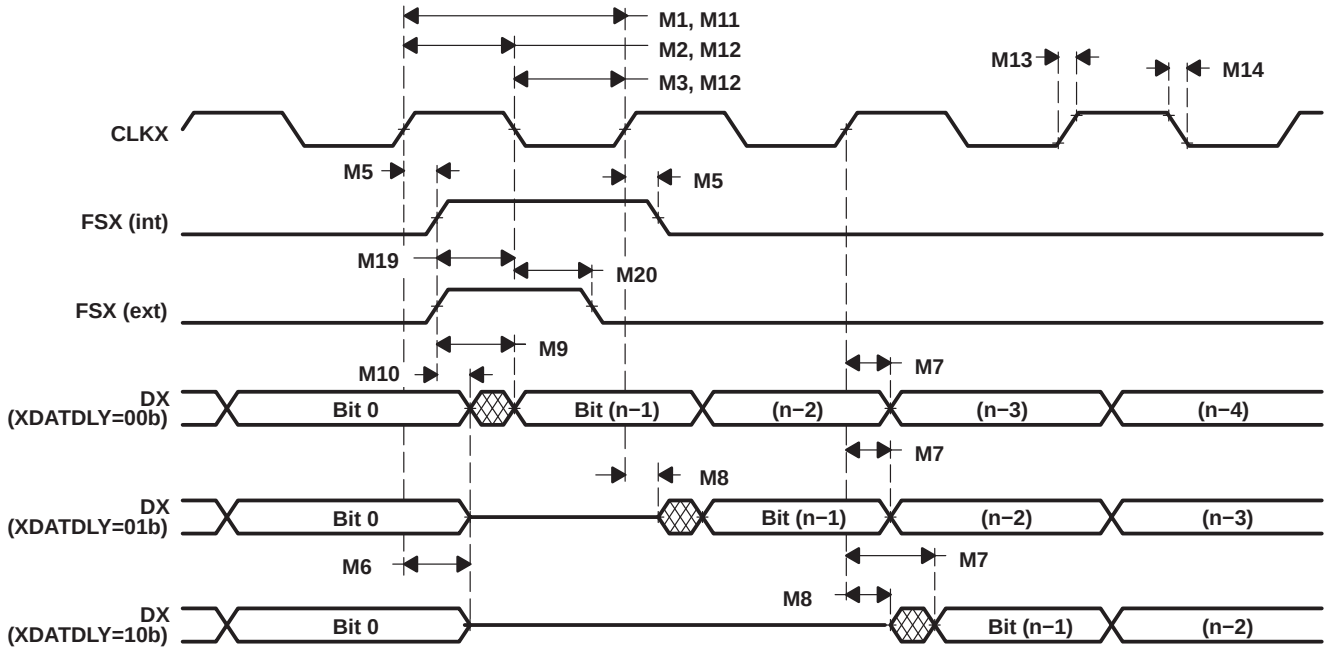


Figure 6−43. McBSP Transmit Timing

6.31.2 McBSP as SPI Master or Slave Timing

Table 6−49. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 10b, CLKXP = 0)

NO.			MASTER		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M30	$t_{su(DRV-CKXL)}$	Setup time, DR valid before CLKX low	P−10		8P−10		ns
M31	$t_{h(CKXL-DRV)}$	Hold time, DR valid after CLKX low	P−10		8P−10		ns
M32	$t_{su(BFXL-CKXH)}$	Setup time, FSX low before CLKX high			8P+10		ns
M33	$t_{c(CKX)}$	Cycle time, CLKX	2P		16P		ns

Table 6−50. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 10b, CLKXP = 0)†

NO.	PARAMETER		MASTER		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M24	$t_{h(CKXL-FXL)}$	Hold time, FSX low after CLKX low	2P				ns
M25	$t_{d(FXL-CKXH)}$	Delay time, FSX low to CLKX high	P				ns
M28	$t_{dis(FXH-DXHZ)}$	Disable time, DX high impedance following last data bit from FSX high	6		6P + 6		ns
M29	$t_{d(FXL-DXV)}$	Delay time, FSX low to DX valid	6		4P + 6		ns

† 2P = 1/CLKG

For all SPI slave modes, CLKX has to be minimum 8 CLKG cycles. Also CLKG should be LSPCLK/2 by setting CLKSM = CLKGDV = 1. With maximum LSPCLK speed of 75 MHz, CLKX maximum frequency will be LSPCLK/16 , that is 4.5 MHz and P =13.3 ns.

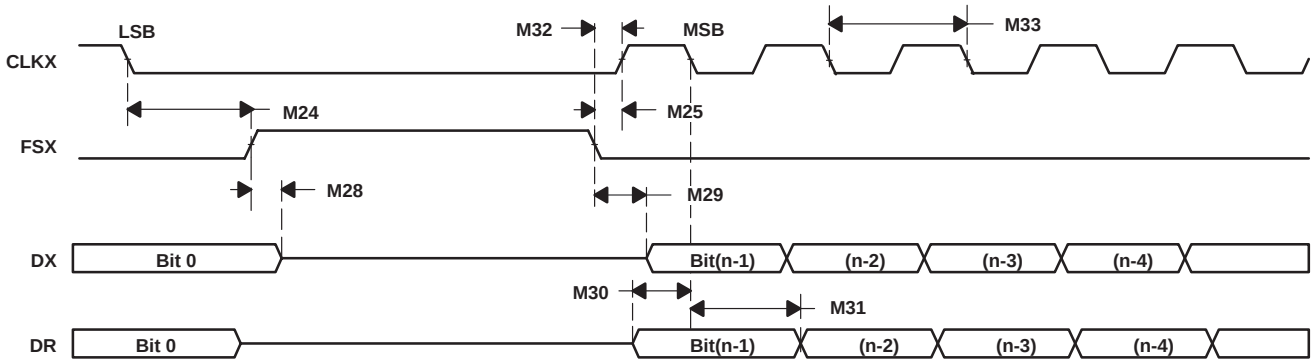


Figure 6−44. McBSP Timing as SPI Master or Slave: CLKSTP = 10b, CLKXP = 0

Table 6−51. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 11b, CLKXP = 0)†

NO.			MASTER		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M39	$t_{su(DRV-CKXH)}$	Setup time, DR valid before CLKX high	P−10		8P−10		ns
M40	$t_{h(CKXH-DRV)}$	Hold time, DR valid after CLKX high	P−10		8P−10		ns
M41	$t_{su(FXL-CKXH)}$	Setup time, FSX low before CLKX high			16P+10		ns
M42	$t_{c(CKX)}$	Cycle time, CLKX	2P		16P		ns

Table 6−52. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 11b, CLKXP = 0)†

NO.	PARAMETER		MASTER		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M34	$t_{h(CKXL-FXL)}$	Hold time, FSX low after CLKX low	P				ns
M35	$t_{d(FXL-CKXH)}$	Delay time, FSX low to CLKX high	2P				ns
M37	$t_{dis(CKXL-DXHZ)}$	Disable time, DX high impedance following last data bit from CLKX low	P + 6		7P+6		ns
M38	$t_{d(FXL-DXV)}$	Delay time, FSX low to DX valid	6		4P + 6		ns

† 2P = 1/CLKG

For all SPI slave modes, CLKX has to be minimum 8 CLKG cycles. Also CLKG should be LSPCLK/2 by setting CLKSM = CLKGDV = 1. With maximum LSPCLK speed of 75 MHz, CLKX maximum frequency will be LSPCLK/16 , that is 4.5 MHz and P =13.3 ns.

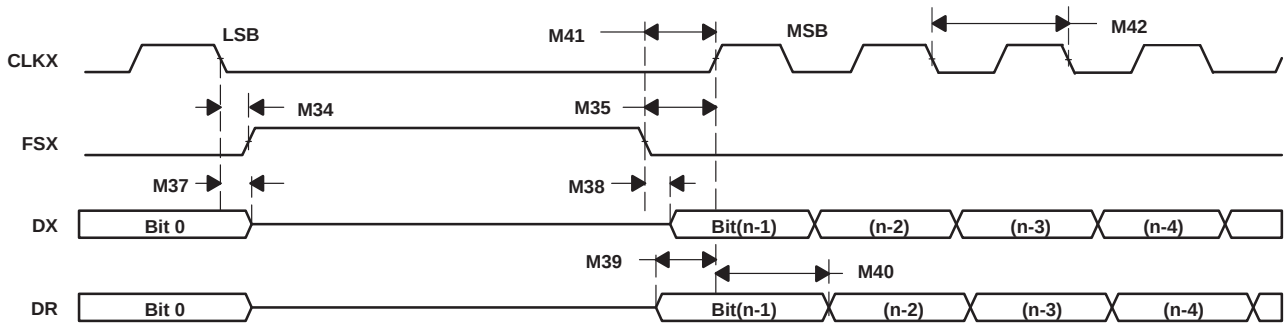


Figure 6−45. McBSP Timing as SPI Master or Slave: CLKSTP = 11b, CLKXP = 0

Table 6−53. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 10b, CLKXP = 1)†

NO.			MASTER		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M49	$t_{su(DRV-CKXH)}$	Setup time, DR valid before CLKX high	P−10		8P−10		ns
M50	$t_{h(CKXH-DRV)}$	Hold time, DR valid after CLKX high	P−10		8P−10		ns
M51	$t_{su(FXL-CKXL)}$	Setup time, FSX low before CLKX low			8P+10		ns
M52	$t_{c(CKX)}$	Cycle time, CLKX	2P		16P		ns

Table 6−54. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 10b, CLKXP = 1)†

NO.	PARAMETER		MASTER		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M43	$t_{h(CKXH-FXL)}$	Hold time, FSX low after CLKX high	2P				ns
M44	$t_{d(FXL-CKXL)}$	Delay time, FSX low to CLKX low	P				ns
M47	$t_{dis(FXH-DXHZ)}$	Disable time, DX high impedance following last data bit from FSX high	6		6P + 6		ns
M48	$t_{d(FXL-DXV)}$	Delay time, FSX low to DX valid	6		4P + 6		ns

† 2P = 1/CLKG

For all SPI slave modes, CLKX has to be minimum 8 CLKG cycles. Also CLKG should be LSPCLK/2 by setting CLKSM = CLKGDV = 1. With maximum LSPCLK speed of 75 MHz, CLKX maximum frequency will be LSPCLK/16 , that is 4.5 MHz and P =13.3 ns.

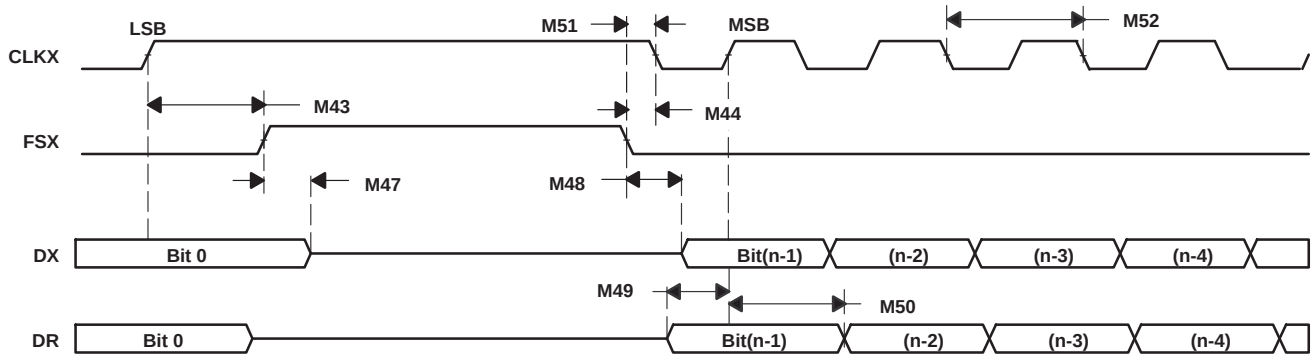


Figure 6−46. McBSP Timing as SPI Master or Slave: CLKSTP = 10b, CLKXP = 1

Table 6−55. McBSP as SPI Master or Slave Timing Requirements (CLKSTP = 11b, CLKXP = 1)†

NO.			MASTER		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M58	$t_{su(DRV-CKXL)}$	Setup time, DR valid before CLKX low	P − 10		8P − 10		ns
M59	$t_{h(CKXL-DRV)}$	Hold time, DR valid after CLKX low	P − 10		8P − 10		ns
M60	$t_{su(FXL-CKXL)}$	Setup time, FSX low before CLKX low			16P + 10		ns
M61	$t_{c(CKX)}$	Cycle time, CLKX	2P		16P		ns

Table 6−56. McBSP as SPI Master or Slave Switching Characteristics (CLKSTP = 11b, CLKXP = 1)†

NO.	PARAMETER		MASTER‡		SLAVE		UNIT
			MIN	MAX	MIN	MAX	
M53	$t_{h(CKXH-FXL)}$	Hold time, FSX low after CLKX high	P				ns
M54	$t_{d(FXL-CKXL)}$	Delay time, FSX low to CLKX low	2P				ns
M56	$t_{dis(CKXH-DXHZ)}$	Disable time, DX high impedance following last data bit from CLKX high	P+6		7P + 6		ns
M57	$t_{d(FXL-DXV)}$	Delay time, FSX low to DX valid	6		4P + 6		ns

† 2P = 1/CLKG
For all SPI slave modes, CLKX has to be minimum 8 CLKG cycles. Also CLKG should be LSPCLK/2 by setting CLKSM = CLKGDV = 1. With maximum LSPCLK speed of 75 MHz, CLKX maximum frequency will be LSPCLK/16 , that is 4.5 MHz and P =13.3 ns.
‡ C = CLKX low pulse width = P
D = CLKX high pulse width = P

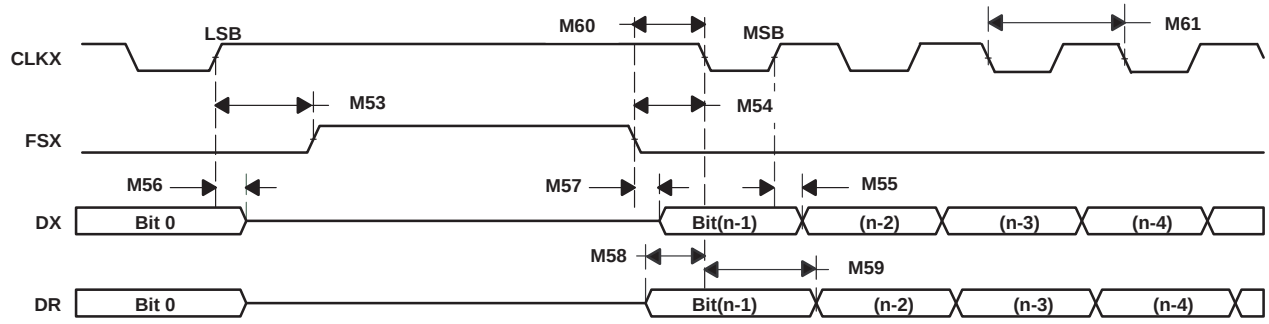


Figure 6−47. McBSP Timing as SPI Master or Slave: CLKSTP = 11b, CLKXP = 1

6.32 Flash Timing (F281x Only)

6.32.1 Recommended Operating Conditions

			MIN	NOM	MAX	UNIT
N_f	Flash endurance for the array (Write/erase cycles)	0°C to 85°C	100	1000		cycles
N_{OTP}	OTP endurance for the array (Write cycles)	0°C to 85°C			1	write

Table 6−57. Flash Parameters at 150-MHz SYSCLKOUT†

PARAMETER			MIN	TYP	MAX	UNIT
Program Time	16-Bit Word			35		µs
	8K Sector			170		ms
	16K Sector			320		ms
Erase Time	8K Sector			10		S
	16K Sector			11		S
$I_{DD3VFLP}$	V_{DD3VFL} current consumption during the Erase/Program cycle	Erase		75		mA
		Program		35		mA
I_{DDP}	V_{DD} current consumption during Erase/Program cycle			140		mA
I_{DDIOP}	V_{DDIO} current consumption during Erase/Program cycle			20		mA

† Typical parameters as seen at room temperature using flash API V1 including function call overhead.

Table 6−58. Flash/OTP Access Timing

PARAMETER		MIN	MAX	UNIT
$t_{a(fp)}$	Paged Flash access time	36		ns
$t_{a(fr)}$	Random Flash access time	36		ns
$t_{a(OTP)}$	OTP access time	60		ns

NOTE: For 150 MHz, PAGE WS = 5 and RANDOM WS = 5
For 135 MHz, PAGE WS = 4 and RANDOM WS = 4

Table 6−59. Minimum Required Wait-States at Different Frequencies (F281x devices)

SYSCLKOUT (MHz)	SYSCLKOUT (ns)	PAGE WAIT-STATE‡	RANDOM WAIT STATE‡§
150	6.67	5	5
120	8.33	4	4
100	10	3	3
75	13.33	2	2
50	20	1	1
30	33.33	1	1
25	40	0	1
15	66.67	0	1
4	250	0	1

‡ Formulas to compute page wait state and random wait state:

$$\text{Page Wait State} = \left[\left(\frac{t_{a(fp)}}{t_{c(SCO)}}\right) - 1\right] \text{ (round up to the next highest integer), or 0 whichever is larger}$$

$$\text{Random Wait State} = \left[\left(\frac{t_{a(fr)}}{t_{c(SCO)}}\right) - 1\right] \text{ (round up to the next highest integer), or 1 whichever is larger}$$

§ Random wait state must be greater than or equal to 1

Table 6−60. ROM Access Timing

PARAMETER			MIN	MAX	UNIT
$t_{a(rp)}$	Paged ROM access time		23		ns
$t_{a(rr)}$	Random ROM access time		23		ns
$t_{a(ROM)}$	ROM (OTP area) access time (see Note 2)		60		ns

NOTES: 1. For 150 MHz, PAGE WS = 3 and RANDOM WS = 3
For 135 MHz, PAGE WS = 3 and RANDOM WS = 3
2. In C281x devices, a 1K x 16 ROM block replaces the OTP block found in Flash devices.

Table 6−61. Minimum Required Wait-States at Different Frequencies (C281x devices)

SYSCLKOUT (MHz)	SYSCLKOUT (ns)	PAGE WAIT-STATE†	RANDOM WAIT STATE†‡
150	6.67	3	3
120	8.33	2	2
100	10	2	2
75	13.33	1	1
50	20	1	1
30	33.33	0	1
25	40	0	1
15	66.67	0	1
4	250	0	1

† Formulas to compute page wait state and random wait state:

$$\text{Page Wait State} = \left[\left(\frac{t_{a(rp)}}{t_{c(SCO)}}\right) - 1\right] \text{(round up to the next highest integer), or 0 whichever is larger}$$

$$\text{Random Wait State} = \left[\left(\frac{t_{a(rr)}}{t_{c(SCO)}}\right) - 1\right] \text{(round up to the next highest integer), or 1 whichever is larger}$$

‡ Random wait state must be greater than or equal to 1

6.33 Migrating From F281x Devices to C281x Devices

The migration issues to be considered while migrating from the F281x devices to C281x devices are as follows:

- The 1K OTP memory available in F281x devices has been replaced by 1K ROM C281x devices.
- Power sequencing is not needed for C281x devices. In other words, 3.3-V and 1.8-V (or 1.9-V) can ramp together. C281x can also be used on boards that have F281x power sequencing implemented; however, if the 1.8-V (or 1.9-V) rail lags the 3.3-V rail, the GPIO pins are undefined until the 1.8-V rail reaches at least 1 V.
- Current consumption differs for F281x and C281x devices for all four possible modes. See the appropriate electrical section for exact numbers.
- The V_{DD3VFL} pin is the 3.3-V flash core power pin in F281x devices but is a V_{DDIO} pin in C281x devices.
- F281x and C281x devices are pin-compatible and code-compatible; however, they are electrically different with different EMI/ESD profiles. Before ramping production with c281x devices, evaluate performance of the hardware design with both devices
- Addresses 0x3D7BFC through 0x3D7BFF in the OTP and addresses 0x3F7FF2 through 0x3F7FF5 in the main ROM array are reserved for ROM part-specific information and are not available for user applications.
- The ADC module in C281x devices can operate at 24.9k bias on ADCRESEXT pin for the full range 1−25MHz. While migrating the F281x designs to C281x, use a 24.9k resistor for biasing the ADC.
- The paged and random wait-state specifications for the flash and ROM parts are different. While migrating from flash to ROM parts, the same wait-state values must be used for best performance compatibility (for example, in applications that use software delay loops or where precise interrupt latencies are critical).

For errata applicable to 281x devices, see the *TMS320F2810, TMS320F2811, TMS320F2812, TMS320C2810, TMS320C2811, TMS320C2812 Digital Signal Processors Silicon Errata* (literature number SPRZ193).

7 Mechanical Data

Table 7−1 through Table 7−4 provide the thermal resistance characteristics for the various packages.

Table 7−1. Thermal Resistance Characteristics for 179-Ball GHH

PARAMETER	179-GHH PACKAGE	UNIT
Psi_{JT}	0.658	°C/W
Θ_{JA}	42.57	°C/W
Θ_{JC}	16.08	°C/W

Table 7−2. Thermal Resistance Characteristics for 179-Ball ZHH

PARAMETER	179-ZHH PACKAGE	UNIT
Psi_{JT}	0.658	°C/W
Θ_{JA}	42.57	°C/W
Θ_{JC}	16.08	°C/W

Table 7−3. Thermal Resistance Characteristics for 176-Pin PGF

PARAMETER	176-PGF PACKAGE	UNIT
Psi_{JT}	0.247	°C/W
Θ_{JA}	41.88	°C/W
Θ_{JC}	9.73	°C/W

Table 7−4. Thermal Resistance Characteristics for 128-Pin PBK

PARAMETER	128-PBK PACKAGE	UNIT
Psi_{JT}	0.271	°C/W
Θ_{JA}	41.65	°C/W
Θ_{JC}	10.76	°C/W

The following mechanical package diagram(s) reflect the most current released mechanical data available for the designated device(s).

PACKAGING INFORMATION

Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/Ball Finish	MSL Peak Temp [3]
F741814APBL-1	ACTIVE					None	Call TI	Call TI
TMS320C2810	ACTIVE					None	Call TI	Call TI
TMS320F2810PBKA	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2810PBKQ	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2811PBKA	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2811PBKQ	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2812GHHA	ACTIVE	BGA	GHH	179	160	None	Call TI	Level-3-220C-168HR
TMS320F2812GHHAR	ACTIVE	BGA	GHH	179	1000	None	Call TI	Level-3-220C-168HR
TMS320F2812GHHQ	ACTIVE	BGA	GHH	179	160	None	Call TI	Level-3-220C-168HR
TMS320F2812PGFA	ACTIVE	LQFP	PGF	176	40	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2812PGFQ	ACTIVE	LQFP	PGF	176	40	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2812ZHHA	ACTIVE	BGA MICROSTAR	ZHH	179	160	Green (RoHS & no Sb/Br)	Call TI	Level-3-260C-168HR
TMS320F2812ZHHQ	ACTIVE	BGA MICROSTAR	ZHH	179	160	Green (RoHS & no Sb/Br)	Call TI	Level-3-260C-168HR
TMX320F2812GHHS	PREVIEW	BGA	GHH	179		None	Call TI	Call TI
TMX320F2812PGFS	PREVIEW	LQFP	PGF	176		None	Call TI	Call TI

[1] The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

[2] Eco Plan - May not be currently available - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
None: Not yet available Lead (Pb-Free).
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Green (RoHS & no Sb/Br): TI defines "Green" to mean "Pb-Free" and in addition, uses package materials that do not contain halogens, including bromine (Br) or antimony (Sb) above 0.1% of total product weight.

[3] MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDECindustry standard classifications, and peak solder temperature.

PACKAGING INFORMATION

Orderable Device	Status [1]	Package Type	Package Drawing	Pins	Package Qty	Eco Plan [2]	Lead/Ball Finish	MSL Peak Temp [3]
F741814APBL-1	ACTIVE					None	Call TI	Call TI
TMS320C2810	ACTIVE					None	Call TI	Call TI
TMS320F2810PBKA	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2810PBKQ	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2811PBKA	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2811PBKQ	ACTIVE	LQFP	PBK	128	90	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2812GHHA	ACTIVE	BGA	GHH	179	160	None	Call TI	Level-3-220C-168HR
TMS320F2812GHHAR	ACTIVE	BGA	GHH	179	1000	None	Call TI	Level-3-220C-168HR
TMS320F2812GHHQ	ACTIVE	BGA	GHH	179	160	None	Call TI	Level-3-220C-168HR
TMS320F2812PGFA	ACTIVE	LQFP	PGF	176	40	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2812PGFQ	ACTIVE	LQFP	PGF	176	40	None	CU NIPDAU	Level-2-220C-1YR
TMS320F2812ZHHA	ACTIVE	BGA MICROSTAR	ZHH	179	160	Green (RoHS & no Sb/Br)	Call TI	Level-3-260C-168HR
TMS320F2812ZHHQ	ACTIVE	BGA MICROSTAR	ZHH	179	160	Green (RoHS & no Sb/Br)	Call TI	Level-3-260C-168HR
TMX320F2812GHHS	PREVIEW	BGA	GHH	179		None	Call TI	Call TI
TMX320F2812PGFS	PREVIEW	LQFP	PGF	176		None	Call TI	Call TI

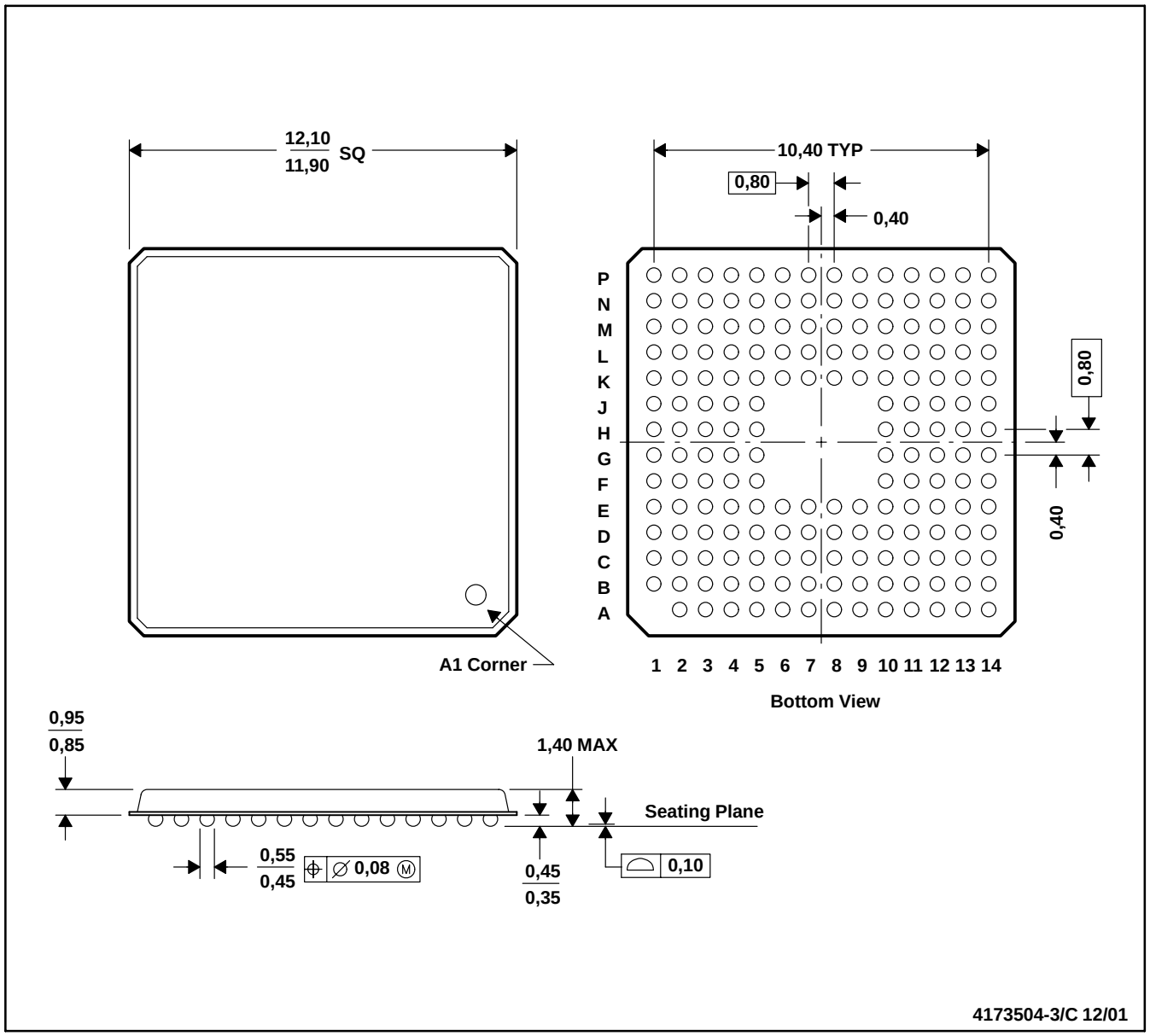
[1] The marketing status values are defined as follows:
ACTIVE: Product device recommended for new designs.
LIFEBUY: TI has announced that the device will be discontinued, and a lifetime-buy period is in effect.
NRND: Not recommended for new designs. Device is in production to support existing customers, but TI does not recommend using this part in a new design.
PREVIEW: Device has been announced but is not in production. Samples may or may not be available.
OBSOLETE: TI has discontinued the production of the device.

[2] Eco Plan - May not be currently available - please check http://www.ti.com/productcontent for the latest availability information and additional product content details.
None: Not yet available Lead (Pb-Free).
Pb-Free (RoHS): TI's terms "Lead-Free" or "Pb-Free" mean semiconductor products that are compatible with the current RoHS requirements for all 6 substances, including the requirement that lead not exceed 0.1% by weight in homogeneous materials. Where designed to be soldered at high temperatures, TI Pb-Free products are suitable for use in specified lead-free processes.
Green (RoHS & no Sb/Br): TI defines "Green" to mean "Pb-Free" and in addition, uses package materials that do not contain halogens, including bromine (Br) or antimony (Sb) above 0.1% of total product weight.

[3] MSL, Peak Temp. -- The Moisture Sensitivity Level rating according to the JEDECindustry standard classifications, and peak solder temperature.

MPBG058B – JANUARY 1998 – REVISED MAY 2002

GHH (S–PBGA–N179) **PLASTIC BALL GRID ARRAY**



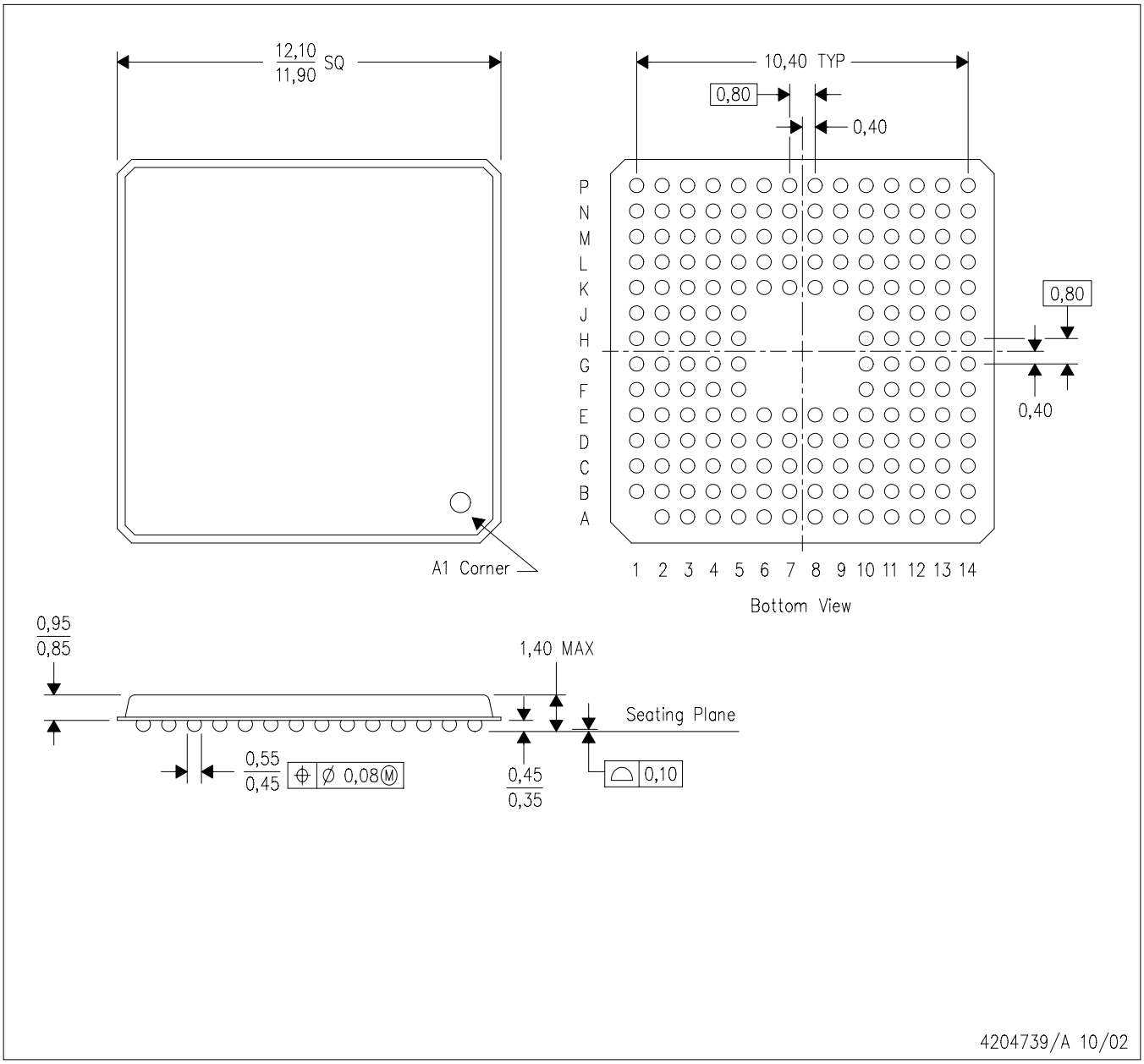
NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. MicroStar BGA™ configuration.

MicroStar BGA is a trademark of Texas Instruments.

ZHH (S–PBGA–N179)

PLASTIC BALL GRID ARRAY

MPBG349 - MARCH 2003

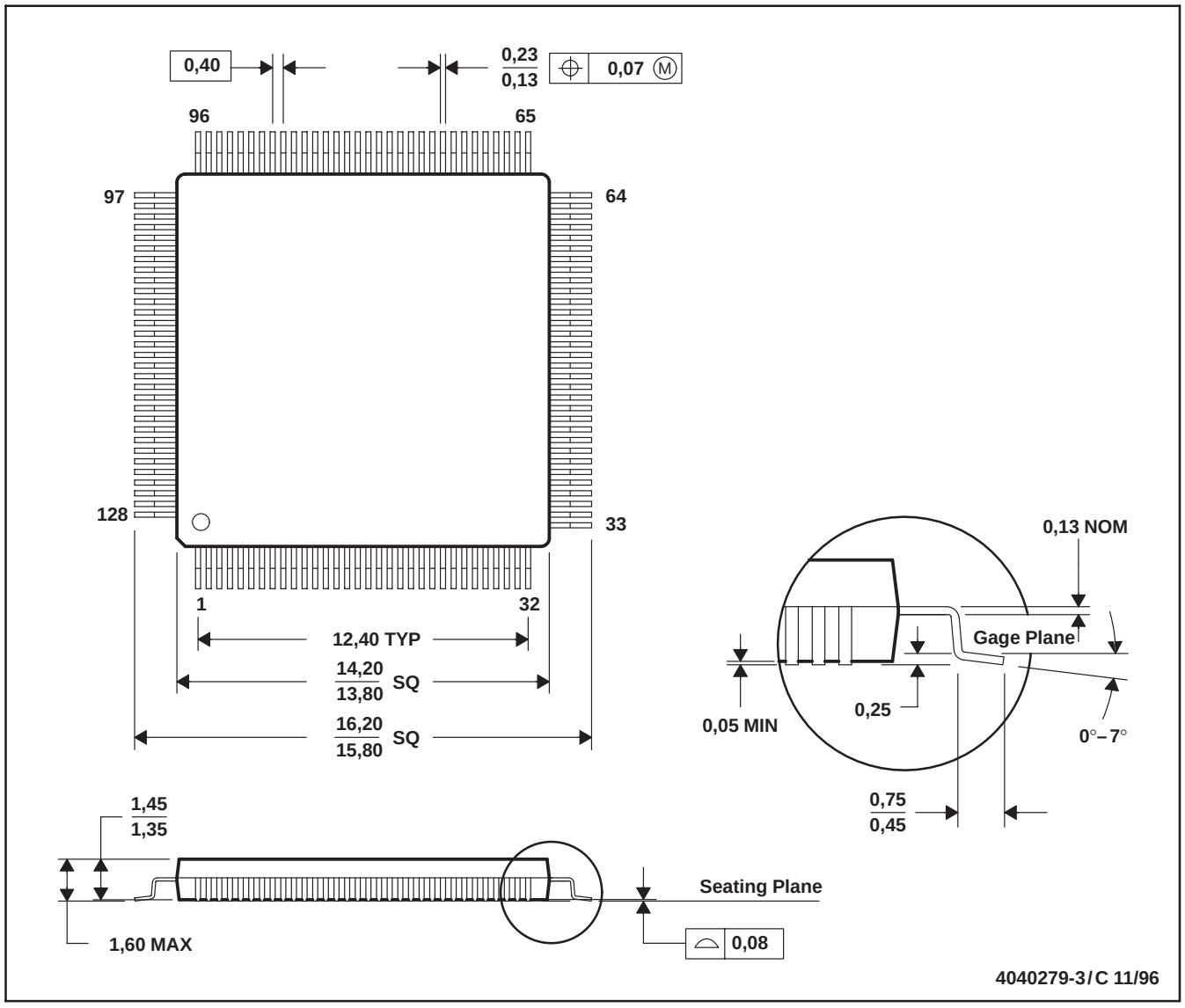


NOTES:
A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Micro Star BGA™ configuration.
D. This package is lead–free.

MicroStar BGA is a trademark of Texas Instruments

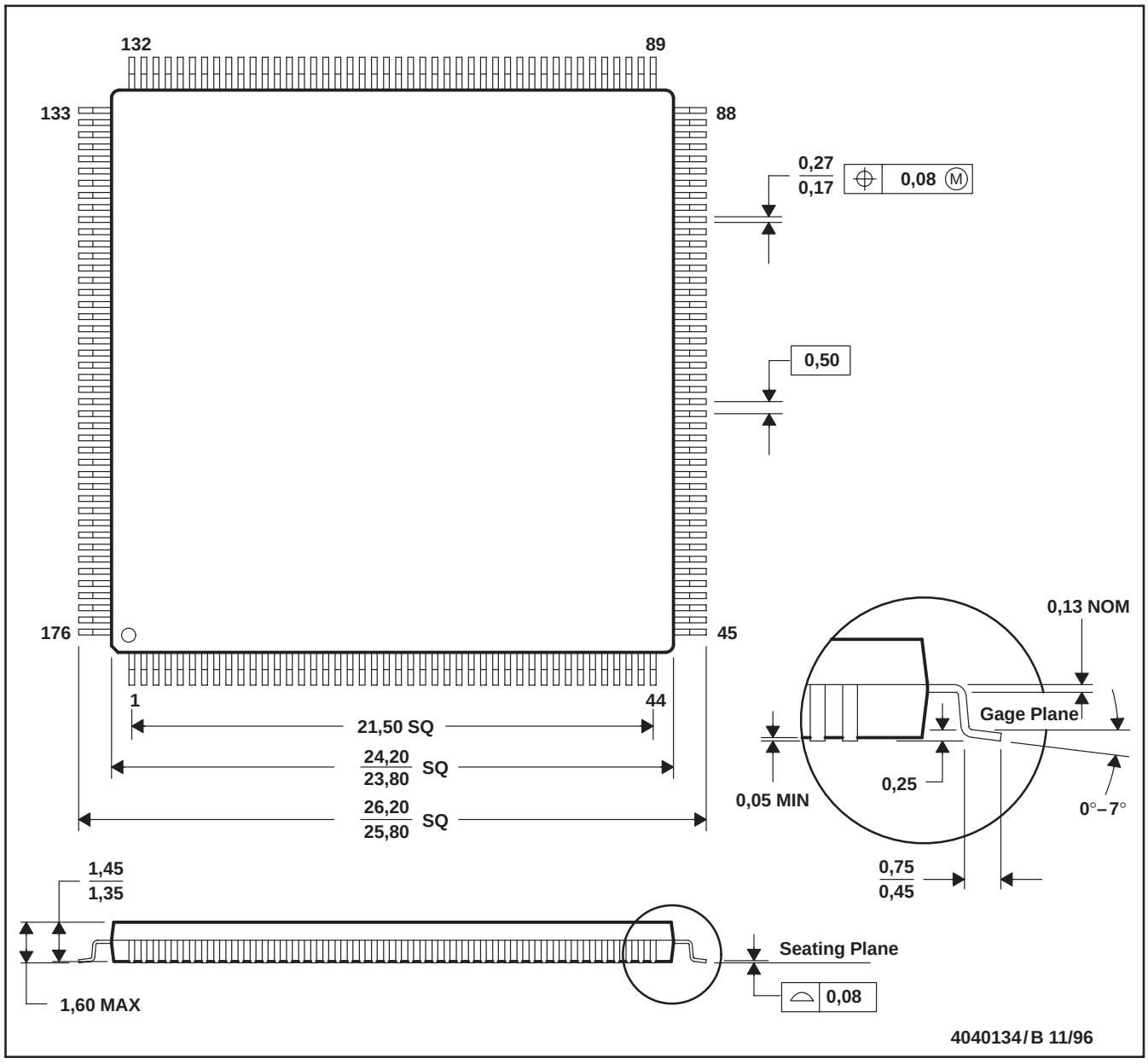
SCALE	SIZE		REV	SHEET
NONE	A	4204739	A	2/3

PBK (S-PQFP-G128) **PLASTIC QUAD FLATPACK**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS-026

PGF (S-PQFP-G176) **PLASTIC QUAD FLATPACK**



NOTES: A. All linear dimensions are in millimeters.
B. This drawing is subject to change without notice.
C. Falls within JEDEC MS-026