UNIVERSITY OF NATAL

MODULAR MECHATRONIC CIM CONTROL FOR INTERNET MANUFACTURING

Johan-Gerhard Potgieter

MScEng

School of Mechanical Engineering

University of Natal

Durban

Submitted in fulfilment of the academic requirements for the degree of Doctor of Philosophy in Engineering in the School of Mechanical Engineering, University of Natal.

December 2002

The author hereby states that this entire thesis, unless specifically indicated otherwise, is his own original work, and has not been submitted in part or in whole to any other University. This dissertation records the work completed by the author at the Department of Mechanical Engineering at the University of Natal between January 1999 and November 2001. The work contained herein has not been submitted for a degree at any other university.

J Potgieter

To my Son Jordan

ABSTRACT

Mechatronics encompass a holistic approach to the design, development, production, maintenance and disposal of complex engineering systems, products and processes. The control and modelling of the manufacturing process are carried out in a networked environment allowing for realistic real time control and simulation. This is achieved through the declarative definition of Computer Integrated Manufacturing (CIM) components, the standardisation of CIM interfaces and the object-orientated approach to model development and data management. The development of the Modular Mechatronic CIM control system is aimed at intelligently scheduling, controlling and monitoring manufacturing processes in real-time over Internet capable networks.

Modular Mechatronics is an alternative design approach that requires the decomposition of a project into separate modules, identifiable by their individual mechatronic functionality. Modular Mechatronic control for Internet manufacturing produces an efficient and effective solution for CIM processes. This approach allows a remote user to monitor and control CIM processes in real time over the Internet allow for a supervisory control structure to control and manage these processes. The modular mechatronic design approach has been applied to the development of the CIM Internet control system, to optimise the overall function of the CIM system.

A flexible, low cost Modular Mechatronic design approach was used to develop the CIM architecture and computer interface network, which served as the backbone of the Modular Mechatronic CIM control system. The modular designed control system was used to control CIM components in real time over the Internet. The Modular Mechatronic building block development allows for future integration of other CIM components.

ACKNOWLEDGEMENTS

This work presented in this thesis was carried out under the supervision of Prof G Bright of the School of Mechanical Engineering at the University of Natal, Durban. I wish to thank Prof Bright for his enthusiastic supervision of this research and the friendly guidance and constant support he has offered me throughout my studies.

I also wish to thank:

- My wife, Kerry, for her inspiration and enthusiasm that has helped me attain my goals.
- My mother, for her love, unfailing loyalty and absolute support.
- My sister, Anwyn, for all her love.
- My extended family and friends for their support through good times and bad.
- My colleagues, Dr N S Tlale and Dr J R S Mayor, for the friendly and stimulating working environment they created.
- Miss Wendy Jannsens for all her assistance during the project.
- Mr Ben Vorster and staff of the Mechanical Engineering Workshop, for their technical expertise and assistance.
- The National Research Foundation (NRF) and the University of Natal, for their financial support.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1

1 INTRODUCTION

Multimedia works well in the computerised manufacturing environment mainly because it is cost-effective, convenient and the results can be measured [Rahman et.al.]. There is a need for enterprise integration due to the fact that modern manufacturing operations are more decentralised in nature. This may be achieved using multimedia capable computer networks, especially in the process of integrating various functional areas such as marketing, design and planning, production and distribution within the enterprise [Rahman et.al.]. To succeed, the Internet-based multimedia products have to be based on standard products including transceivers, network interface cards, gateways, small servers, sensors and controllers, and a widely accepted operating system.

Some functional areas of manufacturing that have been considered for Internet uses are Computer Integrated Manufacturing (CIM), Flexible Manufacturing Systems (FMS), Computer Aided Design (CAD), Automated Storage and Retrieval Systems (AS/RS), and Robot control and scheduling. When using a modular design approach for each component of the manufacturing process, control and implementation of an Internet-based manufacturing process is significantly simplified. In CIM, the traditionally separate functions of research and development, design, production, assembly, inspection, and quality control are linked. Consequently, integration requires that quantitative relationships among product designs, materials, manufacturing processes and equipment capabilities, and related activities be well understood [Kalpakjian et.al.]. The effectiveness of a CIM system depends greatly on the presence of some large-scale, integrated communications system involving computers, machines, and their controls. Each component of the CIM system must be seen as an actuator, broken down into its most simplistic components. A microprocessor controlled actuators within a CIM cell consist of sensors, motors and linear/rotary actuators capable of position feedback and speed control. All these CIM cell components are connected to a PC-based controller which connects to a host controller as in Figure 1.1.

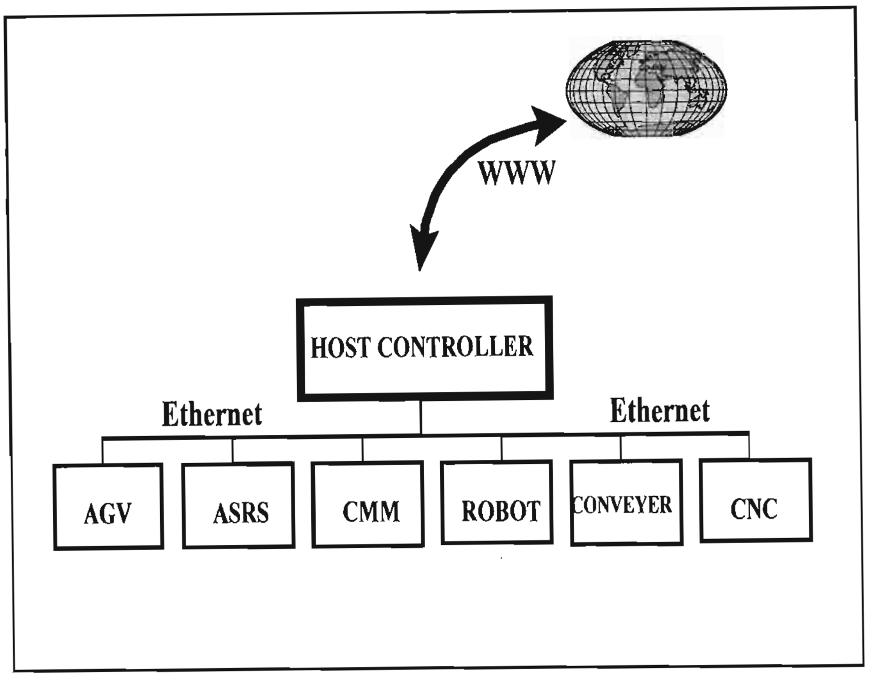


Figure 1.1. CIM Architecture.

On a primary level these CIM components are controlled using modular mechatronic principles. Mechatronics encapsulate the integration of electronics, control engineering and mechanical engineering, where there is a trend towards the development of modular actuating systems.

1.1 PROJECT BACKGROUND AND OBJECTIVES

The scope of this research addresses the topics of advanced manufacturing technologies, specifically computer integrated manufacturing, and Modular Mechatronics.

The modular mechatronic Internet-based manufacturing system was designed to be integrated into the CIM system being developed at the School of Mechanical Engineering at the University of Natal, Durban, South Africa. The developed technology would have to be PC-based, whilst being cost-effective, making it accessible to all industrial sectors and institutions of higher learning.

The primary objective of the research project was the design and implementation of a low-cost flexible Internet controlled manufacturing environment. Specific objectives include:

1. The design and development of the necessary power electronics required to operate the motion systems of the conveyer material handling system and PC-based PUMA industrial robot, using a modular mechatronic approach.

2. The design and development of the necessary mechatronic feedback devices required to monitor the motion systems of the conveyer material handling system and PC-based PUMA industrial robot, using a modular mechatronic approach.

3. The design and development of the technologies for Internet applications.

4. The design and development of a PC-based controller for a conveyer material handling system.

5. The design and development of a PC-based robot controller.

6. The integration of the PC-based robot and conveyer material handling technologies into a CIM system for Internet control.

1.2 NEW CONTRIBUTIONS

The research discussed in this dissertation makes a meaningful and original contribution to the ongoing research in the fields of Internet manufacturing technologies and flexible, low cost modular mechatronics.

1.2.1 Internet Manufacturing Technologies

The next wave of the Internet has already started to build. In this wave, everything electric will be connected. That means everything in the manufacturing enterprise will be part of an overall network that's based on Internet protocols. The new connectivity and communications tools will boost productivity, profits, speed to market, and flexibility, for those manufactures able to change.

In Engineering today there is a rapid change taking place, where information technology (IT) is now becoming a new challenge for the designer. IT can provide information for the designer, and recently knowledge management systems have been developed to provide greater assistance [Counsell et.al.]. Despite the technical revolution few IT software tools have been developed which provide designers with a usable methodology for requirements capture, conceptual design and control.

Web-based manufacturing has captured the interest of researchers and developers. The World Wide Web can be used as the infrastructure for teleoperation. Manual closed-loop control, where the human operator forms part of the control loop is the earliest and most studied form of teleoperation [Taylor et.al.]. At its simplest, communication is through mechanical linkages and feedback is by direct viewing.

Where there is significant time delay in communication, instability occurs and manual closed-loop control is no longer suitable. One technique that can improve operator performance and avoid instability problems is a shared or supervisory control scheme, where the control of the robot or device is shared between a local control loop and the human operator. World Wide Web teleoperation is a concept that involves the control of the remote device from within an Internet application.

Connectivity would then become a feature of everything within an enterprise, every device will be connected to a network. And that network will be based on the Internet or a corporate intranet, a wide-area network (WAN) or virtual private network (VPN) based on TCP/IP (Transmission Control Protocol/Internet Protocol), the computer protocols of the Internet.

The major difference will be that they won't necessarily be using a computer. Once every device is connected, we'll be working online through other devices, including wireless phones and also through the manufacturing machines and systems that are connected. "This will truly empower the knowledge worker on the shop floor by giving him or her access to information on machinery maintenance, repairs, upgrades, process improvements or whatever else he or she needs access to at any time, right at the system that needs repairs," says Jim Fall, president and CEO of Manufacturing Data Systems, Inc. (MDSI), an Ann Arbor, Michigan firm that develops and implements motion control software applications.

Commerce over the Internet has grown to $101 billion in 1998 that is, actual sales of products and services to consumers or businesses over the Web, according to a study by Internet Indicators. This sector has also been growing at an incredible rate of over 50 percent per year. Industry analysts expect e-commerce in the business-to-business sector to explode in the next few years.

There are also more computer and Internet-enabled devices available on the market, from industrial motion controllers to vision systems to medical instruments to automatic teller machines. These embedded electronics were originally developed to control one small set of functions, and to communicate with a single control system. As technology developed and the Internet became an indispensable part of almost every manufacturer in North America, users started to demand more functions and wider connectivity for such devices. Hence the growth of embedded Web browsers.

Embedding Web connections into a wide range of non-computer devices requires an operating system that can fit in the restricted memory of a cell phone or process control system. Currently, there are three main contenders: Windows CE, embedded Java and a new version of OS/2.

Microsoft's Windows CE is a small operating system for portable devices that allows communications with Windows-based personal computers. The first devices to use Windows CE were palm-top computers, but a growing number of consumer electronics such as game systems, set-top boxes, pagers and Web-ready cell phones use embedded Windows CE.

Sun Microsystems embedded Java technology uses the Java Virtual Machine running on top of the embedded device's operating system. Java is platform independent, yet allows a range of high-level functions with relatively small amounts of coding. And because it's platform independent, the same type of functionality can be given to different products. It's being used in products from mobile phones to printers and network switches, medical instruments and industrial process controllers.

IBM is almost ready to release an embedded Web browser called NetDiver, which is based on Sun's Java technology. NetDiver takes on 700 kilobytes of memory, plus another four megabytes of RAM for Web caching. Running on an embedded version of the OS/2 operating system, it's a sign of a new possible direction for the development of this alternative OS to Windows. It could be found soon on medical instruments, handheld computer terminals used in warehousing and manufacturing, process and machine control systems as well as wireless phones and automated teller machines.

Within manufacturing concerns Enterprise Asset Management (EAM) and Product Lifecycle Management (PLM) are communicating with each other, and sharing data. Increasingly, this is done through Internet connections.

The emergence of the Application Service Provider is another signal that the Internet is becoming the computer software platform, more than the type of operating system or computer manufacturer. In the ASP model, the user or software customer doesn't own the software, but rather "rents" it. The application stays on the developer's computer server, and the customer connects to it through the Internet.

The Internet and its communications protocols have a number of advantages. "IP is imperfect, and it's everywhere," says Al Smith, vice-president of software development at Bluestone, which develops technology that enables ASPs (Active Server Pages) and ASP software. "Perfect technologies rarely get out of the lab," he explains, because to be perfect they typically only fit a very limited set of criteria. The Internet, however, is easy to use and widely accepted. Because it's everywhere, there is a great wealth of solutions, and a lot of expertise to help users implement and troubleshoot what they're trying to do on the Internet.

The Internet is easy to use and widely accepted. Because it's everywhere, there is a wealth of solutions, and a lot of expertise to help users implement and troubleshoot what they're trying to do online.

"The biggest reason for the use of the Internet for machine-to-machine communication is cost," says Don Thompson, a consultant with Deloitte Consulting in Toronto. Since most corporate IT infrastructures already use intranets, it makes sense to base new communications efforts on it. They don't need to add new physical networks or hire more experts to maintain a different type of network.

The Internet is being used even at the shop floor level. For instance, OpenCNC 5.1 from Manufacturing Data Systems, Inc. connects computer numerical control devices (CNCs) through intranets or the Internet, production planning, or maintenance systems. "Fieldbus level networks aren't oriented for communication between platforms, or between buildings or plants," says Jim Fall, CEO of MDSI. Basing shop-floor data communication on IP, on the other hand, makes it simpler to connect those systems with enterprise management systems.

Basing machine-to-machine communications on IP means setting up protocols and security in software. This makes it so much easier to allow certain people access to certain sections of the system, according to whatever criteria you like; maintenance people need certain parts of the data, but not others; operators would be able to access a limited number of devices; managers would be allowed to monitor, but not change anything, etc.

Furthermore, IP is robust enough to handle the communications. It supports real-time communication (as opposed to batch processes) and it's faster and more robust than the virtual private networks or value-added networks that came before it. As Jim Fall points out, it has bandwidth to spare.

The Internet is where all the exciting developments are happening. Technologies such as Java and ActiveX allow developers to build a wide range of tools, controls and functions to address almost any manufacturer's needs.

New technologies such as the Extensible Markup Language (XML) are now making it easier to share data between different application programs, and to set up computers to take actions based on criteria for instance, to order supplies when inventories reach a critical low point.

With the Internet used within and between enterprises, communicating between management, production planning, execution, maintenance, sales, accounting, shipping and purchasing activities, the totally automated, "24 hour, lights-out factory" might not be far off. It has already arrived in some plants: the manufacturing operation that can be operated completely by remote control, with no one on the site other than maintenance personnel when needed. Using Internet protocols makes the development of new communications simple, fast and relatively cheap. And while the Internet seems popular now, it's going to grow even faster as more Web-ready devices hit the markets.

1.2.2 Modular Mechatronics

The research presented in this dissertation employs a design methodology referred to as, *Modular Mechatronics*, which optimises the development of the Internet-based manufacturing environment. The modular mechatronic design methodology is a rapid system development tool integrating system building blocks comprising of established mechatronic principles and systems. The Internet-based CIM control system presented in this research was designed and implemented using a low cost modular mechatronic rapid system development tool and therefore provided a practical design strategy for modern mechatronic systems.

Manufacturing is a complex system composed of many diverse physical and human elements, some of which are difficult to predict and control [Kalpakjian et.al.]. Ideally a manufacturing system should be represented by mathematical and physical models, which shows the nature and extent of interdependence of variables involved. In a manufacturing system, a change or disturbance anywhere in the system requires that it adjust itself system wide in order to continue functioning efficiently. For CIM systems these models include process flow, control information flow, data flow and cost. Similarly, the demand for a product may fluctuate randomly and rapidly, on account of its style, size, or production volume. Modelling such a complex system can be difficult due to a lack of comprehensive or reliable data on many of the variables involved. The standardisation of the design process of the CIM architecture simplifies the design of the manufacturing system and minimises the systems complexity.

[Honekamp et.al.] Proposes a modular design approach that requires the functional decomposition of the project into mechatronic functional modules and implementing the hierarchical structuring of the mechatronic functional modules to develop mechatronic systems. The base level of the mechatronic functional modules is defined to be an encapsulated mechatronic system comprising of a supporting structure, actuator/sensor groups and controller elements [Honekamp et.al.].

The modular mechatronic design approach allows the project definition to be decomposed into independent sub-tasks according to the individual mechatronic functionality of each module. These sub-tasks define the mechatronic functional modules and therefore represent encapsulated mechatronic systems.

The success of the modular mechatronic design methodology is dependant on the computer-based integration of the various functional mechatronic modules. The responsibilities of the computer-based controller can be organised in multiple levels, ranging from low-level control through supervision to general system management [Isermann]. Figure 1.2 shows the functional modular mechatronic sub-tasks required by the Internet-based CIM control system presented in this research.

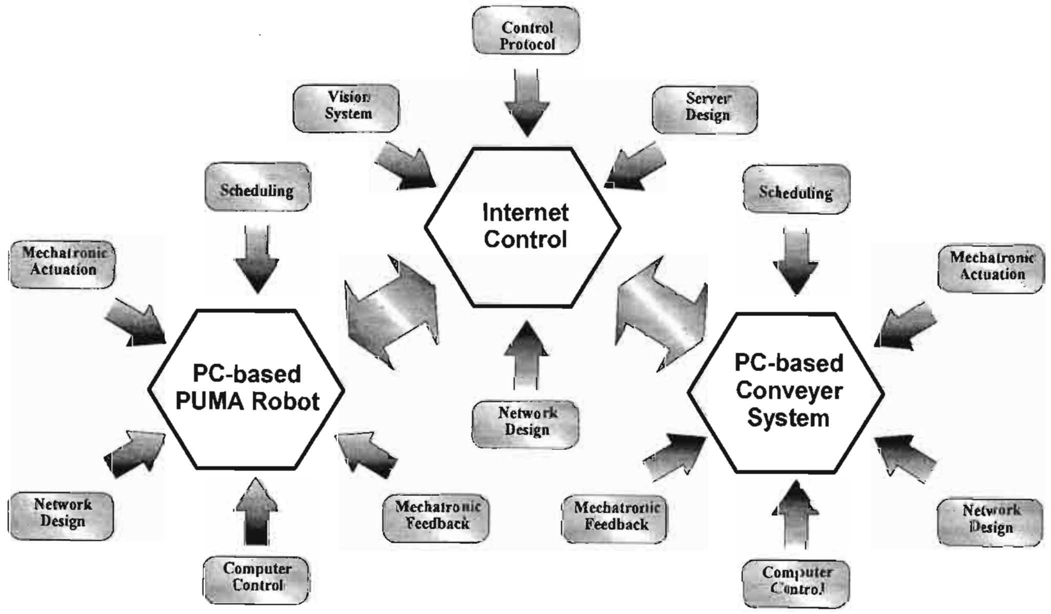


Figure 1.2. The functional modular mechatronic sub-tasks required by the Internet-based CIM control system. Note how functional mechatronic modules are shared between systems.

The integration of the modular mechatronics systems in a software domain is achieved by the development of algorithms that control the coordinated interaction of the individual sub-tasks. The modular mechatronic rapid system development tool should therefore be able to ensure the rapid development of flexible Internet-based manufacturing technologies.

1.3 RESEARCH PUBLICATIONS

1.3.1 Refereed Journals

- G. Bright and J. Potgieter. "Robotic Internet System for CIM Processes", SA Mechanical Engineer, South Africa, May 2000.

1.3.2 Refereed Conference Proceedings

- G. Bright and J. Potgieter. "PC-Based Mechatronic Robotic Plug and Play System for Part Assembly Operations", IEEE International Symposium on Industrial Electronics, Pretoria, South Africa, July, 1998.

- G. Bright and J. Potgieter. "PC-Based Modular Robotic System for Part Assembly Operations", IASTED International Conference on Robotics and Manufacturing, Banff, Canada, July 1998.

- G. Bright and J. Potgieter. "Operating System for part Assembly Operations using PC-Based Plug and Play Mechatronic Technology", 9th DAAAM International Symposium on Intelligent Manufacturing, Automation and Networking, Technical University Cluj-Napoca, Romania, October, 1998.

- G. Bright and J. Potgieter. "Flexible PC-Based Modular Mechatronic Operating Systems for Computer Integrated Manufacturing", The 15th ISPE/IEE International Conference on CAD/CAM, Robotics and Factories of the Future, Aguas de Lindoia, SP, Brazil, August, 1999.

- G. Bright and J. Potgieter. "Mechatronic Modular Robotic Internet-Based Control System for Computer Integrated Manufacturing Processes", 7th IASTED International Robotics and Applications 2000, Honolulu, Hawaii, USA, August 2000.

- G. Bright and J. Potgieter. "Mechatronic Internet-Based Control System for Computer Integrated Manufacturing Processes", International Conference on Competitive Manufacturing, COMA'01, Stellenbosch, South Africa, February 2001.

- G. Bright and J. Potgieter. "Integrated Mechatronic Control Approach for Global Manufacturing Enterprises", IASTED International Conference on Robotics and Manufacturing (RM2001), Cancun, Mexico, May 2001.

- G. Bright and J. Potgieter. "Mechatronic Control Approach for Global Internet Manufacturing", The 17th ISPE/IEE International Conference on CAD/CAM, Robotics and Factories of the Future, Durban, South Africa, July 2001.

1.4 THESIS LAYOUT

The layout of this thesis is based on the implementation of the modular mechatronics design approach of the Internet-based computer integrated manufacturing control system. The chapters present the research into each sub-task in the sequential order that reflects the design methodology of the modular mechatronics approach.

Chapter 2 introduces the field of modular mechatronics and discusses current CIM technologies. Chapter 3 covers the mechatronic principles and technologies that provide primary power (actuation) of the modular CIM components, as discussed in Chapter 2. Chapter 4 then discusses the collection of feedback information essential to the control of the CIM cell and the associated mechatronic principles used. Chapter 5 discusses the CIM teleoperation system, with the objective to design modular network and Internet systems. Chapter 6 illustrates the integration of the mechatronic modules discussed in Chapters 3,4, and 5 for control of the PC-based conveyer system. Chapter 7 discusses the design and implementation of the modular mechatronic Internet control strategy for the PC-based PUMA robot.

A performance analysis and operational review of the modular mechatronic Internet-based control system have been included in Chapter 8 and Chapter 9. Finally, Chapter 10 presents a conclusion to this thesis that discusses the meaningful and original contributions of the research presented in this thesis and offers some suggestions for future work.

CHAPTER 2

2 MODULAR MECHATRONICS FOR CIM SYSTEMS

Today, cost-effective electronics, microcomputers, and digital signal processors have brought space-age technology to appliances and consumer products [Craig]. As Figure 2.1 illustrates, mechatronics is the synergistic combination of mechanical engineering, electronics, control systems, and computers. The key element of mechatronics being the integration of these areas through the design process. In order to design and build quality precision consumer products in a timely manner, the present-day mechanical engineer must be knowledgeable (both analytically and practically) in many different areas.

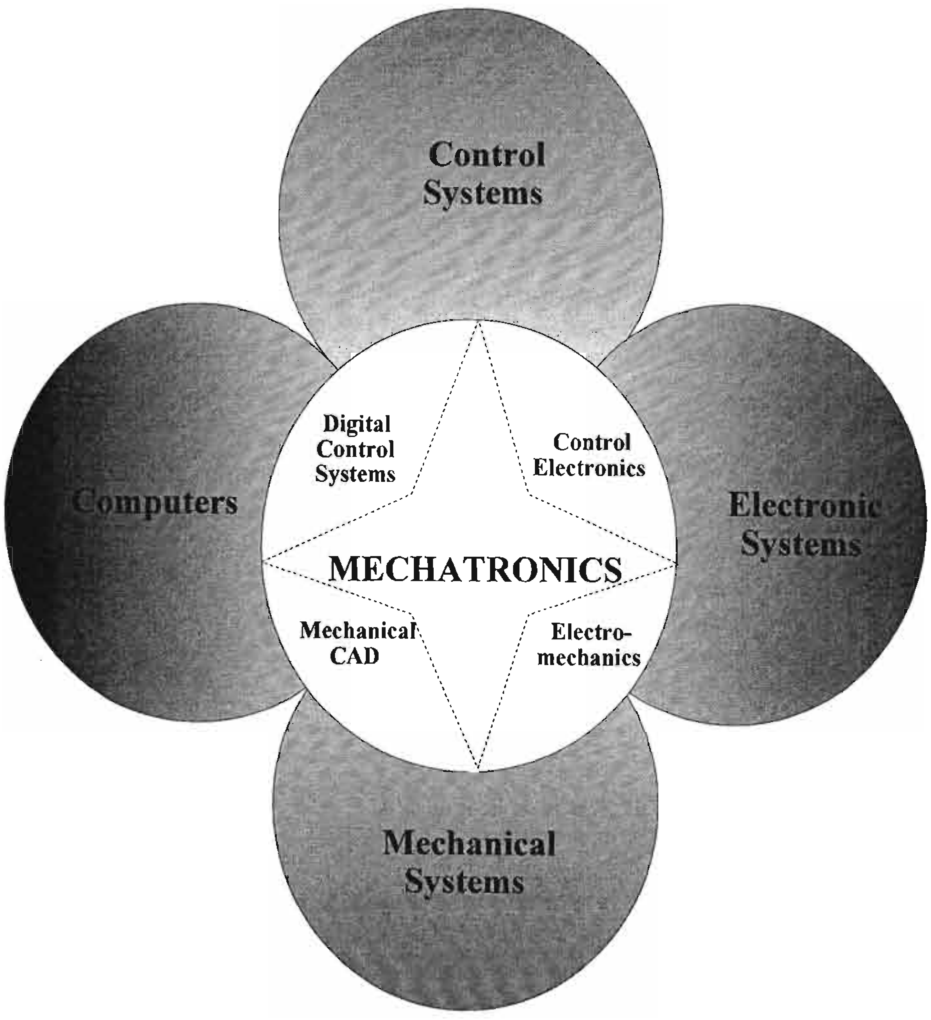


Figure 2.1. Mechatronics : synergism and integration through design.[Craig].

The ability to design and implement analogue and digital control systems, with associated analogue and digital sensors, actuators, and electronics are an essential skill for every engineer wanting to control a mechanical system. Knowledge of mechatronics helps an engineer generate more and better concepts and facilitates the communication with team members in other disciplines [Craig]. In mechatronics balance is paramount. The essential characteristic of a mechatronic engineer is a balance between two sets of skills:

1. Modelling (physical and mathematical), analysis, and control design of dynamic physical systems.
2. Experimental validation of models and analysis and understanding the key issues in hardware implementation and design.

Figure 2.2 shows the diagram of the procedure for a dynamic system investigation that emphasises this balance. Here the physical system can be an actual device or system that needs improving, or a representation of a concept being evaluated in the design process. Engineers can no longer evaluate each design concept by building and testing. It is too costly and time consuming. They must rely on the modelling and analysis of a previous design experience to evaluate each design concept. This concept gives rise to the design of modular mechatronic subsystems or building blocks.

Modular mechatronic systems can be applied to a variety of automated manufacturing operations. These operations can be extended further by including information processing functions, utilising an extensive network of interactive computers. The result is CIM, which is a broad term describing the computerised integration of all aspects of design, planning, manufacturing, distribution, and management [Kalpakjian et.al.]. Computer Integrated Manufacturing is a methodology and a goal, rather than an assemblage of equipment and computers. Implementation of CIM in existing plants may begin with modules in various phases of operation. This provides an ideal platform for the implementation of the modular mechatronic design methodology.

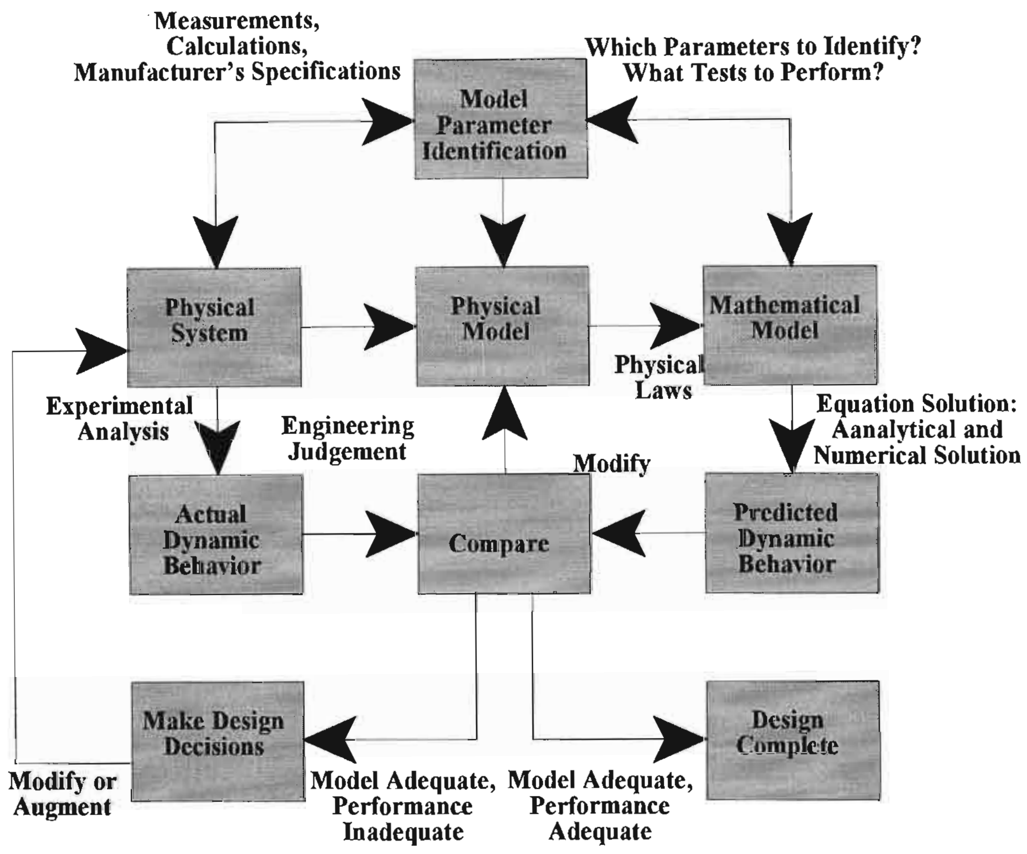

Measurements,
Calculations,
Manufacturer's Specifications
Which Parameters to Identify?
What Tests to Perform?
Model Parameter Identification
Physical System
Physical Model
Mathematical Model
Physical Laws
Experimental Analysis
Engineering Judgement
Modify
Equation Solution: Aanalytical and Numerical Solution
Actual Dynamic Behavior
Compare
Predicted Dynamic Behavior
Make Design Decisions
Design Complete
Modify or Augment
Model Adequate, Performance Inadequate
Model Adequate, Performance Adequate

Figure 2.2. Dynamic system investigation [Parrish].

Although the emphasis on mechatronics should not change the fundamental content of it's related engineering disciplines, it does provide the opportunity to contemporise and enhance content, and improve sequencing of modelling and analysis, computing, electrical circuits and machines, measurements and instrumentation, microprocessors and design. Figure 2.3 shows the technical development and integration of the related engineering disciplines.

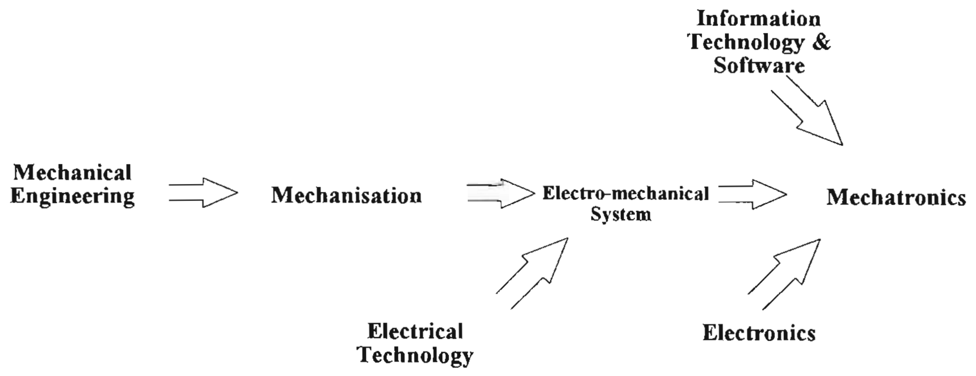


Figure 2.3. The technical development and integration of related engineering disciplines [Bradley et.al.].

If it is possible to implement and control mechatronic systems using modular building blocks, then the nature and characteristic of mechatronic systems would remain the same. Figure 2.4 shows these similar characteristics in a manufacturing system context diagram.

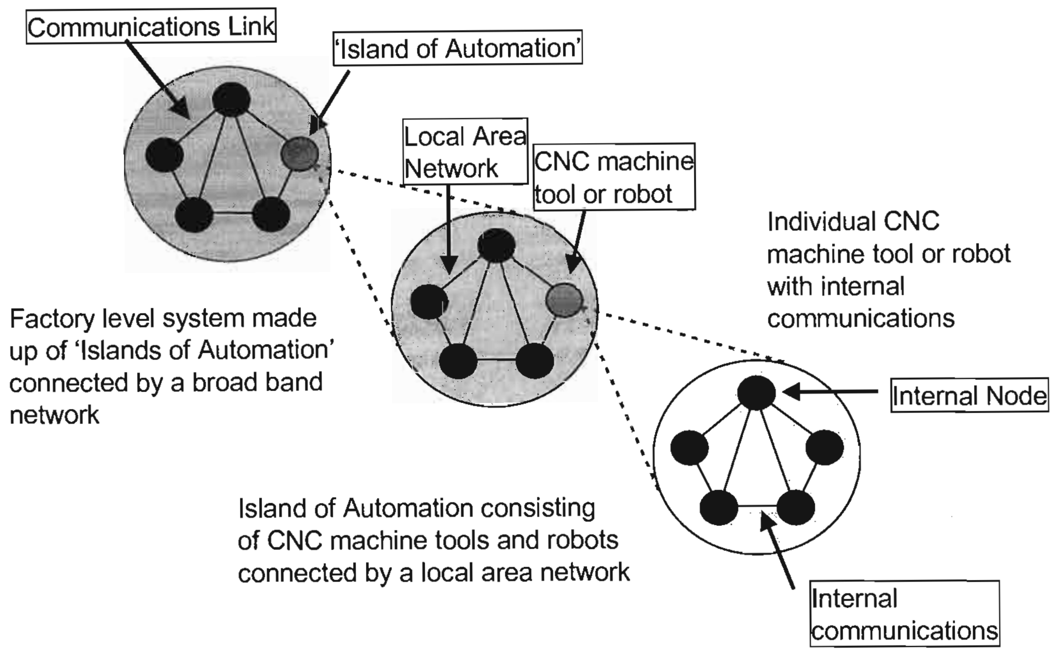

Communications Link
'Island of Automation'
Local Area Network
CNC machine tool or robot
Individual CNC machine tool or robot with internal communications
Internal Node
Factory level system made up of 'Islands of Automation' connected by a broad band network
Island of Automation consisting of CNC machine tools and robots connected by a local area network
Internal communications

Figure 2.4. Manufacturing system context diagram [Bradley et.al.].

Referring to Figure 2.4, the world of a robot joint, which is a mechatronic system, is the robot itself, while the world of the CIM may be the market. Each component of the CIM constitutes a mechatronic system within an 'Island of Automation', consisting of a variety of modular mechatronic building blocks. Figure 2.5 shows the information structure for the mechatronic CIM system. The information structure has a hierarchy. Referring to Figure 2.5, layer 4 might be a robot within the CIM system. The robot has its own process knowledge and has its own controller, responsible for primary control. Layer 3 might be a robot host controller or macro controller, controlling more than one robot. This layer contains process knowledge of the manufacturing and assembly task being performed, while the robots being controlled, have no knowledge about each others operation. Between layer 3 and layer 4 there exists a knowledge filter allowing only the relevant information to be passed between layers in the process and procedural mechatronic pyramid.

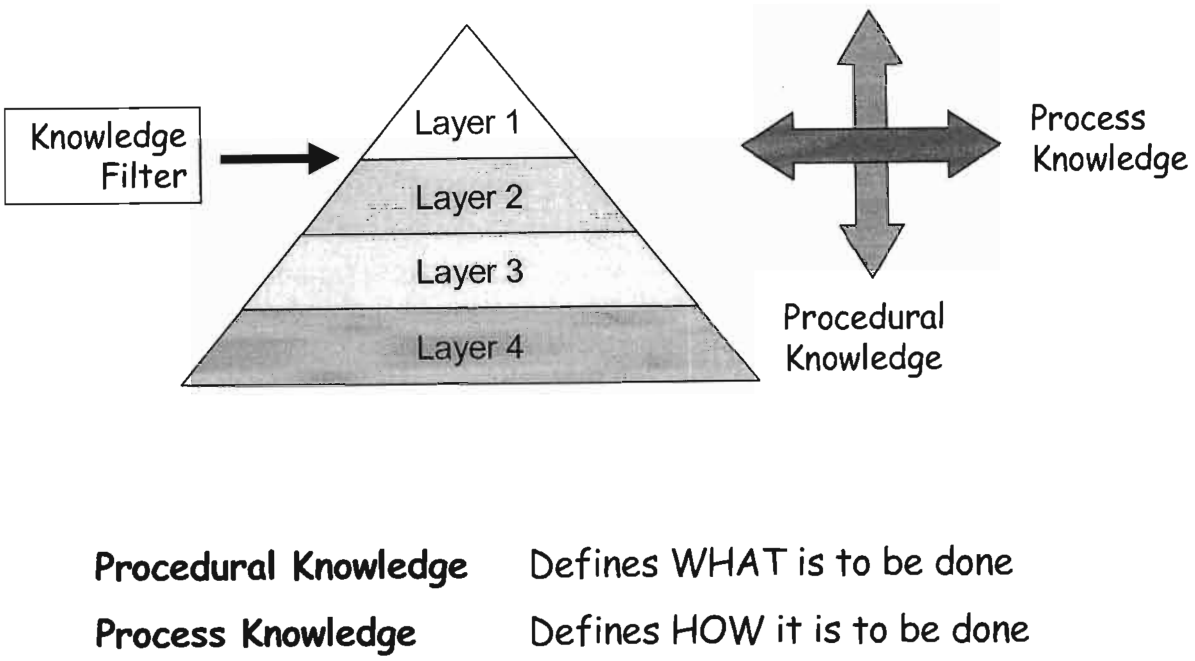

Knowledge Filter
Layer 1
Layer 2
Layer 3
Layer 4
Process Knowledge
Procedural Knowledge
Procedural Knowledge Defines WHAT is to be done
Process Knowledge Defines HOW it is to be done

Figure 2.5. The relationship between procedural and process data in a mechatronic context. Note the sharing of procedural information occurs through a knowledge filter [Bradley et.al.].

2.1 COMPONENTS OF CIM ARCHITECTURE

The integration of the automated industry is now recognised by the widely accepted label of CIM. Factory integration enables better organisation of material and information flows [Parrish]. Part of the CIM component is the application of flexible manufacturing systems (FMS) technology. CIM architecture is a framework, providing a structure of computer hardware and software for computer systems in manufacturing companies.

The automation of the direct, and some indirect, production activity of a factory is classified under the heading of advanced manufacturing technology (AMT). AMT aims to automate and integrate all functions of a factory concerned with manufacturing operations such as design, production and quality. AMT includes the automation of the manufacturing and engineering activities such as computer-aided manufacturing (CAM), CIM and computer aided engineering (CAE). Figure 2.6 maps the specialist area of CIM.

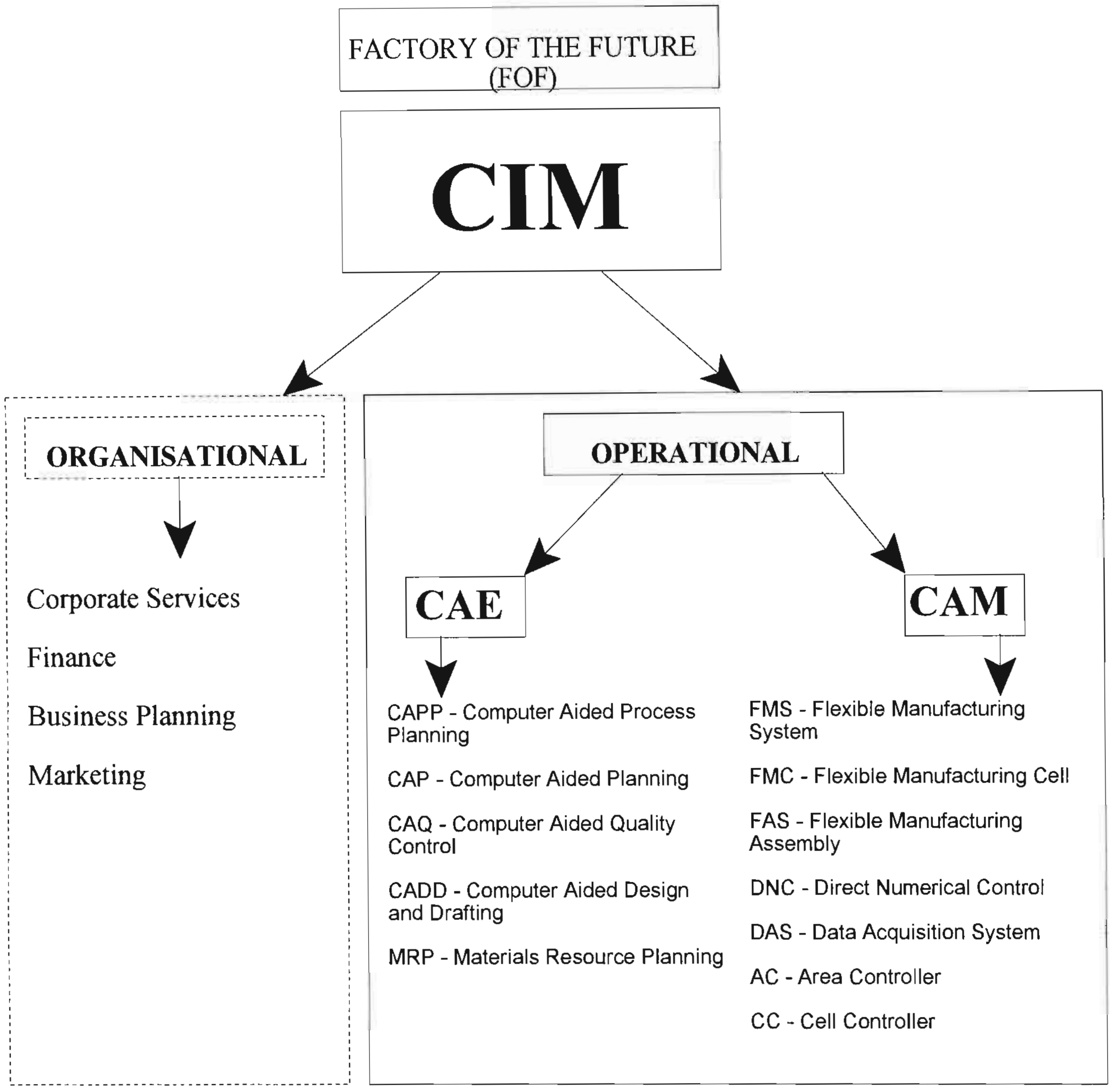

FACTORY OF THE FUTURE (FOF)
CIM
ORGANISATIONAL
Corporate Services
Finance
Business Planning
Marketing
OPERATIONAL
CAE
CAM
CAPP - Computer Aided Process Planning
CAP - Computer Aided Planning
CAQ - Computer Aided Quality Control
CADD - Computer Aided Design and Drafting
MRP - Materials Resource Planning
FMS - Flexible Manufacturing System
FMC - Flexible Manufacturing Cell
FAS - Flexible Manufacturing Assembly
DNC - Direct Numerical Control
DAS - Data Acquisition System
AC - Area Controller
CC - Cell Controller

Figure 2.6. An illustration of the specialist area of CIM [Parrish].

The activities of a CIM environment can be logically distributed into a hierarchy that operates on a data exchange network system. There are five describable levels of control or organisation (ref. Figure 2.7). Within the CIM hierarchy, host systems are designed to control groups of machines and equipment or to organise facilities or data for the equipment.

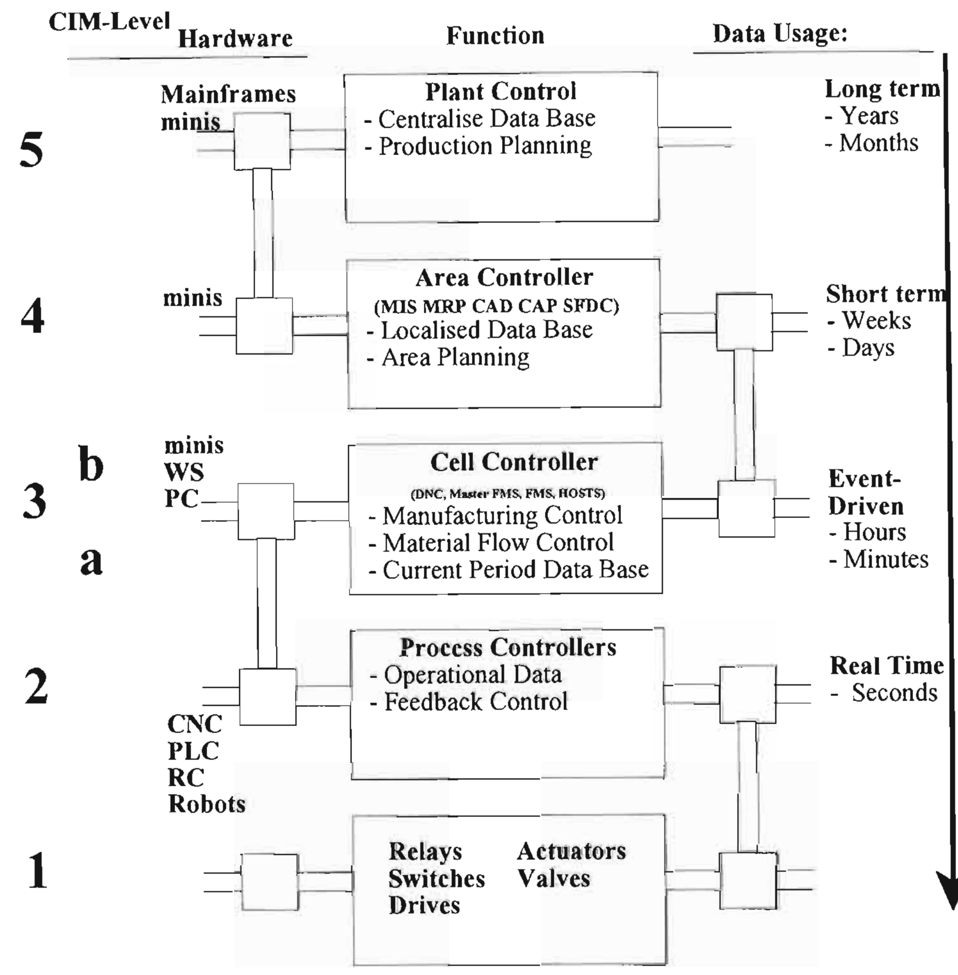


Figure 2.7. Shows the typical CIM hierarchy [Parrish].

Levels 3 to 5 organise areas of the factory based on decisions taken from data received from other levels in the hierarchy. Control is the execution and monitoring of the decisions of levels 1 and 2. These functional levels found in the factory environment are not to be confused with the seven ISO/OSI (International Standard Organisation/ open system interface) layers upon which the Manufacturing Automation Protocols (MAP) are based. The CIM hierarchy levels support the organisation of a factory to be able to carry out functions. The ISO/OSI levels enable communication between the functions found in a factory, such as the data transfer between a CAD computer and an NC programming computer. Level 1 and 2 of the CIM hierarchy, consisting of Robots, CNC, PLC etc., is where the control functions are executed. Levels 3, 4 and 5 define the organisational levels such as the FMS host and area controller. At level 1 of the hierarchy is the modular mechatronic actuators and feedback devices. Level 2 includes the controllers which enable the mechatronic devices and machines to achieve an autonomous standalone capability. The dedication of controllers to equipment or functional areas results in decentralisation of control in the hierarchy. This provides a greater total system reliability with higher system uptime. If any part of the hierarchy fails, the remainder may continue to function.

The production cell host controller is installed (hierarchical control) above the stand-alone controllers to provide the organisation and monitoring of a group of machines. This occurs at level 3a, and consists of a personal computer. The use of controller at this level is usually dependant upon the size and complexity of the manufacturing system. The CIM system in the Mechatronics and Robotics Research Group (MR^{2}G) laboratory is an example of a CIM cell. Figure 2.8 shows an overview of the CIM cell in the MR^{2}G laboratory.

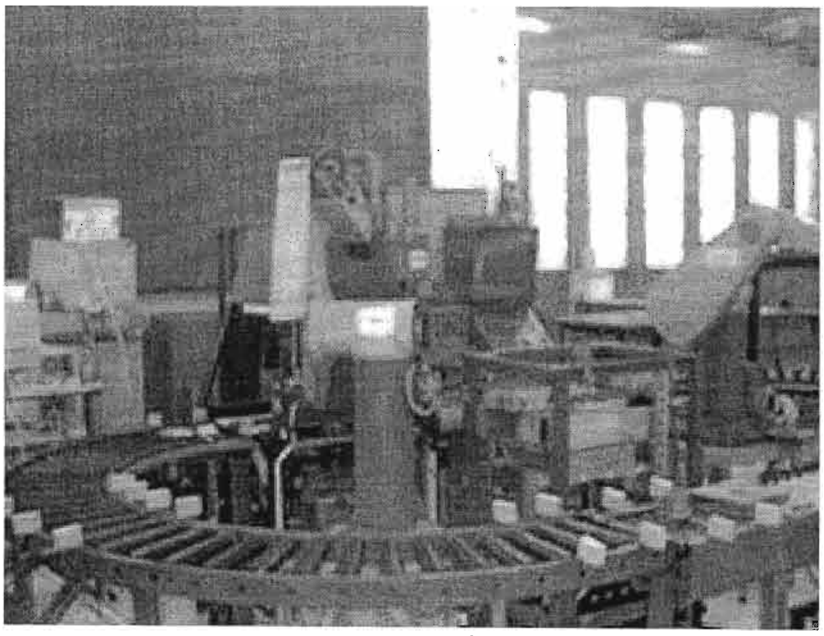


Figure 2.8. The CIM cell in the MR^{2}G laboratory.

Each of the systems within the CIM forms a single autonomous island of production (ref. Figure 2.4). Level 3b, where the host computer is often known as a coordination or master host, is only ever configured in the hierarchy when two or more level 3a production cell controllers require coordination into an even larger production island. They are grouped together under the coordination of this master host computer to form the expanded autonomous workshop.

Presented in this dissertation is the control of two different and unique material handling systems:

1. The PC-based conveyer system in the MR^{2}G laboratory is controlled by a desktop computer, which takes care of all the primary functions of the conveyer material handling system,
2. The PC-based PUMA robot is controlled by a dedicated desktop computer.

The PC-based conveyer and PUMA robot controller are connected to a host controller that forms part of a single island of automation. This control technique uses a similar group technology (GT) approach, where similar machine operations are grouped and controlled by one hierarchical cell controller.

Level 4 configures the control level for an area within the factory. The input and output of material into the area are planned at this level, as is the capacity planning of the equipment in the area. The computers and mainframes of level 5 provide the automation of the corporate functions.

2.2 CIM COMMUNICATION STRUCTURE

In order to maintain a high level of coordination and efficiency of operation in CIM, an extensive, high-speed, and an interactive communications network is required. A major advance in communications technology has been the local area network (LAN). In this hardware and software system, logically related groups of machines and equipment communicate with each other. An LAN can be very large and complex, linking hundreds or even thousands of machines and devices in several buildings. Different types of networks can be linked through *gateways*. Access control to these LANs are important, otherwise collisions can occur when several workstations transmit simultaneously. Continuous scanning of the transmitting medium is essential. In the 1970's, a carrier sense multiple access with collision system (CSMA/CD) was developed and implemented in Ethernet, which has now become the industry standard [Kalpakjian et.al.].

Three basic network topologies for an LAN's used are (ref. Figure 2.9):

1. The star topology is suitable for situations that are not subject to frequent configuration changes. All messages pass through a central station. All the messages from the central station are passed to each user station, while each individual user station communicates directly to the central station.
2. The ring topology has a configuration where all the user stations are connected in a continuous ring. The message is forwarded from one station to the next until it reaches the assigned destination. Although the wiring is relatively simple, the failure of one station shuts down the entire network.
3. In the bus topology all stations have independent access to the bus. The system is reliable and easier to connect and maintain than the other two services.

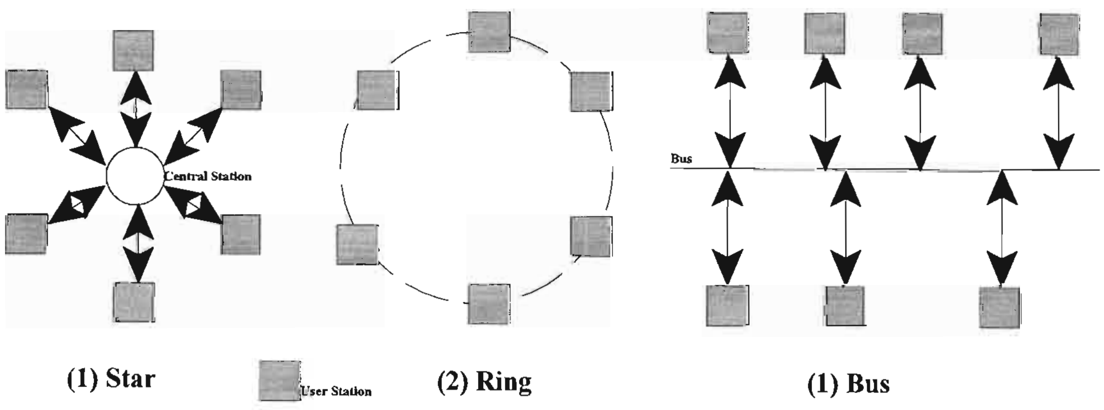


Figure 2.9. The basic types of topology for a LAN.

Each user station is seen as a module of the communication structure. The communication structure used in the modular mechatronic CIM for Internet control is based on the star topology.

Each system module computer has its own specifications and proprietary standards and is not allowed to communicate beyond the cell, unless through a host controller in the network. This situation created islands of automation. A major trend is the use of Internet tools within a company to link all departments and functions into a self-contained and fully compatible intranet. The linking of the islands of automation formed part of the Intranet control structure within the MR^{2}G laboratory. The host controller allows the outside world to connect to the Intranet through the use of Internet specific tools.

2.3 THE MR^{2}G CIM LABORATORY

As research into CIM advanced, the technologies required to control other aspects of the manufacturing process were developed. Control and communication strategies were developed, enhancing the CIM process.

Before any CIM cell can be assembled, a clear objective for the CIM function and areas of research must be determined. A clear distinction must be made whether the equipment in the CIM will drive the research or will the research drive the equipment. In the MR^{2}G laboratory the latter have been achieved.

Figure 2.10 shows the MR^{2}G CIM cell in the School of Mechanical Engineering, at the University of Natal, Durban.

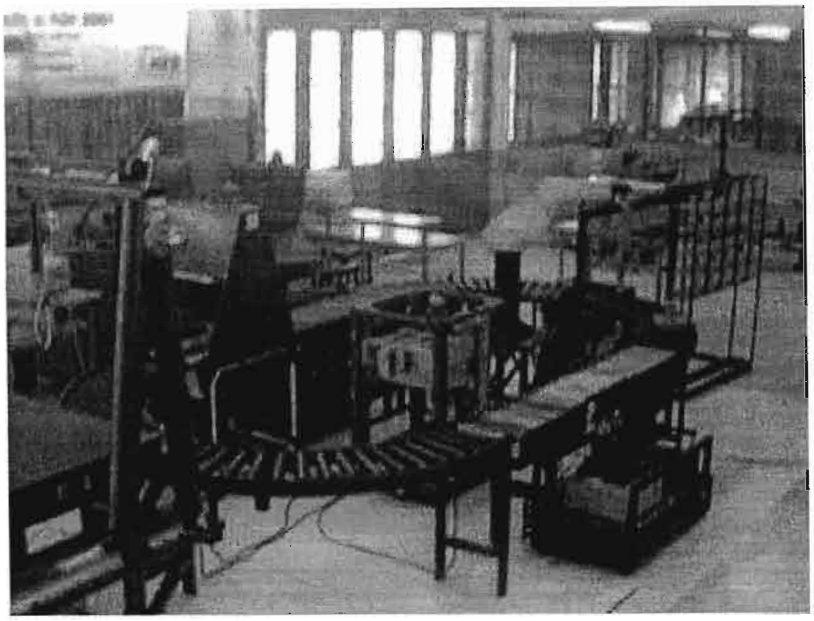


Figure 2.10. The MR^{2}G CIM cell in the School of Mechanical Engineering.

The CIM cell incorporated s variety of manufacturing disciplines and consisted of:

- A PC-Based PUMA 560 series robot. The PUMA robot was a six-degree of freedom industrial robot. The PUMA robot original dedicated controller was discarded as it was nonoperational, and replaced with a PC-based controller. The function of the robot within the CIM cell is material handling and part placement. Figure 2.11 shows the PC-based PUMA robot.

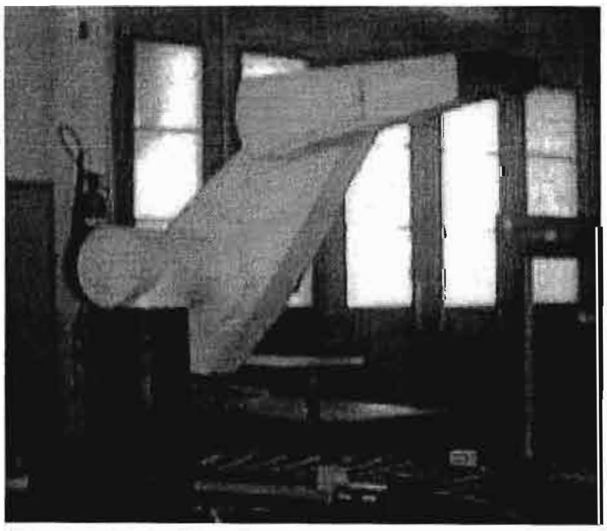


Figure 2.11. The PC-based PUMA 560 series robot.

- An Automated Guided Vehicle (AGV). The AGV formed part of the material handling system of the CIM cell, and was designed and built in the MR^2G laboratory. The AGV has a path tracking guidance system with the ability to deliver materials to the conveyer and automated storage and retrieval system(AS/RS). Figure 2.12 shows the AGV integrating with the conveyer system.

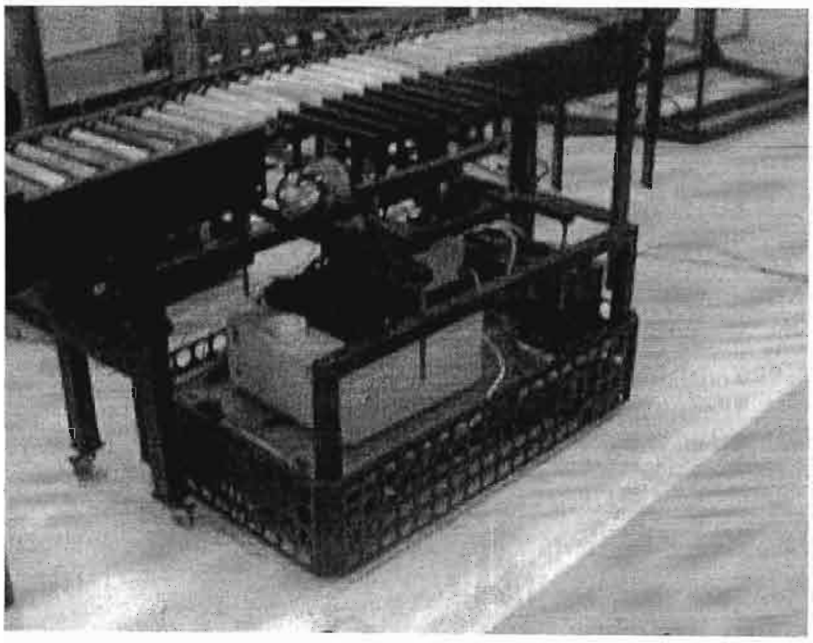


Figure 2.12. The AGV aligning with the conveyer system in the CIM cell

- A modular PC-based conveyer system. The conveyer was configured as an oval track and was responsible for the primary material handling functions of the CIM cell (ref. Figure 2.10). The conveyer track consisted of six re-configurable sections, each wired to a central hardware cabinet. The hardware cabinet contained the power supplies and the *power relay modules* (ref. Chapter 3) of each section of conveyer. Figure 2.13 shows the hardware cabinet.

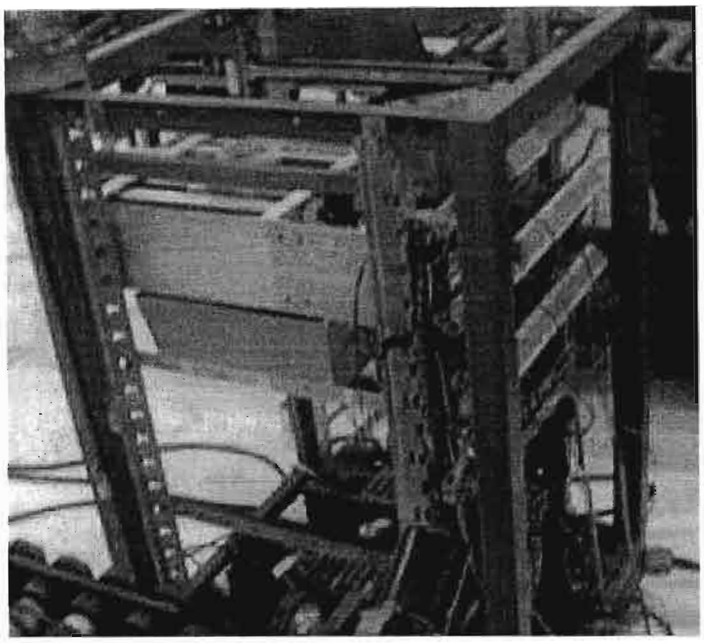


Figure 2.13. The CIM cell conveyer hardware cabinet.

- An Automated Visual Inspection Apparatus (AVIS). The apparatus was responsible for the visual inspection and verification of machined and assembled parts in the CIM cell. The AVIS was a computer-based technology integrating modern machine vision technology. Figure 2.14 shows the AVIS.

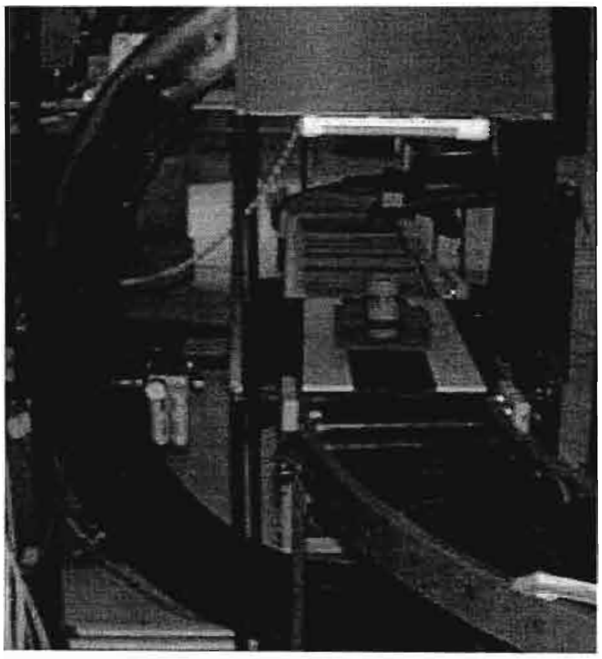


Figure 2.14. The Automated Visual Inspection Apparatus.

- The Coordinate Measuring Machine (CMM). The CMM is a three-axis contact-sensor measuring device. The contact-sensor-based CMM utilised an inspection probe that makes physical contact with the part during the inspection process. Figure 2.15 shows the CMM in the MR^{2}G laboratory.

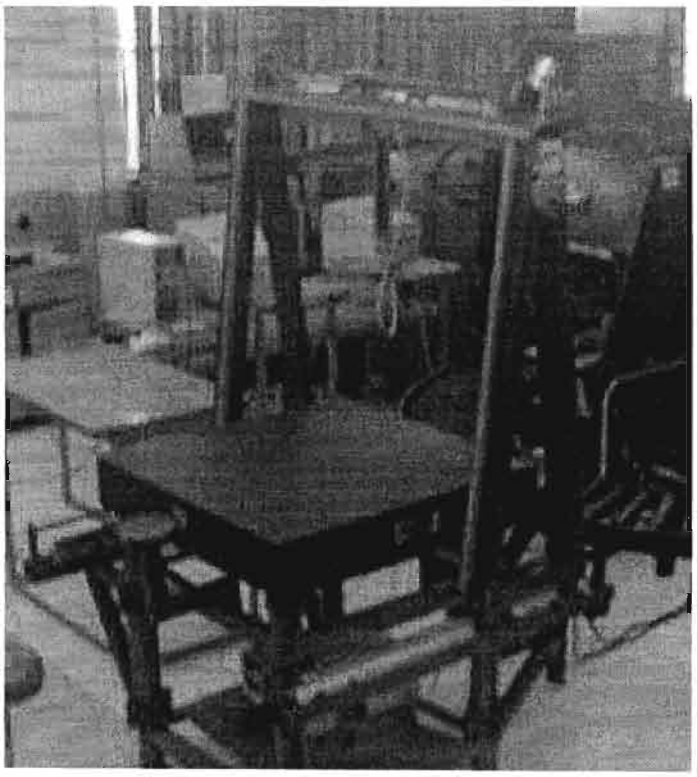


Figure 2.15. The contact-sensor CMM in the MR^{2}G laboratory.

- The Automated Storage and Retrieval System (AS/RS). The AS/RS was integrated as a multi functional storage and material handling system. The AS/RS delivered pelletised material directly to the conveyer system and the AGV. Figure 2.16 shows the AS/RS.

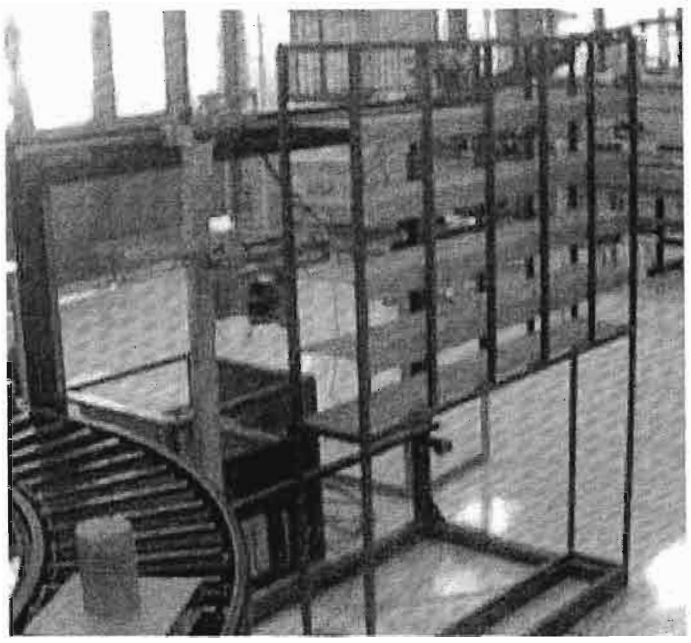


Figure 2.16. The Automated Storage and Retrieval System.

- The MAHO MH-400C CNC milling machine. The milling machine was retrofitted with a Heidenhain TNC-426 controller. The controller was connected to a PC, allowing part machining operations to be downloaded from the PC to the CNC controller. Figure 2.17 shows the MAHO milling machine.

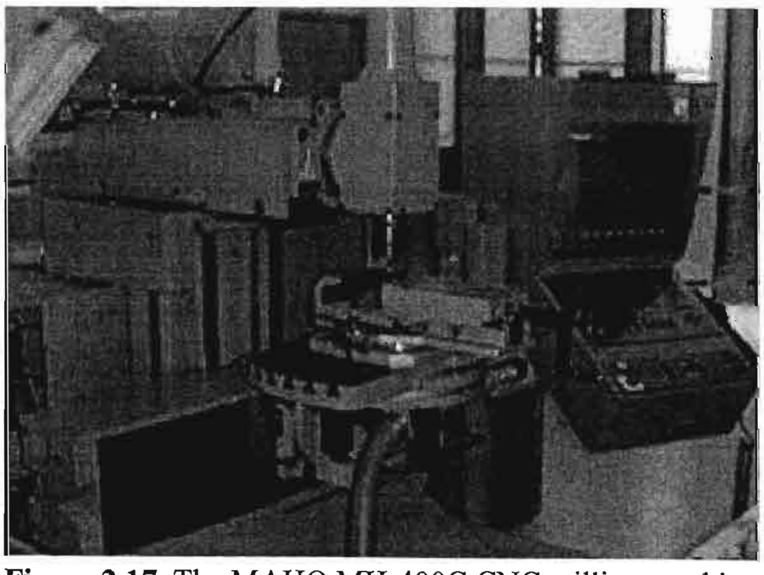


Figure 2.17. The MAHO MH-400C CNC milling machine with a Heidenhain TNC-426 controller.

All the CIM cell components, were designed and controlled using *modular mechatronic actuators* (ref. Chapter 3) and *modular mechatronic feedback devices* (ref. Chapter 4).

CHAPTER 3

3 MECHATRONIC ACTUATION

Most mechatronic systems involve motion or action of some sort. Actuators are devices that produce this motion, resulting in a required action. This motion can be applied to any articulated structure. The ability to affect some form of system response or action is fundamental in ensuring an effective interaction with the system and its environment [Harishima]. *Modular mechatronic actuation* is a field of engineering that is concerned with the design and implementation of modular actuating systems for specific system motions. The most common actuation technologies implemented in the field of modular mechatronic engineering are related to electrical machines, pneumatics and hydraulics.

The modular mechatronic CIM control system has been developed as a low-cost computer-based technology facilitating its integration into a PC-based flexible manufacturing environment. The power requirements of the CIM control system exceeded the power limitations of the standard PC-based controllers, and certain power electronic driver modules were designed.

This chapter discusses the fundamental concepts of the mechatronic actuators used in the design and implementation of the *power electronic drive modules* used to control the six-degree of freedom industrial robot and conveyer part manipulation and positioning system, implementing a modular design approach. The motors used in the industrial robot are 40V DC permanent magnet servomotors, and the motors used in the conveyer system are 12V DC permanent magnet servomotors.

3.1 DC MOTORS

This section summarises the works of [Auslander et.al.] and [Stadler] and reviews the fundamental concepts of rotating DC motors used in the industrial robot and conveyer system..

3.1.1 Fundamental Operating Model

The fundamental operating model of a DC motor can be derived from the "black-box" representation of the motor. In its most simplistic model representation, a DC motor has a pair of wires on one side and an output shaft on the other. For the exception of the electrical power through the wires and the motion of the output shaft, there is no other energy interaction with the motor. For this conceptual analysis, thermal effects, bearing friction and airflow are all neglected. Assuming that there are no mechanisms for energy storage or dissipation inside the "black-box" labelled motor (ref. Figure 3.1).

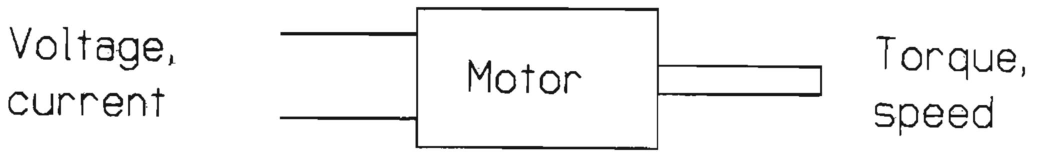


Figure 3.1. The simplified "black-box" representation of a DC motor.

$$\text{Mechanical power} = \text{Electrical power} \tag{3.1}$$

The power on each side of equation 3.1 is the product of the variables:

$$P = Vi = T\omega \tag{3.2}$$

and

$$T = K_T i \tag{3.3}$$

Substituting equation 3.3 into equation 3.2 allows the back emf equation to be determined. The back emf is the voltage produced by the motor as a result of its speed.

$$Vi = K_T i\omega \tag{3.4}$$

therefore

$$e_b = K_b \omega \tag{3.5}$$

The mathematical model can further be expanded to consider the "black-box" motor driving an inertial load (ref. Figure 3.2). The inertial load is a flywheel.

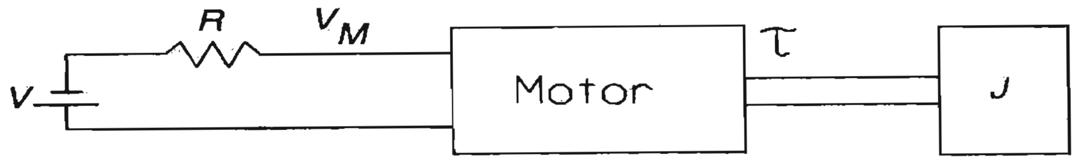


Figure 3.2. The "black-box" motor with drive and load.

The motor is driven by a voltage, with a series resistance representing the motor winding and external circuit resistance elements. The mechanical side of the "black-box" is governed by the rotary equivalent of Newton's law, relating to torque and acceleration:

$$T = J_L \frac{\partial \omega}{\partial t} \tag{3.6}$$

The current depends on the voltage drop across the resistor:

$$i = \frac{V - V_m}{R} \tag{3.7}$$

And the voltage across the motor is related to the speed from equation 3.5. The torque can now be expressed as a function of the shaft speed:

$$T = \frac{K_T(V - V_m)\omega}{R} \tag{3.8}$$

Substituting equation 3.8 into equation 3.6 results in the final equation of the motor model in terms of the shaft speed ω:

$$J_L \frac{\partial \omega}{\partial t} = \frac{K_T(V - V_m)\omega}{R} \tag{3.9}$$

Finally,

$$\frac{\partial \omega}{\partial t} = \frac{K_T}{J_L} \frac{(V - V_m)\omega}{R} \tag{3.10}$$

This final equation has an equilibrium point where the speed across the motor becomes zero. This phenomenon occurs when the back-emf is equal to the applied voltage. The significance of this result is that the motor will seek a constant operating speed regardless of the inertia of the load it is driving. The load inertia will only affect the transient behaviour of the motor. Figure 3.3 shows the rise of the motor velocity with time as a motor is started from rest. The two curves represent two separate values of inertia. Note how the response time varies while the equilibrium value stays constant for both loading conditions.

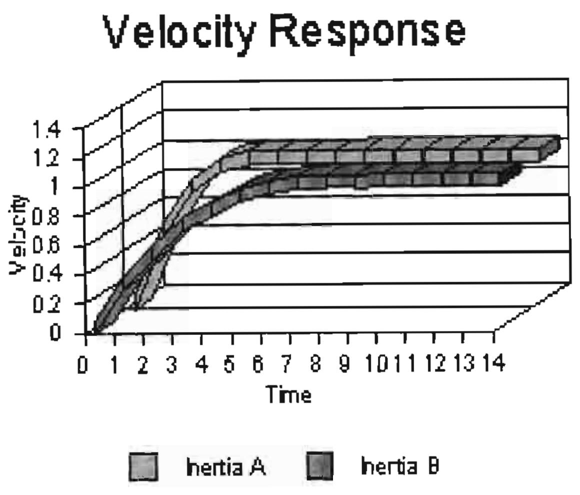


Figure 3.3. Velocity response of a DC motor under varying inertia conditions.

Impedance matching is critical when implementing a DC motor system, as the torque-speed characteristics of DC motors are fixed by their design parameters. Potential load variations outnumber standard motor availability, so motor to load or impedance matching is not always optimal. Power transmission devices such as gearboxes, pulleys or lead screws are used as impedance matching devices. These devices are the mechanical equivalents of electrical transformers. Figure 3.4 shows the general arrangement of an inertia transforming or impedance matching elements.

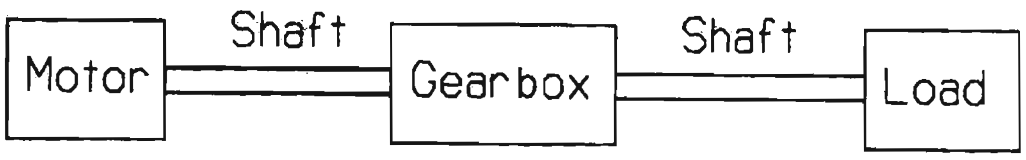


Figure 3.4. The “black-box” DC motor with a gearbox for impedance matching.

Referring to Figure 3.4, the inertia transforming element is a gearbox. The gearbox is used to match the needed static characteristics of the motor-load system. The speed and torque range of the system are the two most important static properties. If maximum power transfer to the load so as to achieve maximum acceleration is the most important operating characteristic, the classic impedance matching rule provides optimisation. For this assumes the load and motor are of pure inertia. Losses due to friction are neglected. Then the total sum of the system inertia on the motor side of the gearbox is the sum of the motor and reflected load inertia:

$$J_T = J_m + \frac{1}{r^2} J_L \tag{3.11}$$

Where r is the gear ratio. The kinetic energy of the load is given by:

$$KE_L = \frac{1}{2} J_L \omega_L^2 \tag{3.12}$$

when differentiated with respect to time, an expression for power is determined by:

$$P = \frac{\partial}{\partial t} KE = J_L \omega_L \frac{\partial \omega_L}{\partial t} \tag{3.13}$$

for a motor the equivalent power equation is:

$$P_L = J_m \omega_m \frac{\partial \omega_m}{\partial t} \tag{3.14}$$

The maximum power can be achieved only by delivering the maximum current to the motor. The acceleration is constant and proportional to the ratio of the (applied torque)/(total inertia). Since the acceleration is constant, the power to the load is:

$$P_L = \frac{J_L T_m^2 t}{r^2 J_T^2} \tag{3.15}$$

Optimise the value of P_L with respect to the gearbox ratio, r:

$$J_m = \frac{1}{r^2} J_L \tag{3.16}$$

3.2 SWITCHING DEVICES

Many actuators rely on electromagnetic forces to create the required action. When a conductor carrying current is moved in a magnetic field, a force is produced in a direction perpendicular to the current direction and magnetic field direction [Histand et.al.].

3.2.1 Electromechanical Relays

An electromechanical relay is a solenoid used to make or break mechanical contacts between electrical leads. A small voltage input to the solenoid controls a potentially large current through the relay contacts. Relays offer a simple on/off switching action in response to a control signal. Figure 3.5. illustrates the principle of operation. When a current flows through the coil of wire a magnetic field is produced. This pulls a movable arm that forces the contacts to open or close. Applications include power switches and electromechanical control elements.

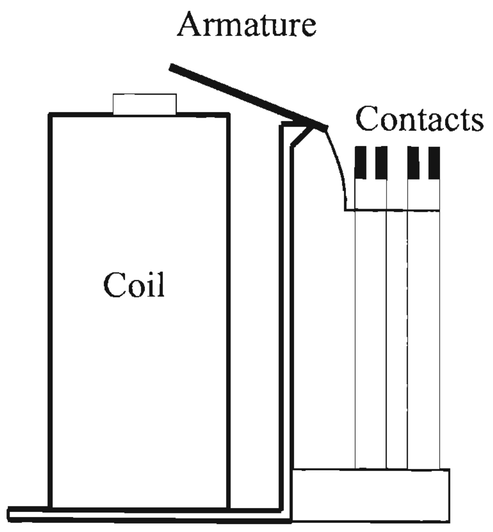


Figure 3.5. Schematic of the electromagnetic relay.

3.2.2 Solid-State Devices

There are a number of solid-state devices, which are used for switching the devices and circuits. The ones related to CIM control environment include:

1. Diodes
2. Bipolar transistors

3.2.2.1 Diodes

A diode allows a significant current to flow in only one direction. A diode is a directional element only passing a current when forward biassed (ref. Figure 3.6.).

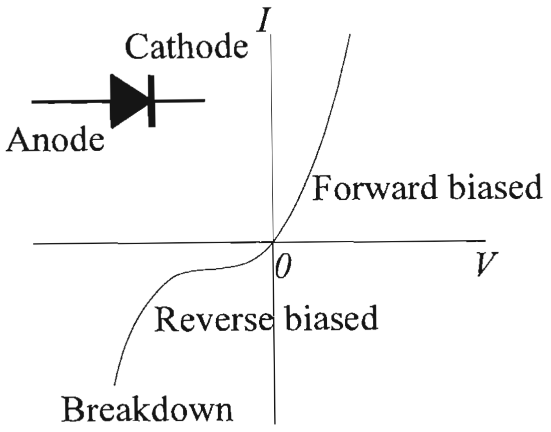


Figure 3.6. Diode characteristics, showing the forward- and reverse biassed conditions.

If the diode is reverse biassed, it will break down. If an alternating voltage is applied across a diode, it can be regarded as only switching on when the direction of current flow is from the anode to cathode ie. Forward biassed. The resultant current is half rectified. This is the principle of operation for a bridge rectifier.

3.2.2.2 Bipolar Transistors

Bipolar transistors come in two modes of operation. These include NPN and PNP. Figure 3.7. Shows the configuration of each type.

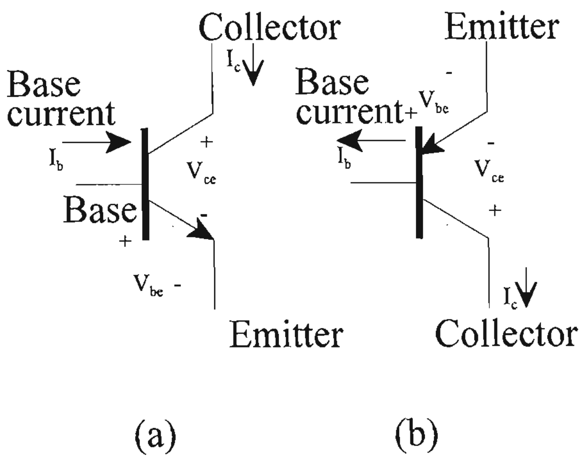


Figure 3.7. The two type of resistor symbols (a) NPN and (b) PNP.

Consider the NPN transistor, the main current flows in at the collector and flows out the emitter. The base is where the controlling signal is applied. The path from the base to the emitter (ref. Figure 3.7.) has an arrow, which is of the same form as a diode. When the transmitter is in normal operation there, will be a voltage drop of about 0.7V from the base to emitter. The collector current I_c is an exponential function of the base-emitter voltage V_{be}. PNP transistors function essentially like NPN transistors, with the main distinction being polarity.

3.3 CONVEYER SYSTEM POWER ELECTRONICS

The conveyer system is a PC-based technology that implements a discrete component control system based on the transistor-transistor logic TTL protocol. The power requirements of the PC-based conveyer system exceeded the power capabilities of the desktop computer, necessitating the design and implementation of certain power electronics. The design characteristics of the power electronics were:

1. TTL input signals to produce the appropriate and corresponding high power outputs.

2. The power electronics must be modular in design, to interchange conveyer sections, enabling a flexible manufacturing environment.

The conveyer system implemented ten permanent magnet DC servomotors (ref. Chapter 2). The design approach to the power electronics included a high degree of modularity. The power electronic drive circuits were not designed as application specific circuits, but rather as robust power electronic modules that were flexible in terms of application and operation.

3.3.1 DC Servomotor Drive Circuitry

The conveyer system implemented ten permanent magnet DC servomotors to power the pallet transport system. For the pallet transport conveyer system, the oval conveyer configuration had to be flexible and interchangeable. The conveyer system consisted of six sections, which included:

1. Two straight sections, each driven by one 12V DC servomotor.
2. Four 90°curved sections, each section driven by two 12V DC servomotors.

For each pallet transport conveyer section the DC servomotors were required to drive inertial and frictional loads in both directions. The motor drive circuitry had to therefore incorporate a logic decoding module that was capable of accepting TTL input signals containing direction information. The basic configuration of the modular design was to incorporate the power electronics for each section of the pallet conveyer system onto one drive module. Each modular was designed to drive two DC servomotors, capable of direction switching. Based on the control information, the drive circuitry had to incorporate power electronics to enable an external power source to provide the high-power drive signals required to drive the DC servomotors (ref. Figure 3.8).

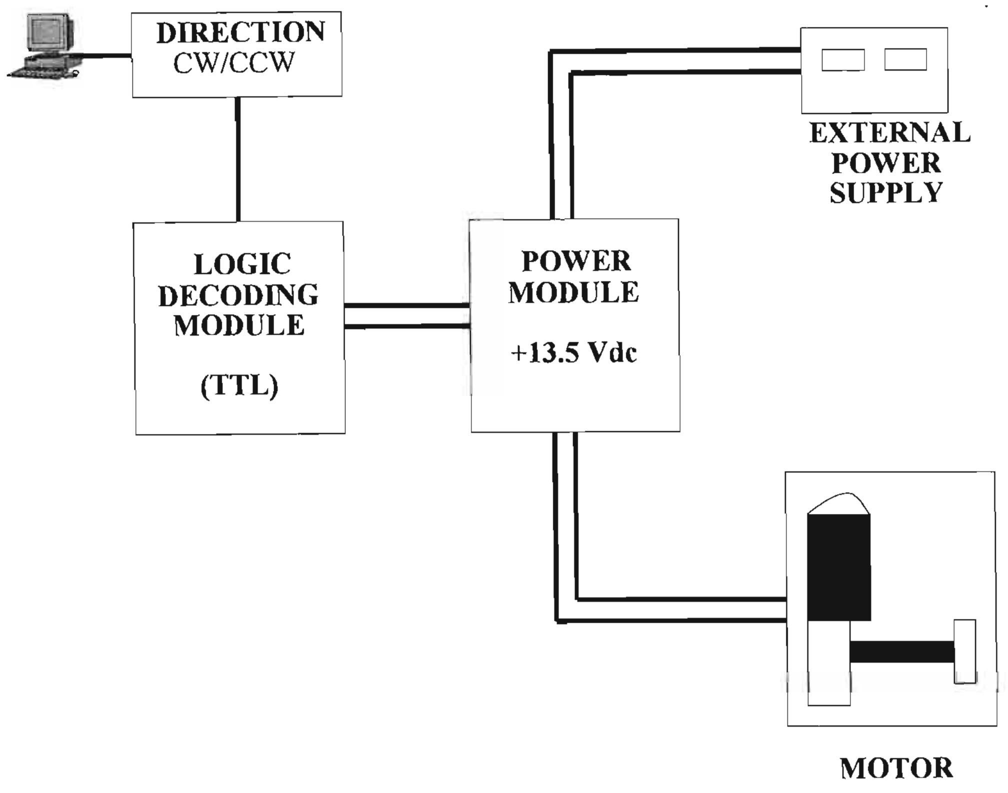


Figure 3.8. A schematic representation of the components of the DC motor drive circuitry. Note the different signal levels.

The *relay power module* drive circuitry used a transistor - relay circuit that allowed a low power TTL control input signal to activate relays to drive the DC servomotors from the external power supply. The relay solenoid was connected to a collector-emitter circuit of a low cost, low power bipolar NPN-transistor. The TTL signals were applied to the base of the transistor, setting the voltage bias of the base. When the PC-based control signal was a digital high $2.8\text{Vdc} < V_b < 5\text{Vdc}$ the base was biassed such that the current flowed in the collector-emitter circuit, energising the relay coil. Similarly if the control signal was a logic low $2.1\text{Vdc} > V_b > 0.8\text{Vdc}$ the biassing of the base would not allow current to flow through the collector-emitter circuit, not allowing the coil to be energised. The switching state of the relay was controlled by setting the appropriate control signal TTL levels to either high or low. The selection of the appropriate relay for the power modules of the drive circuitry depended on the peak voltage and current levels of the DC servomotors driving the conveyer system. The peak demand values were based on physical mechanical measurements. The selection of an appropriate relay was then based on a maximum supply voltage of 15V dc with current rating 10A.

The power circuitry utilised double-pole double-throw relays that required a 12V dc excitation voltage to energise the coil and switch the relay contacts. As the coil of the relay could not be energised using a TTL signal voltage, a low current 12V dc power supply was included in the circuit design. The drive circuitry consisting of three transistor-relay modules that controlled the direction of the two DC servomotors connected to each circuit. The direction module (ref. Figure 3.9) reverses the polarity of the excitation voltage supplied to the two fixed speed motor modules (ref. Figure 3.10).

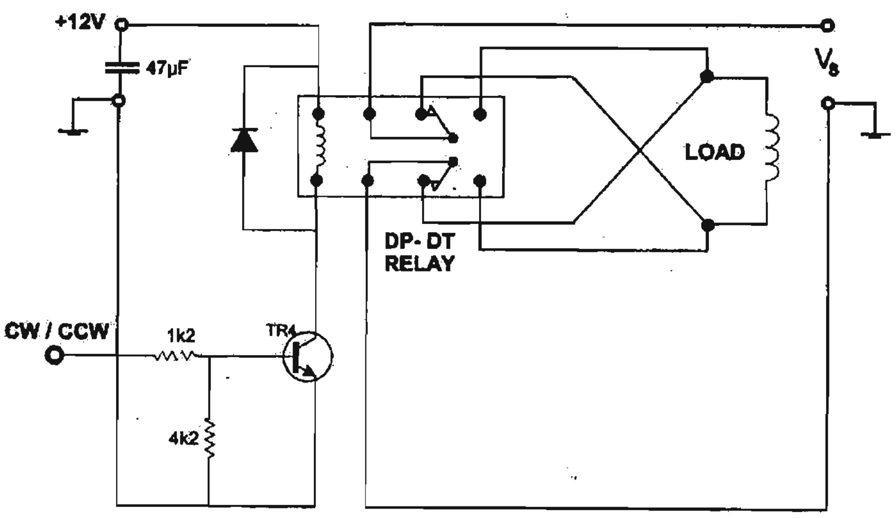


Figure 3.9. The circuit diagram for the direction control module of the DC motor drive circuitry. The control voltage is applied to the base of the transistor TR4.

The design of both the direction module and the speed modules ensured that the high-power drive section was isolated from the control section of the modules. The transistor circuits were isolated from the power circuit by the contacts of the relays. Noise reduction in the logic circuits included a freewheel diode connected across the relay coil thereby providing a switching current path.

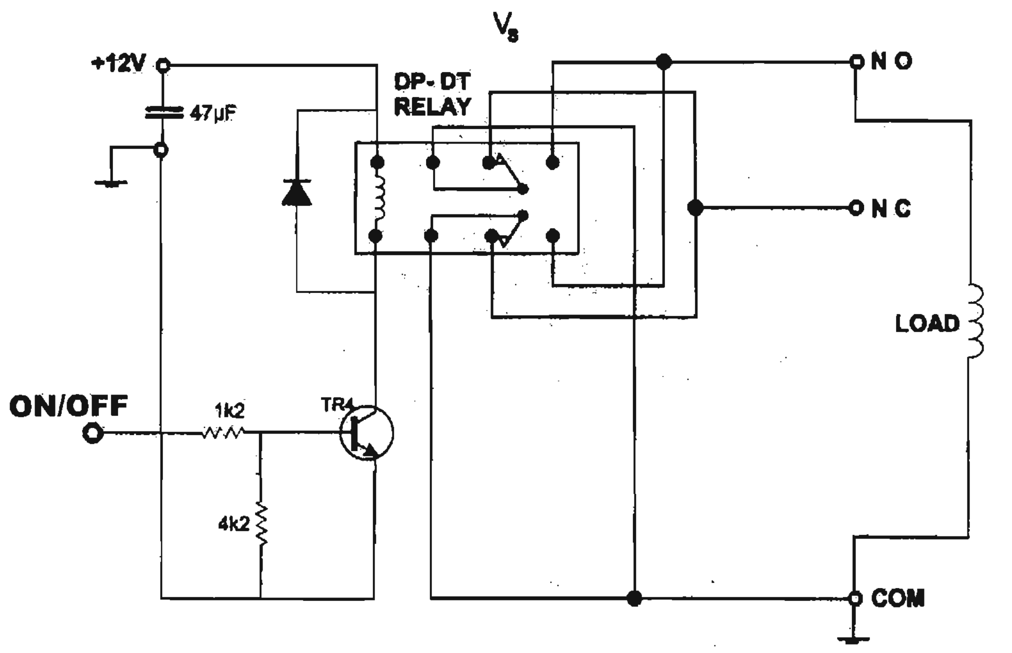


Figure 3.10. The circuit diagram for the on/off control module of the DC motor drive circuitry. The control signal is applied to the base of the transistor TR4.

The transistor circuit required a low current 12V dc supply that powered the collector-emitter circuit. A decoupling capacitor was connected across the supply in order to reduce any noise picked up lower current logic circuits. The direction and two power control relay modules were mounted on the same printed circuit board (PCB). Each board was designed as a module, to provide the power control signals for each section of the conveyer system. Figure 3.11 illustrates the connections of the modular mechatronic transistor-relay driver circuit.

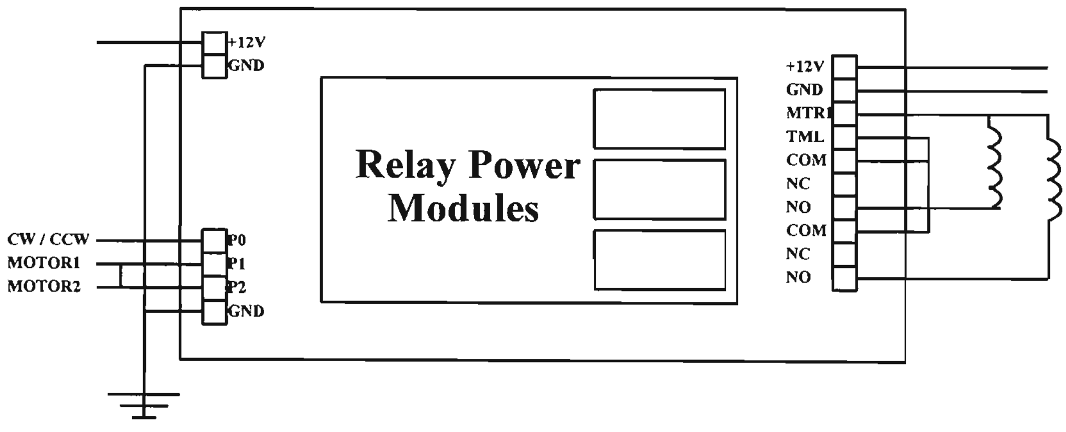


Figure 3.11. The driver board that incorporated all three power modules. The coils indicate the two separate motors.

3.4 INDUSTRIAL ROBOT POWER ELECTRONICS

There was a need for a high-power low-cost driver circuit allowing the PC peripherals to interface and control the robot. The control signals supplied to the *LM-12 power module* robot driver circuits were -10V to +10V analogue signals. These signals were generated by an Advantech© PLC-832 servo control card, and control voltages were generated according to a P control algorithm (ref. Chapter 7). The control signal voltages were low-current analogue signals, unable to drive and control the 40V dc Servomotors of the PUMA industrial robot. Modular driver circuitry had to be designed. The peak operating voltages of each of the six DC servomotors were of the magnitude 40V dc at a peak dissipation current of 15A each.

A dual package LM12 150W operational amplifier was used in the modular mechatronic robot driver circuit. The LM12 has typical uses as a high power audio amplifier component. The LM12 consisted of an internal H-bridge configuration.

3.4.1 High Power Robot Drive Circuitry

The LM12 was a power operational amplifier capable of driving ±35V at ±10A, while operating from ±40V dc supplies. The monolithic integrated circuit delivered 150W of sine wave power into a 4Ω load with 0.01% distortion [National Semiconductor]. The power bandwidth was 60 kHz, and a peak dissipation capability of 800W allowed the driver circuit to handle reactive loads without derating. The operational amplifier included the following features:

- Input protection.
- Controlled turn on.
- Thermally limiting.
- Over voltage shutdowns.
- Output current limiting.
- Dynamic safe-area protection.

The IC delivered a peak output current of 10A at any output voltage and was protected against overloads, including shorts to the supply. The LM12 operational amplifier delivered considerably high power outputs, without resorting to complex switching schemes. The LM12 operational amplifiers could be parallelled or bridged for greater output capabilities.

Figure 3.12. Shows the modular LM12 driver circuit. The input reference signal to the LM12 was provided by the digital-to-analogue converters of the Advantech© PCL-832 servomotor control card. The modular LM12 driver circuit is biassed for direction and speed control. The LM12 driver circuit has a peak current rating of 30A, as three LM12 operational amplifiers are connected in parallel. During operation the parallelled LM12 operational amplifier's outputs were kept in phase by the series connected 0.47Ω resistors while receiving the same analogue control signal. Using the LM12 as a driver module allowed for the simple "building block" type of approach to be used to solve over-current problems. The driver circuit could be modified for the control of smaller motors, by removing the extra LM12 operational amplifiers connected in parallel. The operating voltage of the modular LM12 driver circuit was form ±12V dc to ±40V dc.

Each set of LM12 power operational amplifiers was mounted on a heat sink to dissipate the excessive heat generated during normal operating conditions. The maximum DC thermal resistance of the LM12 operational amplifier was 2.3 °C/Watt. The peak operating case temperature was kept below 70 °C.

The power electronic drive circuits for the conveyer and robot systems were not designed as application specific circuits, but rather as robust power electronic modules that were flexible in terms of application and operation. This was achieved, as the modules were used to control a variety of DC motor systems in the MR^{2}G laboratory.

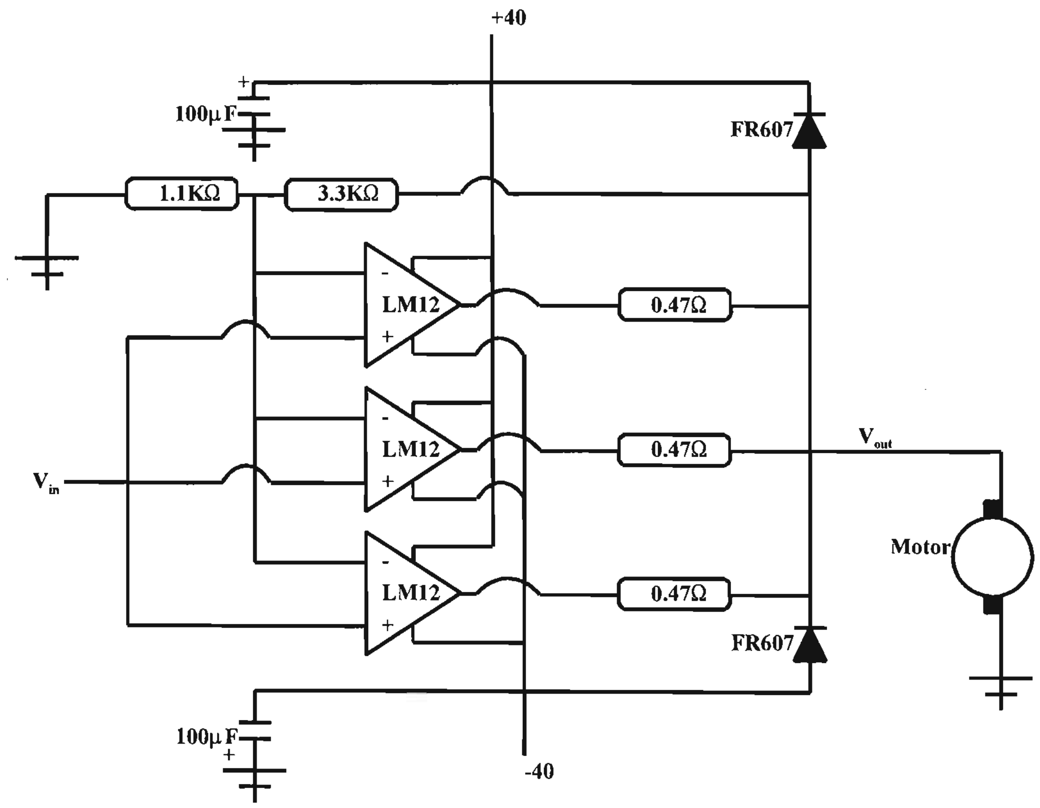


Figure 3.12. The modular LM12 driver circuit.

CHAPTER 4

4 MECHATRONIC-BASED FEEDBACK INFORMATION

Mechatronics bring together the areas of technology involving sensors and measurement systems, drive and actuation systems, analysis of the behaviour of systems, control systems and microprocessor systems [Bolton]. One way to measure the sophistication of a CIM cell is in terms of its independent interaction with its environment, more specifically in terms of its capability to sense the environment and its ability through programming to interpret it. Measurement devices are required for accurate closed loop feedback control. Figure 4.1 illustrates the system requirements for the accurate control of modular mechatronic CIM cell.

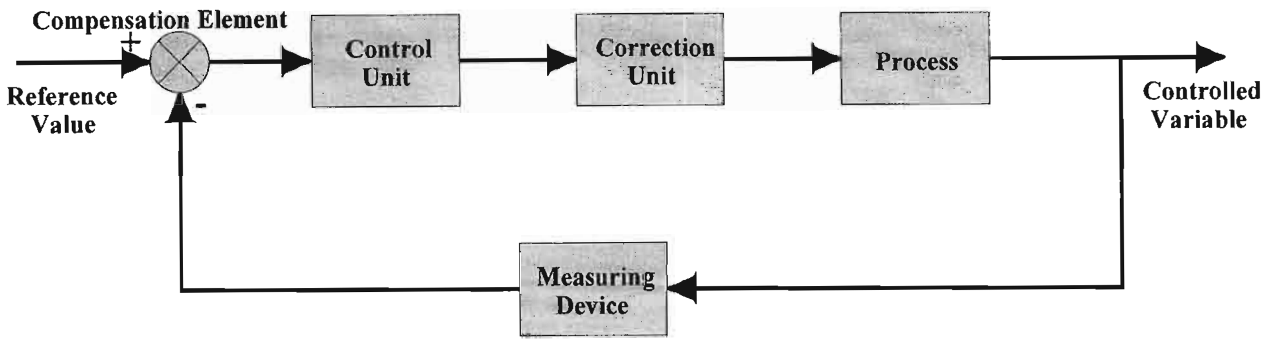


Figure 4.1. Elements of a closed loop control system for the modular mechatronic CIM cell.

Measuring devices are classified as static if the data varies slowly with time or not at all, and dynamic if the measured variables vary rapidly. In addition there is a distinction between analogue and digital instrumentation. Measuring devices never measure what they set out to measure, but rather the combined system incorpOrating the measuring device. In the derivation of the mathematical models of the combined system, the separate mathematical models of the measuring devices are needed. These models are usually linear and of the first- or second-order [Stadler].

The output signals from the mechatronic-based feedback information devices must be processed. These signals may be too small, contain noise or require linearisation. The signals may be analogue and have to be converted to digital signals. All these changes are referred to as signal conditioning.

This chapter discusses the mechatronic systems implemented to collect position feedback information used to control the PC-based industrial robot and conveyer part manipulation and positioning system. In the field of mechatronic-based feedback information there are a large variety of devices used. This discussion will detail the specific devices used in the PC-based CIM control system to collect real time feedback information regarding the status of the individual motion systems used and the development of the *mechatronic feedback modules*.

4.1 THE PC-BASED CONVEYER SYSTEM

This conveyer system transported the workpiece from the individual machining and part delivery systems of the CIM cell. The control of the conveyer system required the implementation of a feedback system that was capable of accurately monitoring the position of the workpiece along the conveyer track.

The pallet must be tracked and delivered to the correct part processing stations. The conveyer system consisted of roller conveyers and formed part of the modular flexible manufacturing system into which the PC-based robot was integrated.

The control resolution required the pallet to be accurately positioned and transported. An byte array feedback technique was developed to obtain the feedback information from the part conveyer system. The continuous scanning of the byte array information from the conveyer system was compared to a database enabling real-time positional information of a pallet within the conveyer system. Fourteen critical sensing points were implemented to monitor the pallet position and were interfaced directly to two 8-bit ports of a computer interface card. The value of the byte read from the interface ports was cross-referenced against a list of possible values in order to determine the position of the pallet along the conveyer system.

The critical sensor points were positioned to monitor the pallet leaving and entering a conveyer zone. As there were six modular conveyer sections, one sensor was placed at each conveyer end. Extra sensors were placed at the Automated Guided Vehicle (AGV) docking station and PC-based robot material handling station. The critical sensing points were implemented using long range infrared (IR) switches. Each of the *IR switch modules* comprised of a transmitter and a receiver unit. As the leading edge of a pallet would arrive at a sensor point, the IR beam would be broken. Once the trailing edge of some pallet passes the sensor point, the receiver is once again irradiated and the beam is restored. Figure 4.2 shows the long range infrared transmitter circuit.

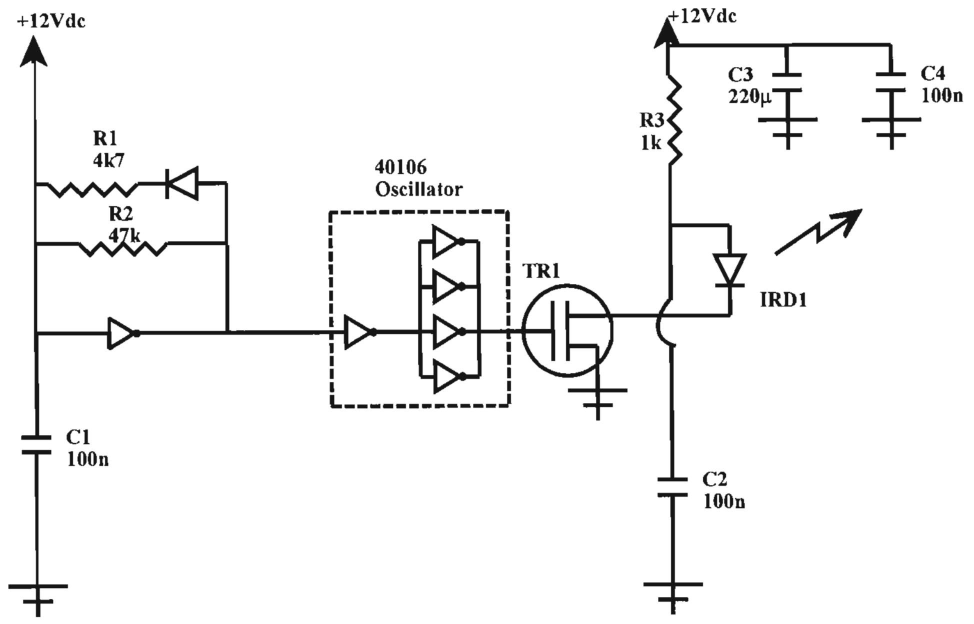


Figure 4.2. The long range infra-red transmitter circuit.

The circuit uses a 40416-oscillator circuit to generate a rectangular waveform with a short duty cycle; a greater percentage of the period is spent in the low state as opposed to the high state. The frequency of the pulse train set by resistors R1 and R2 must be significantly higher than the scanning frequency at which the states of the individual sensor points are monitored. The modulated signal is used to bias the base-emitter circuit of the field effect transistor, TR1.

The infrared emitter, IRD1, is connected in the collector-emitter (drain-source) circuit of TR1 with the capacitor C2 connected across it. The capacitor is charged by the supply when the transistor is not biassed. When current flows in the collector-emitter circuit the capacitor discharges through the diode resulting in a high current spike that dramatically increases the transmitting power of the diode. Furthermore the short duration of the pulses ensures that the diode will not be burnt out. The capacitor group, C3 and C4, decouple the supply from the circuit in order to smooth out the high amplitude switching transients of the supply voltage, thereby eliminating noise and upholding the integrity of the transmitted pulse.

The receiver circuit is based on a 12 Vdc photo-transistor, TR1 (ref Figure 4.2). The photo-transistor is biassed by the intensity of infrared radiation detected at its base-emitter junction. The photo-transistor receives a high frequency radiation pulse which results in the switching of the collector-emitter current thereby necessitating the use of AC coupling capacitors, C3, C4 and C5 (ref Figure 4.3). Since the current in the collector-emitter circuit is dependent on the intensity of the detected radiation, the output from TR1 is coupled via capacitor C3 to the base of transistor TR3 that forms the second half of the Darlington pair.

The output from transistor TR3 is coupled to a modulating circuit, R4 and R5, that allows a small additional current to flow from the supply through R5. Therefore the pulse train generated by TR1 and TR2 is modulated and as a result does not display the current fluctuations of the photo-transistor collector-emitter circuit. The stabilised pulse signal is applied to the base of transistor TR4. The TR4 collector-emitter circuit performs a simple level shifting function that reduces the supply voltage to TTL level using the potential divider resistance pair, R6 and R7. The output 0/P is pulled high when TR4 is not biassed, ie the photo-transistor is not being irradiated by resistor R6. The polarised capacitor C6 decouples the output signal thereby reducing the ground noise interference in the output signal.

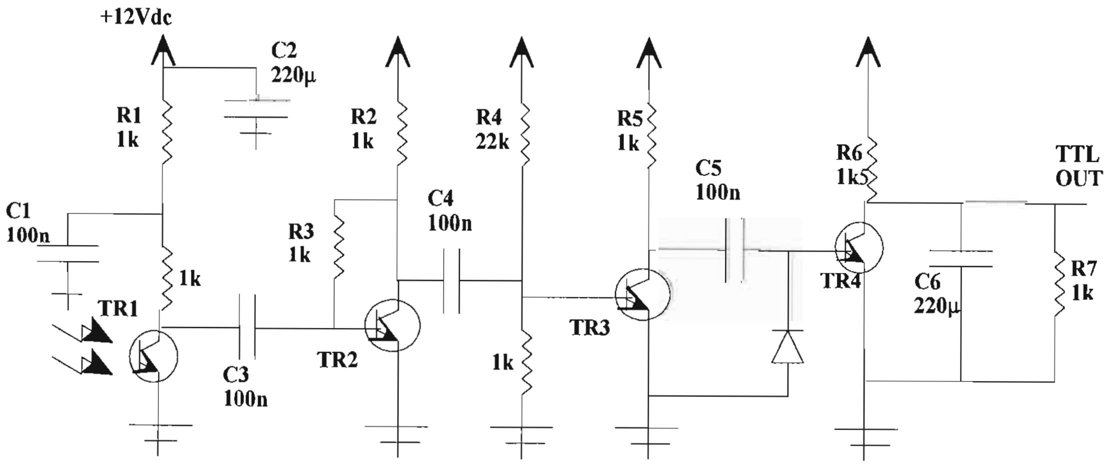


Figure 4.3. The infra-red receiver circuit. The circuit produces a clean low-to-high transition whenever the input infra-red beam is blocked.

4.2 INDUSTRIAL ROBOT FEEDBACK DEVICES

Each joint of the PC-based robot is monitored by an incremental encoder and resistive potentiometer. The resistive potentiometer is a contacting sensor, returning absolute joint position feedback at robot startup. The incremental encoders provide the relative joint position from point of startup.

4.2.1 Resistive Potentiometers

The robot sliding contact resistive sensors are used to express the robot's angular displacement in terms of voltage. The potentiometers of each joint are directly coupled to the DC servomotors in a straight line configuration. The calibration of each joint position is not influenced by this direct coupling, but is dependant on the gear train from the DC servomotors to robot revolute joints. The best resolution is obtained when the potentiometer simply consists of a wire and sliding contact connected to the displaced device. For a resistive winding the resolution is equal to the reciprocal of the number of turns. Figure 4.4 shows a displacement resistor, where:

$$v_0 = \frac{v_R a\theta}{L}\left[1 + \frac{R}{R_0}\left(\frac{a\theta}{L}\right)\left(1 - \frac{a\theta}{L}\right)\right]^{-1}$$

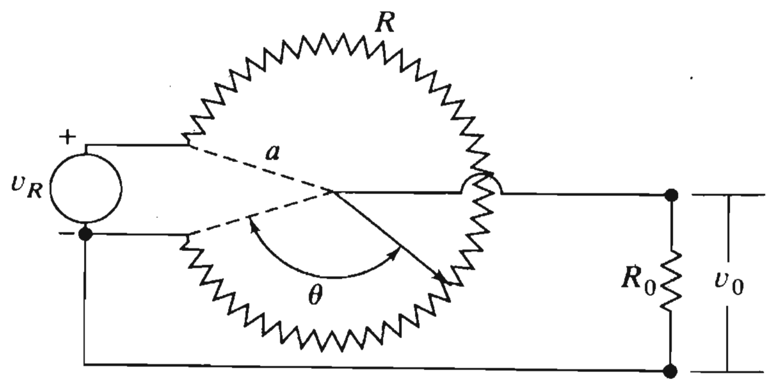


Figure 4.4. Illustrates the single wire resistive wire of a rotary potentiometer [Stadler].

Potentiometers produce analogue signals which contain certain elements of noise. Filtering techniques are used, this reduces processing data acquisition speeds and reduces potentiometers to static devices. The best results from potentiometers are obtained at the start of an operating sequence. Potentiometers act as calibrating devices for absolute position, from which the analogue noise is filtered. The filters are either electronic hardwired filters or software generated filters. With increased computer processing speeds and software reliability, software filtering has become an everyday occurrence. Simple software algorithms allow for adaptive noise filtering to occur as system changes occur.

4.2.2 Optical Incremental Encoders

An optical encoder is a device that converts motion into a sequence of digital pulses. By counting a single bit or by decoding a set of bits, the pulses can be converted to relative or absolute position measurements. The most common type of encoder is a rotary encoder. Rotary encoders are manufactured in two basic forms:

1. Absolute encoders, where an unique digital word corresponds to each rotational position of the shaft.
2. Incremental encoders, which produce digital pulses as the shaft rotates allowing measurement of the relative shaft position.

As shown in Figure 4.5 most rotary encoders are composed of a glass or plastic code disk, with a photographically deposited radial pattern organised in tracks.

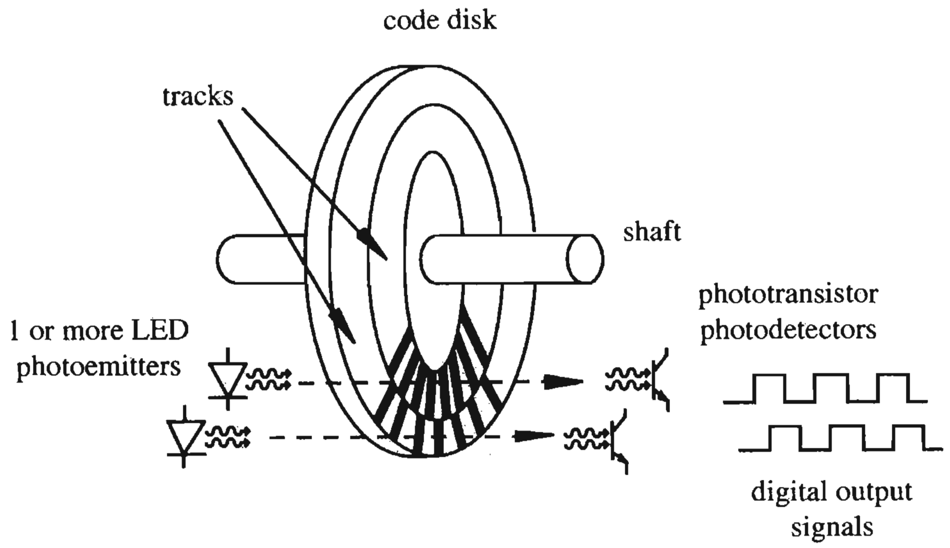


Figure 4.5. Illustrates the components of a rotary optical incremental encoder [Histand et.al.].

Digital pulses are produced as the lines on each track interrupt the beam between the photo emitter-detector pair. The photo emitter pair can be oriented in two configurations:

1. As in Figure 4.5, the beam shines through the path of the rotating disk or,
2. The beam is reflected of the surface of the rotating disk, and this last type of rotary incremental encoder is used as the incremental feedback device of the PUMA robot.

The incremental encoder, sometimes called the relative encoder, is simpler in design than the absolute encoder. It consists of two sensors whose outputs are called channels A and B. As the shaft rotates, pulse trains occur on these channels at a frequency proportional to the shaft speed, and the phase relationship between the signals yields the direction of rotation. The code disk pattern and output signals A and B is illustrated in Figure 4.6. The A and B channels are used to determine direction of rotation, the signals from the two channels are ¼ cycle out of phase with each other and are known as quadrature signals. A third output channel called INDEX, yields one pulse per revolution.

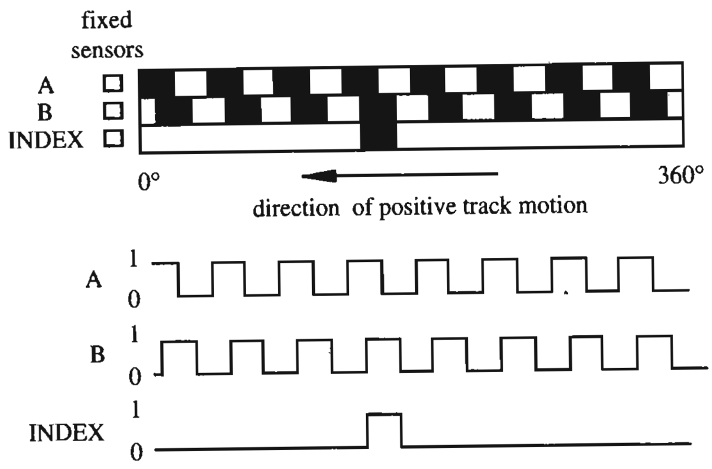


Figure 4.6. Illustrates the encoder disk track patterns [Histand et.al.].

The quadrature encoder signals A and B can be coded to yield the direction of rotation shown in Figure 4.7.

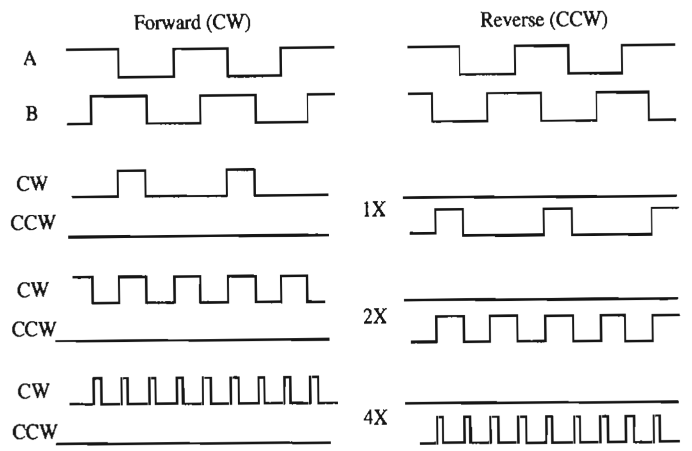


Figure 4.7. Illustrates the quadrature direction sensing and resolution enhancement [Histand et.al.].

Decoding transitions of A and B by using sequential logic circuits in different resolutions of the output pulses: 1X, 2X and 4X. 1X resolution only provides a single pulse for each cycle in one of the signals A or B. 4X resolution provides a pulse at every edge transition in the two signals A and B providing four times the 1X resolution. The direction of rotation (CW or CCW) is determined by the level of one signal during an edge transition of the second signal. Figure 4.8 shows the circuit originally designed to determine the direction of shaft rotation of the PC-based robot DC servomotors.

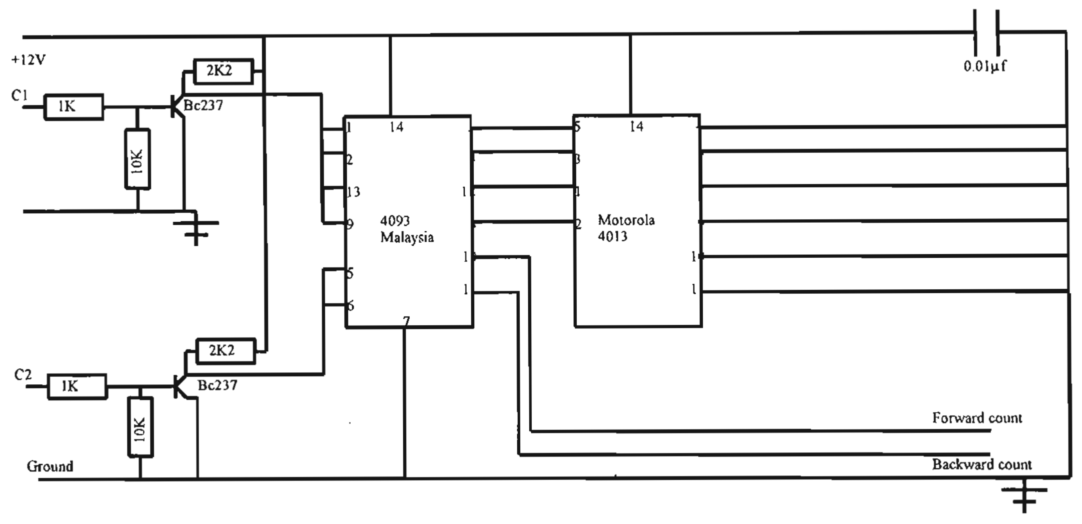


Figure 4.8. The quadrature encoder circuit.

The PCL-836 servo control card however has this built in capability, so the quadrature encoder circuit was discarded. The quadrature encoder circuit however can be used for any modular mechatronic encoder application.

4.3 SIGNAL CONDITIONING MODULES

Signal conditioning modules were developed to take the signals from the robot sensors and actuator circuitry and make them suitable for display, and to exercise control of the system. The robot input and output devices were interfaced to a microprocessor system through ports. The interface contained signal conditioning and protection, to prevent damage to the microprocessor system. The following are some of the processes that occur in conditioning a signal:

- Protection to prevent damage to any system element.
- Conditioning and changing a signal to suite the data acquisition equipment requirements.
- Eliminating and reducing noise. Increasing the signal to noise ratio (SNR).

- Signal manipulation to eliminate linearity problems.

4.3.1 LM12 - Buffer Circuit Module

The basis of many signal conditioning modules is the operational amplifier. The operational amplifier is a high gain DC amplifier. The LM741 is an example of a typical operational amplifier. The LM741 operational amplifier has two inputs, known as the inverting (-) and the non-inverting (+) input, the output of the LM741 is dependent upon the inputs made to these connections.

A particular form of this amplifier is when the feedback loop is a short circuit. Then the voltage gain is 1. Such an amplifier circuit is referred to as a voltage follower. The voltage follower circuit prevents the LM12 drive module (ref. Chapter 3) from loading the output from the PC servo control cards, acting as a buffer between the drive modules and the PC. The LM12- buffer circuit module is a low-cost protection solution.

4.3.2 Encoder - Signal Conditioning Module

Figure 4.9 shows the LM741 operational amplifier connected as a non-inverting amplifier. The output can be considered to be taken from across a potential divider circuit consisting of R_1 in series with R_2. The voltage V_a is then the fraction of the supply voltage V_{cc}. This relationship translates to the following expression:

$$\begin{aligned} V_a &= \frac{R_1}{R_1 + R_2} V_{cc} \\ V_a &= \frac{10}{10+10} 5 = \underline{2.5V} \end{aligned} \tag{4.5}$$

The supply latches as the input to pin 3 reaches V_a, producing a square waveform as input to the PC.

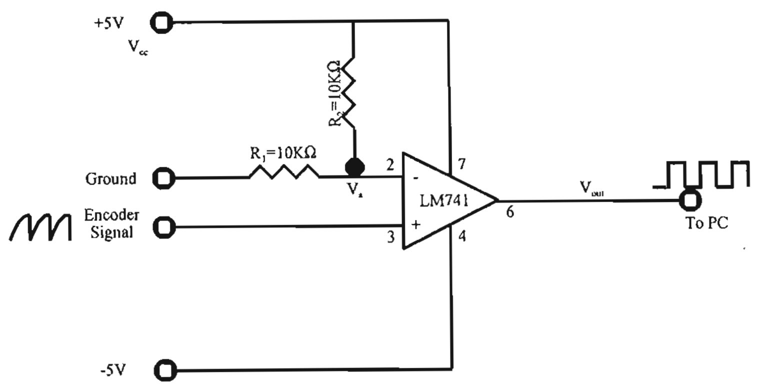


Figure 4.9. The bidirectional encoder conditioning module circuit.

The circuit converts the sawtooth waveforms of the PUMA robot bidirectional encoders to square waveforms. The PCL 832 Servo control cards require square waveforms to enable encoder pulse counting (ref. Chapter 7).

This chapter discussed the mechatronic systems implemented to collect position feedback information used to control the PC-based industrial robot and conveyer part manipulation and positioning system. This discussion detailed the specific devices used in the PC-based CIM control system to collect real time feedback information regarding the status of the individual motion systems used and the development of the *mechatronic feedback modules*. The feedback systems used were a low-cost solution to some feedback information gathering techniques used in the field of Mechatronics.

CHAPTER 5

5 THE CIM TELEOPERATION SYSTEM

Today's Internet technology provides for the development of integrated network environments for the diversified applications of different manufacturing systems. To be successful in real-world applications, Internet controlled machines require a high degree of autonomy and local intelligence to deal with the restricted bandwidth and arbitrary transmission delays of the Internet [Hu et.al.].

With the rapid development of the Internet, more intelligent devices and systems have been developed. Although the notion of Internet or Web-based manufacturing is relatively new and still in its infancy, it has captured the interest of many researchers worldwide. The research presented in this section has a completely new range of real-world applications, not only in tele-manufacturing, but also in tele-training, tele-surgery, disaster rescue and health care. Although the Internet provides a cheap and readily available communication channel for teleoperation, there are still many problems that need to be solved before successful real-world applications can be achieved. One way to overcome these problems are to remove the closed-loop control of the human operator and provide a high degree of machine intelligence for the uncertainties in real-world applications. As researchers it is essential to find the correct balance between human and machine interaction. Also an intuitive user interface is required for inexperienced people that control machinery over the Internet.

This chapter discusses the Internet concepts used in the framework for the control of the *modular mechatronic CIM teleoperation system*. The Internet control framework provided the macro control and monitoring of the PC-based PUMA robot, conveyer system, and visual feedback system for the CIM cell developed in the MR^{2}G laboratory at the University of Natal, Durban. Essentially this is the macro Internet control framework of the *modular mechatronic actuators* and *mechatronic feedback modules* described in Chapter 3 and Chapter 4.

5.1 BACKGROUND OF THE WORLD WIDE WEB

The Web was created early 1989 by researchers from the European Laboratory of Particle Physics (CERN) in Geneva, Switzerland. The goal was to create an online system that would allow nontechnical users to share data without the need to use arcane commands and esoteric interfaces. Within two or three years, users outside CERN were putting together pages for the Web while developers were designing and creating powerful browsers. By 1993, the Web and its browsers had become the way to move around the Internet.

Today an industry group known as The World Wide Web Consortium (W3C) develops common standards for the Web. The Web is defined in the following way. "It has a body of software, and a set of protocols and conventions. The WWW uses Hypertext and multimedia techniques to make the Web easy for anyone to roam, browse, and contribute to."[McGee].

5.1.1 HTML: The Language of the Web

The language used to program and control a Web page is known as: Hypertext Markup Language (HTML). HTML Web pages have the ability to include active links to other HTML pages either at the same or another Web site. These links are embedded into the page and rendered by the browser so that they are easily recognisable as jumps. When a user clicks one of these links, the browser loads, reads, and renders the destination page, which may be a remote Web page, one local to the Server, or even content on a different part of a displayed page itself. Figure 5.1 illustrates this concept. The underlying software uses the information contained in the link to connect to that site, and then retrieve the page for the browser.

When the link is to a page residing on another Server, a uniform resource locator (URL) in the anchor tag along with the Hypertext Transfer Protocol (HTTP) must be specified. The link information is in the URL, which is divided into several different parts beginning with the protocol. For HTML pages, the protocol is usually the HTTP. An exception to this is the File protocol, which is valid within the local directory or the local network, but not across the Web.

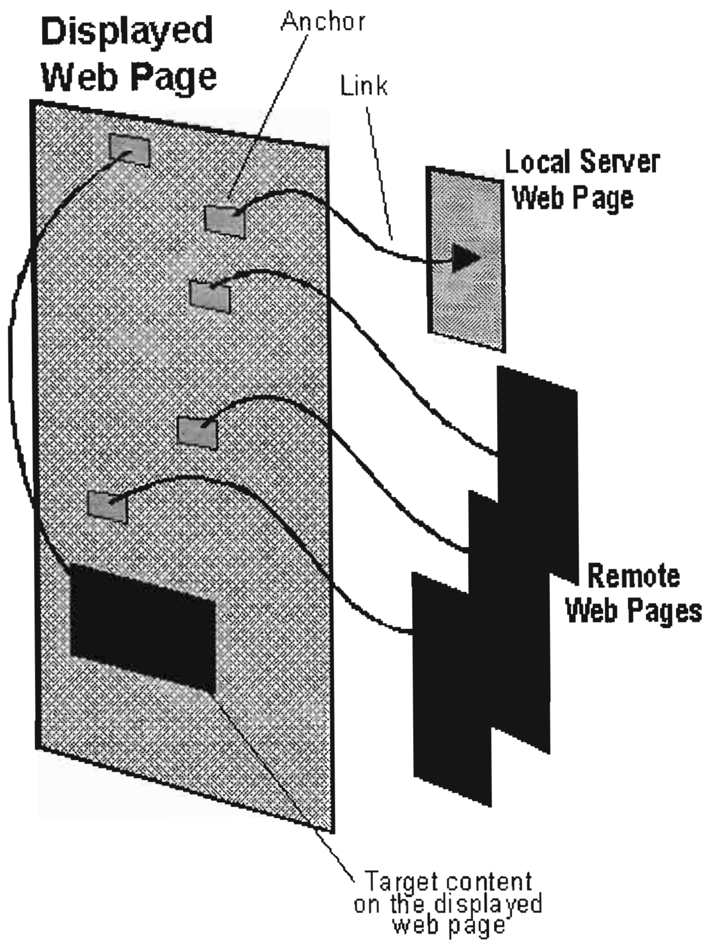


Figure 5.1. Links may be to remote Web pages or within the same page [McGee].

Several new and emerging technologies, however, create a tremendous opportunity to create active, and therefore interesting, Web pages. These technologies allow the creation and insertion of active content and objects into the Web page. Such content can be used to integrate multimedia (animation, video, sound, and 3-D rendering), dedicated applications, and database tools right into the Web page itself.

5.1.2 Visual Basic Scripting

Scripting with Microsoft Visual Basic® is available through the Microsoft Visual Basic Scripting Edition (VBScript). VBScript is a subset of the Microsoft Visual Basic programming language and is upwardly compatible with Visual Basic. VBScript is used to create active online content on a Web page by embedding the script into the HTML page. Similarly, VBScript also allows developers to link and automate a wide variety of objects in Web pages, including ActiveX controls and "applets" (created using the Java™ language from Sun Microsystems, Inc.), which are then loaded and registered in the user's system.

With VBScript, there are many ways to enhance the HTML page with interesting elements. For example, the HTML page with embedded VBScript can respond to user-initiated events quite easily. Functionality now provided by VBScript was previously possible only with a common gateway interface (CGI) application running on the Server. VBScript resides in the Web page, and doesn't demand Server processing time, or require the overhead of a communications link.

5.1.3 JavaScript

JavaScript is a scripting language developed by Netscape Communications and Sun Microsystems, Inc. Compared to Java, JavaScript is limited in performance because it is not compiled before execution. Basic online applications and functions can be added to Web pages with JavaScript, but the number and complexity of available application programming interface functions are fewer than those available with Java. JavaScript code, which is included in a Web page along with the HTML code, is generally considered easier to write than Java. A JavaScript-compliant Web browser, such as Microsoft Internet Explorer or Netscape Navigator, is required to interpret JavaScript code.

5.1.4 ActiveX Controls

ActiveX is a set of technologies that enables software components to interact with one another in a networked environment, regardless of the language in which the components were created. ActiveX is used primarily to develop interactive content for the World Wide Web, although it can be used in desktop applications and other programs.

ActiveX controls are reusable software components that incorporate ActiveX technology. ActiveX controls can be embedded in Web pages to produce animation and other multimedia effects, interactive objects, and sophisticated applications. There are literally hundreds of ActiveX controls available today with functionality ranging from a simple timer control (that fires events to its container at specified intervals) to full-featured animation display control. When a control is used within a control container, such as the Web browser, it can communicate by exposing properties and methods as well as by launching events.

5.1.5 Java Applets

Java Applets are a Java class that is loaded and run by an already-running Java application such as a Web browser. Java applets can be downloaded and executed by a Web browser capable of interpreting Java, such as Microsoft Internet Explorer or Netscape Navigator. The Java language allows the writing of programs that, like ActiveX Controls, can be included in a HTML Web page to create such things as:

- animations.
- graphical objects (bar charts, graphs, diagrams, etc.).
- applications made up of collections of controls like edit areas, buttons, and check boxes.
- new controls.

Java is an object-oriented programming language derived from C++ and extended by means of a collection of libraries. In addition to creating Java applets that are loaded over the Web with the HTML page, the Java environment also allows for the creation of standalone applications.

5.2 THE INTERNET CONTROL FRAMEWORK

CIM components must not only be controlled, but also monitored. The solution of this problem consisted of two parts. The first part covered getting information about the process being monitored. The process was naturally active during this time. The second part represented visualisation of the process on the Internet providing actual information through the World Wide Web. This section presents the research into the modular sub-tasks for the CIM Internet control framework. These modular mechatronic sub-tasks included:

- *The network module.*
- *The control protocol module.*
- *The Server module.*
- *The visual feedback module.*

5.3 THE NETWORK MODULE

The network communication configuration used in the network design sub-task was based on the star topology (ref. Chapter2). This network configuration was already in use by the School of Mechanical Engineering LAN. Figure 5.2 shows the networked star topology used.

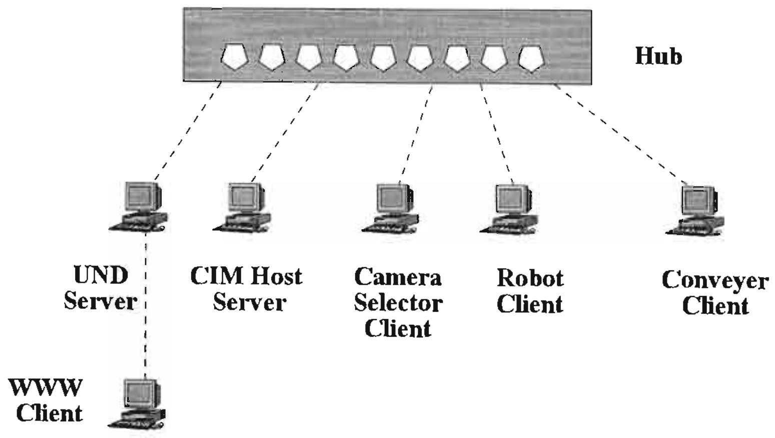


Figure 5.2. The star topology network.

The star topology used a central device with LAN cables extending in all directions. Each networked device was connected via a point-to-point link to a central device called a hub. The Star topology had the following characteristics:

- Star topologies are moderately difficult to install. The design of the network is simple, but a separate media segment must be installed for every arm of the star. This is the reason why cabled star topologies require more cable than most other topologies.
- Star topologies are relatively easy to configure. Changes do not involve more than the connection between changed network devices and the hub.
- Since all the data in a star network goes through a central point where it can be collected, stars are easy to troubleshoot. Stars can also be organised hierarchically, providing there is architectural flexibility and traffic isolation.
- Star networks handle faults relatively well. If a media fault occurs on the network, the hub can be used to identify and remove the offending link from the network. When a media segment does fail, only the segment's units are affected.

Figure 5.3 shows the 8-port DE-809TC Ethernet Hub used as the CIM cell networking device.

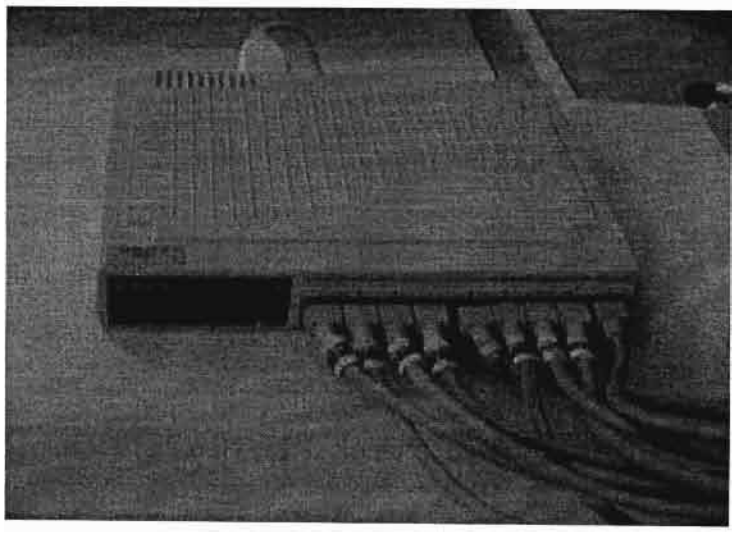


Figure 5.3. The 8-port DE-809TC Ethernet Hub used as the CIM cell networking device.

The CIM cell star topology operated using a Microsoft® workgroup, called the MR2G workgroup. This workgroup was nested within the Microsoft® Windows workgroup residing on the University of Natal (UND) Server. The MR2G workgroup formed part of the hierarchical network topology. In star topologies, electric and electromagnetic signals travel from the networked device, up its LAN cable to the hub. From there the signal is sent to the other networked devices.

5.4 THE CONTROL PROTOCOL MODULE

For Internet connectivity there exists a variety of control protocols, each used depending on the required application. The control protocols discussed in this section are available on the Microsoft Windows® network. The first step towards the implementation of a network control strategy is the understanding how these protocols function, a clear understanding of Windows Sockets and the WinSock Proxy must be established.

5.4.1 Windows Sockets and the Windows WinSock Proxy

Windows Sockets is a mechanism for interprocess communication between network applications running on the same computer, or on different computers connected using a local area network (LAN) or wide area network (WAN). It defines a set of standard API's (Application Program Interface) that an application uses to communicate with one or more other applications, usually across a network.

The Windows Sockets APIs support:

- Initiating an outbound connection for a Client application.
- Accepting an inbound connection for Server application.
- Sending and receiving data on a Client/Server connection.
- Terminating a Client/Server connection.

The specification includes a standard set of APIs supported by all Windows-based TCP/IP (Transmission Control Protocol / Internet Protocol) protocol stacks, and to be used by network applications.

In Windows Sockets, application communications channels are represented by data structures called sockets. A socket is identified by two items:

- An IP address.
- A port number.

Windows Sockets can support both point-to-point connected service (also referred to as stream-oriented communications), and multipoint connectionless service (referred to as datagram-oriented communications). When using the TCP/IP protocol suite, stream-oriented communications use TCP and datagram-oriented communications use UDP.

5.4.2 Connected Service Using TCP Sockets

Most Internet application protocols, including HTTP, and FTP (File Transfer Protocol), are connection-oriented Client/Server protocols. A Client typically initiates a connection to a Server in order to process a specific Client request. A Server waits for connections initiated by Clients, accepts those connections, and begins communicating with each Client following the rules of the specific application protocol. Communications managed expressly between a Server and its Clients in this manner form what is known as a connected service. For application protocols that use the connected service, the Transmission Control Protocol (TCP) has been long established as the transport-level protocol of choice. TCP supports the use of sockets as well to form connected communications between computers on a network. For example, a TCP socket can be formed by first associating an IP address and a TCP port. The IP address is a 32-bit number that uniquely identifies the local IP network interface. The port identifies a virtual channel used for communications at the TCP level. A stream-oriented connection is then formed by associating a local IP/TCP port pair with a remote IP address-

TCP port pair.

A Server goes through the following steps to create a TCP socket with a Client (Figure 5.4):

- The *socket()* API is used to establish a socket and associate it with a specific streaming protocol, such as TCP.
- The *bind()* API is used to associate a local IP address and port with the socket. Most Servers specify that they want to bind the socket to all local IP addresses, and indicate the well-known port for the application protocol (port 80 for HTTP, port 21 for FTP, and so on).
- The *listen()* API is used to enable inbound connections on the IP/port pair.
- When a Client connection is received, the Server uses the *accept()* API to complete the connection process, associate a different socket with the connection, and go back to the listening stage on the original socket to handle future Client connections.
- The Server uses the *recv()* and *send()* APIs to communicate with the Client.
- The Server can use the *getsockname()* API to query the local and remote IP address-port pairs.

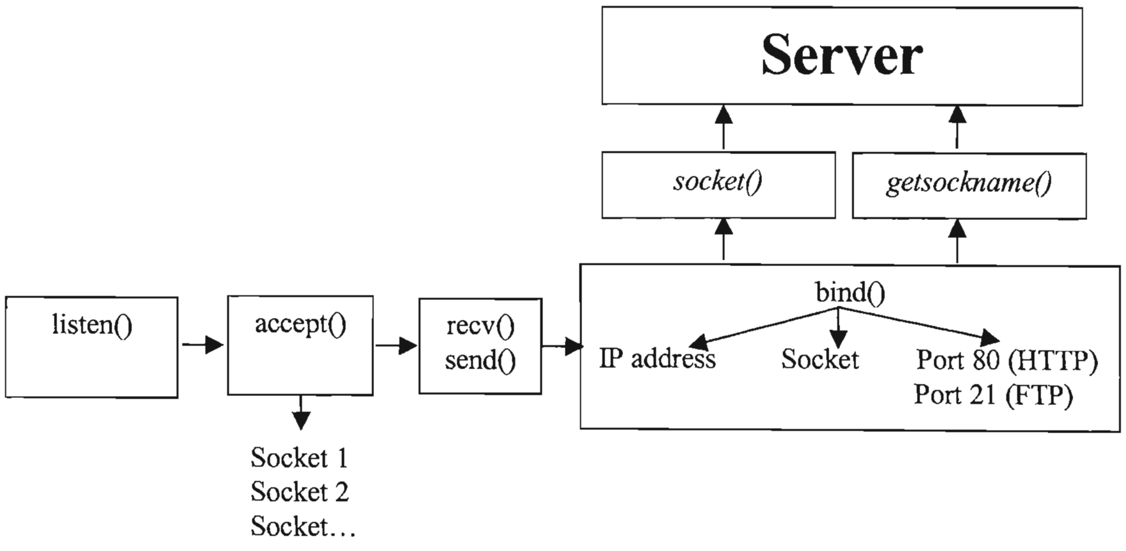


Figure 5.4. The server communication layer.

A Client typically initiates a TCP socket connection to a Server in order to process a user request. With stream-oriented TCP connections, the Client executes the following steps (Figure 5.5):

- The *socket()* API is used to establish a socket and associate it with a specific streaming protocol, such as TCP.
- The *bind()* API is used to associate a local IP address and port with the socket. Most Clients specify that they are willing to use any local IP address and port.
- The *connect()* API is used to initiate a connection to a specified IP address/port pair. The remote IP address identifies the Server, and the port identifies the service.
- The Client uses the *recv()* and *send()* APIs to communicate with the Server.
- The Client can use the *getsockname()* API to query the local and remote IP address-port pairs.

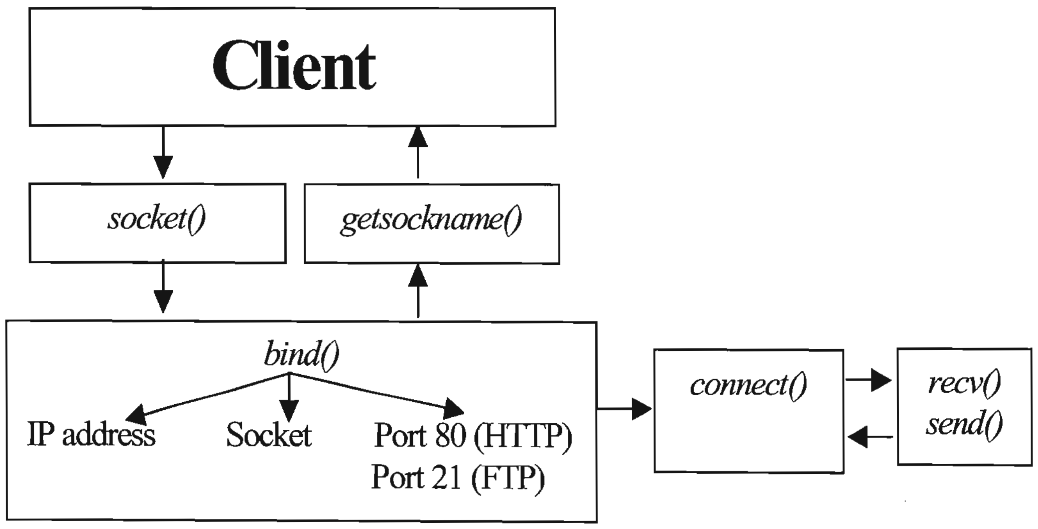


Figure 5.5. The client communication layer.

5.4.3 Connectionless Service Using UDP Sockets

Some Internet applications can use User Datagram Protocol (UDP), a transport protocol that delivers Server data in a form that offers higher throughput performance than TCP. UDP is very effective for delivering data from Servers to Clients at the highest possible speeds by using unacknowledged delivery and packaging data into small uniform-length packets called datagrams. Communications that consist of UDP datagrams sent from a Server to Clients on a network in this manner form what is known as *connectionless service*. This type of service is useful for real-time applications such as streaming audio and video. For example, RealAudio and VDOLive both use UDP to offer connectionless service to Clients.

For UDP, the Client and Server each establish a UDP socket in the following way:

- The *socket()* API is used to establish a socket and associate it with a specific datagram protocol, such as UDP.
- The *bind()* API is used to bind the socket to a local IP address-port pair.
- The *sendto()* and *recvfrom()* APIs are then used to begin immediately sending and receiving data. These APIs specify the IP port to send to, and return the IP port received from.
- While most UDP implementations consist of a Client communicating with a single Server at a time, UDP is a connectionless protocol and supports communications between a Client application and multiple Servers over a single socket.

5.4.4 The WinSock Proxy

WinSock Proxy allows a Windows Sockets application running on a private network Client to perform as though it is directly connected to a remote Internet Server application, when in actuality, the Microsoft Proxy Server serves as the proxy host for this connection. WinSock

Proxy consists of two parts: a service running on a gateway computer, and a dll (Dynamic Link Library) installed on each Client computer.

The WinSock Proxy service runs on Windows NT Server or Windows 2000 Server. It runs as a stand-alone Windows service, and is responsible for creating virtual connections between internal applications and Internet applications. The WinSock Proxy service is also responsible for doing "data pumping" between the two actual communications channels set up for a virtual connection, and acting as a TCP/IP protocol gateway. One of the benefits of this type of design is that all application-level communications are channelled through a single secured computer—the gateway computer running Microsoft Proxy Server. The WinSock Proxy service can also provide application-level event monitoring for redirected Windows Sockets applications on the private network.

With WinSock Proxy, Client applications send Windows Sockets APIs to communicate with Server applications running on Internet computers, and the WinSock Proxy service on the Microsoft Proxy Server intercepts these calls. The Microsoft Proxy Server then handles redirecting these APIs to the remote Server on the Internet. This establishes a logical communications path from the internal Client application to the Internet Server application by way of the computer running Microsoft Proxy Server.

The gateway process is not visible to either the internal network Client or the remote Server. Both Client and Server computers appear to share only a single connection to each other. In truth, both maintain separate connections to the computer running Microsoft Proxy Server through separate network hardware interfaces on the computer running Microsoft Proxy Server.

There are some side effects to using the proxy Server to access the Internet.

- External Servers see different users coming from the same address.
- A Client application that binds a socket to a specific port may fail when the requested port is already assigned to another Client. To avoid this, the Client software can either request *PORT_ANY (0)*, or try a range of ports when a bind fails with the error *WSAEADDRINUSE*.

5.4.5 WinSock Proxy Architecture

On Client computers, the WinSock Proxy uses a specially designed dll to redirect Windows Sockets API calls from the Client to remote Servers. When this dll is installed, it renames the standard Windows Sockets dlls, and the WinSock Proxy dll is given the name of the corresponding Windows Sockets dll (WinSock.dll for 16-bit; Wsock32.dll for 32-bit). This results in all Windows Sockets API calls being forwarded to the WinSock Proxy dll first, and then the WinSock Proxy dll redirecting calls to the renamed Windows Sockets dll as needed, or processing the call itself.

Once the WinSock Proxy dll is actively installed on the Client, it intercepts all Windows Sockets API calls made by applications on the Client computer. Depending on the API, and the current socket status, the Client WinSock Proxy dll may:

- Completely process the Client's request.
- Pass the request to the (renamed) actual Windows Sockets dll on the local computer (after possibly making changes to the request).

-Or–

- Need to pass control information (by use of the WinSock Proxy Control Channel) to the WinSock Proxy service on the computer running Microsoft Proxy Server.

For network communication between local applications on the internal network, the WinSock Proxy Client dll passes Windows Sockets API calls to the previously installed (and renamed) Windows Sockets dll. This allows Windows Sockets communications to continue to work normally. Also, this is true regardless of whether the previous Windows Sockets dll was obtained from a third-party TCP/IP stack or directly from Microsoft. In all cases, the Windows Sockets dll that was installed prior to WinSock Proxy Client setup is maintained for forwarding local network calls. There are two versions of the WinSock Proxy Client dll, a 16-bit version and a 32-bit version. The 16-bit version is installed on Windows 3.1 and Windows For Workgroups 3.11. The 32-bit version is installed on Windows NT and Windows 2000. Both versions are installed on Windows 98.

5.4.6 WinSock Proxy Control Channel

The WinSock Proxy service and Client dlls communicate by using a control channel that is set up when the Client dll is first loaded. The control channel uses the connectionless UDP protocol. UDP allows a single socket on the gateway computer to be used for communications with all WinSock Proxy Clients, and is faster than TCP. A simple acknowledgment protocol is used between WinSock Proxy Client and service to add reliability to the control channel. The goal is to use the control channel as infrequently as possible, and to have as few Windows Sockets APIs that require special processing on the Client computer as possible. For example, for TCP connection requests, the control channel is used to set up the virtual connection, but once the connection is set up, sending and receiving data (*send()* and *recv()* APIs) requires no special processing on the Client: the WinSock Proxy dll simply forwards these requests to the (renamed) Windows Sockets dll. This also means that the Win32 APIs ReadFile and WriteFile, which bypass Windows Sockets, will work with redirected connections.

The WinSock Proxy control channel is used for the following purposes:

- To set up TCP connections for WinSock Proxy Clients to remote Servers. When a connection with a remote application is being established, the control channel is used in establishing the virtual connection. Once the connection is established, sending and receiving data will not require use of the control channel.

- To maintain UDP communications between WinSock Proxy Clients and WinSock Proxy Servers. The control channel is used by WinSock Proxy Clients to contact the WinSock Proxy Server when the UDP socket is bound. Additionally, in order to support multiple remote applications communicating with the internal application, port-mapping information is sent to the Client dll each time a new remote peer sends data. Sending and receiving data to and from known peers does not require the control channel.
- To manage database requests between WinSock Proxy Clients and WinSock Proxy Servers. Redirection of the Windows Sockets database requests, such as DNS name resolution (*gethostbyname()*, and so on) is handled by passing the Client request to the WinSock Proxy service by using the control channel, and the response is forwarded to the Client dll by using the control channel.

When the first application on a Client attempts to make its first Windows Sockets connection, the WinSock Proxy dll is loaded and initialised. At this time, the WinSock Proxy dll does the following:

- It establishes its own WinSock Proxy control channel with the WinSock Proxy service, and notifies the service, by using the control channel, that it is active.
- The WinSock Proxy service then downloads the Local Address Table (LAT). The LAT is a routing table that consists of a list of IP address pairs, each pair indicating a range of addresses located on the internal (private) network. The LAT is used by the Client to determine which requests need to be redirected.

Once the WinSock Proxy dll has initialised service, for future connection attempts by applications, the WinSock Proxy dll attempts to determine if the application is trying to communicate with a local computer (private network) or remote computer (Internet).
For connection attempts and Windows Sockets APIs destined for a local computer, the WinSock Proxy dll simply forwards the API calls to the (renamed) Windows Sockets dll, for normal processing. If a Windows Sockets API call contains no information about the

destination (and therefore no indication as to whether it should be redirected), the WinSock Proxy component assumes it is a local request, and forwards the request to the standard Windows Sockets dll.

When a Windows Sockets database API is called by an application (*gethostbyname()*, and so on) to resolve an Internet name or address, the WinSock Proxy components work together, using the control channel, to redirect the request to the gateway computer, and have the request processed on the Internet.

The architecture of WinSock Proxy requires special processing by the Client's WinSock Proxy dll when establishing a connection with an Internet site, but once a communication channel is established, standard Windows Sockets and Win32 APIs for reading and writing a socket or file can be used with no special processing on the Client. The application performs as if it is reading and writing to the Internet site, while it is actually communicating with the WinSock Proxy service, which forwards the requests.

5.4.7 WinSock Proxy: TCP Redirection

TCP handles point-to-point, connection-oriented communications. For each TCP connection requested by an internal application, two actual connections are set up by WinSock Proxy service:

- A connection between the Client application and the WinSock Proxy service using the WinSock Proxy Microsoft Proxy Server's internal network port interface.
- A connection between the WinSock Proxy service and the Internet application using the WinSock Proxy Microsoft Proxy Server's external (Internet) port interface.

Data received from either connection is forwarded to the other connection, and both applications perform as though they are communicating directly with each other.
A TCP redirected connection is managed in the following way:

The WinSock Proxy control channel is used to set up a TCP redirected connection. Once the TCP redirected connection is set up, the control channel is not used for data transfer.

- The internal application then initiates an outbound TCP connection to an Internet site.
- The *send()* and *recv()* APIs are then called on the Client and Server. The WinSock Proxy Client dll forwards *send()* calls to the real Windows Sockets dll (these APIs do not contain addresses, they simply refer to a socket), and all data is identical to data sent in a normal (non-redirected) socketed connection. (The *ReadFile()* and *WriteFile()* Win32 APIs work on TCP redirected connections as well. For Windows 2000, these APIs are not handled by Windows Sockets and therefore are not intercepted by the WinSock Proxy dll).

Once an internal application's socket has been remotely bound, WinSock Proxy makes it appear that the socket is bound to the proxy computer's Internet interface. If the internal application calls the *getsockname()* API, the data returned will indicate that the socket's local IP address is that of the proxy computer. Thus, it appears to the application that it is on the Internet. This is necessary for protocols such as FTP, in which the Client sends its local IP address to a Server, in order for the Server to initiate a new TCP connection back to the Client.

When an internal application attempts to listen for a TCP connection initiated by an Internet application, WinSock Proxy uses the local IP address to which the application's socket is bound to determine whether the listen should be redirected. If the local IP address is that of the Internal computer's interface (a private network IP address), the listen will be local (passed to the WinSock dll). If the IP address bound to the socket is that of the Microsoft Proxy Server's Internet interface, the listen will be redirected.

When a *listen()* API is redirected, WinSock Proxy does the following:

- Listens for a socket connection on the WinSock Proxy service's Internet IP address and the same port specified as the local port in the internal application's socket bind.

- When an external site connects to the port, creates a socket connection between the internal application (on the local port specified by the application), and the WinSock Proxy service (on its internal IP address and an arbitrary port). This connection is initiated by the WinSock Proxy service, because the internal application is listening for an incoming connection.

Once an internal application's socket is bound to the Microsoft Proxy Server's Internet IP address and an inbound connection is established, a *getsockname()* API call by the application will return the proxy's Internet IP address as the local IP address, and the Internet site's IP address and port as the remote IP and port.

5.4.8 WinSock Proxy: UDP Redirection

UDP offers connectionless communications, and supports multiple applications communicating with an application over the same UDP socket. A UDP-based application uses *sendto()* to send data, specifying the destination IP address, and *recvfrom()* to receive data, returning the source IP address (Figure 5.6).

When an internal application binds a UDP socket, the WinSock Proxy service binds a UDP socket to its Internet IP address, and the same local port as used by the Client. This is the socket used for communications between all Internet peers for the internal application.

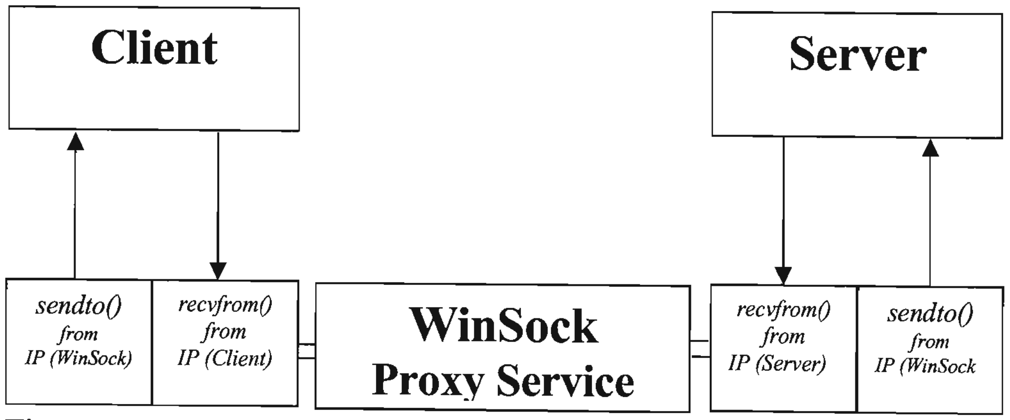

Client
Server
sendto() from IP (WinSock)
recvfrom() from IP (Client)
WinSock Proxy Service
recvfrom() from IP (Server)
sendto() from IP (WinSock

Figure 5.6. The UDP communication layer.

When an internal Client computer receives a packet over UDP from the Internet:

- The packet was actually forwarded by the WinSock Proxy service, and the source address will be that of the computer running Microsoft Proxy Server.
- The WinSock Proxy Client dll needs to change the source port and IP address to that of the actual Internet source before the internal application receives the data. However, the problem is that for UDP there can be multiple sources of data sent to one destination socket.

In other implementations, this problem is sometimes handled by having the Web Proxy Service add a header to the data (which contains the original source port and IP address) before forwarding it to the internal Client. The Client dll would then strip off this header and modify the source IP address and port passed to the application. This solution requires much work on every data packet, including a buffer copy, and may even result in the buffer size being larger than the maximum allowed. In this case, splitting the data into multiple packets needs to be supported, as well as ordering and recombining at the destination. Also, this solution prevents Win32 APIs from working. (On Windows 2000, Win32 APIs are not passed to Windows Sockets.)

Instead, the problem of multiple-source IP addresses is solved by creating a separate UDP socket in Microsoft Proxy Server for each Internet peer sending data to the Client. Each time the first data packet is received from a new Internet port and IP address, WinSock Proxy service creates a new UDP socket on a different local port, in Microsoft Proxy Server (bound to the proxy's internal IP interface). The WinSock Proxy service maintains a table that maps Internet ports and IP addresses to the port number of the WinSock Proxy Server's socket for that Internet site. Each time it changes, the mapping table is forwarded to the WinSock Proxy Client dll by using the control channel.

When the WinSock Proxy service receives data from an Internet application destined for the Client, it sends the data to the Client by using the associated socket on the Microsoft Proxy

Server. The WinSock Proxy Client dll looks at the source (remote) port number of the data packet (proxy-Server port number), and uses the table to map that to an Internet application's port and IP address. The internal application is handed the Internet port and IP address as the source.

The result is that handling UDP communications:

- Does not require extra control channels.
- Does not cause data packets to be modified.
- Does not require use of the control channel when data is sent from an Internet peer that the WinSock Proxy service already knows about.

Win32 APIs also work for reading and writing the socket.

When the internal application sends data to one of the remote peers, the WinSock Proxy dll uses the mapping table to map the destination port and IP address (specified by the internal application) to an WinSock Proxy Server port, and sends the data to the appropriate UDP socket (port) on the WinSock Proxy Server computer.

Since this mechanism requires a new socket for each Internet peer application, extra resources are used in the Microsoft Proxy Server when an internal application uses UDP to communicate with many remote peers. Most Internet Client applications that use UDP (RealAudio, VDOLive, and so on), communicate with a single Server application, so this is an efficient trade-off. For other UDP Client applications, the number of Servers communicating with the Client is usually small. (WinSock Proxy will limit the number of mappings per socket, keeping the most recently used, and re-establishing mappings as needed.). When an internal application calls *getsockname()* for a remote UDP socket, the local IP address returned is that of the proxy's Internet interface.

5.5 USING TCP/IP FOR THE CIM APPLICATION

In summation when using the WinSock control, the first consideration was whether to use the TCP or UDP protocol. The major difference between the two lies in the connection state:

- The TCP protocol is a connection-based protocol, and is analogous to a telephone. The information transmitted using this protocol is secure and always reaches its destination. This is important when network bandwidth is limited and the network is experiencing collisions and congestion.

- The UDP protocol is a connectionless protocol, and the maximum data size of individual sends is determined by the network. Data is not always secure and may never reach it's destination.

The TCP/IP protocol was selected as the transmission protocol of the CIM Internet system, as the data transmission is guaranteed.

5.6 THE SERVER MODULE

For the CIM cell Host computer to be a Server on the Internet, it had to have:

1. An address by which other computers could locate it.
2. The capability to understand and process various protocols.

When calling the Host Server on the Internet, the Server's associated *domain name* which is the friendly addressing format was used. The Server *domain name* also contained the URL. The URL address displayed the first Web page on the Server. It was the starting point from which users accessed the Server Web page.

The software used on the physical computer to make the Server was called the Internet Server Software. The particular software used for the CIM cell used the Windows® 2000 Server operating system with the Microsoft® Internet Information Server (IIS) software installation. Figure 5.7 shows the discrete conversations between a Client computer and an Internet or Web Server.

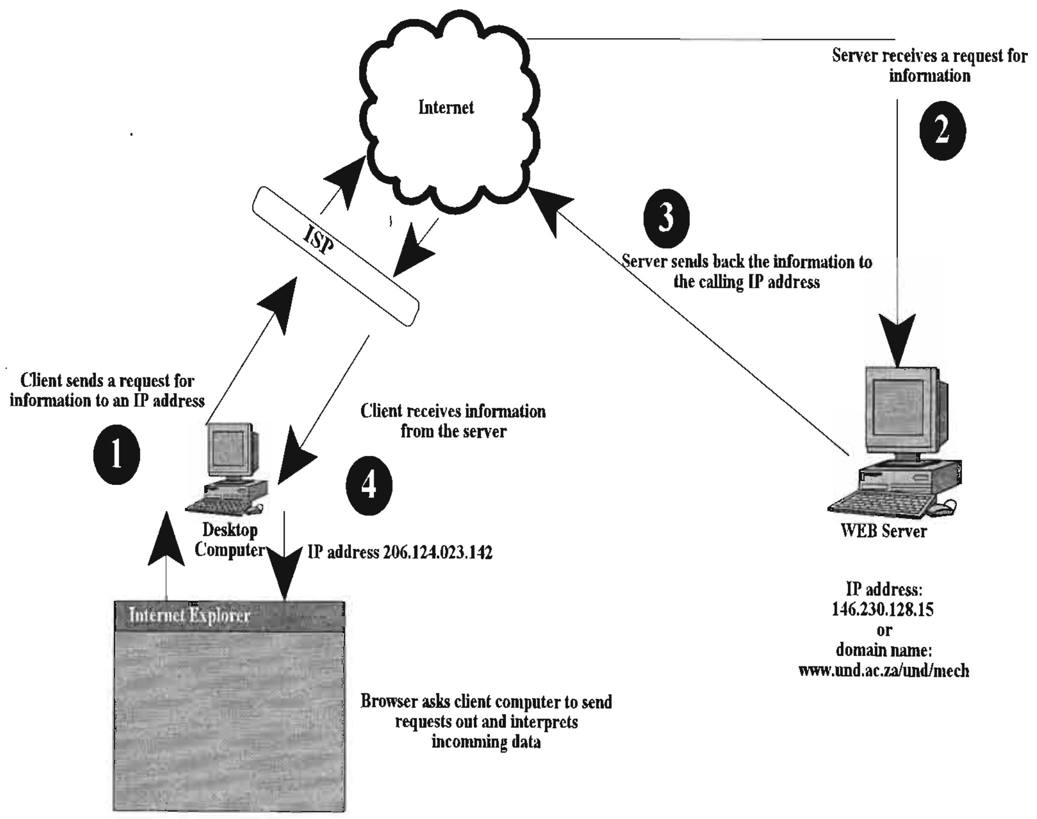

Internet
Server receives a request for information
2
ISP
3
Server sends back the information to the calling IP address
Client sends a request for information to an IP address
1
Client receives information from the server
4
Desktop Computer
IP address 206.124.023.142
Internet Explorer
WEB Server
IP address: 146.230.128.15 or domain name: www.und.ac.za/und/mech
Browser asks client computer to send requests out and interprets incomming data

Figure 5.7. Shows the discrete conversation between a Client computer and an Internet Server.

However using an IIS with VBScript on the Client and Server side, does not follow the modular mechatronic design methodology. If the appropriate Visual Basic software does not reside on either the Client or Server side, and. If the Client does not have Internet Explorer 4 (IE4) or higher the application will not execute. The supervisory Internet control strategy had to execute on all operating systems and Web browsers. However the Graphical User

Interface (GUI) Web page for the supervisory CIM control had to reside on the IIS Server. For the Web page publishing and information transfer between the Server and Client, a Server application layer was required. This application layer consisted of a File Transfer Protocol (FTP) and a Hyper Text Transfer Protocol (HTTP).

FTP makes it possible for a user to transfer files from one location to another over the Internet. URLs of files on FTP Servers begin with ftp: //.
HTTP is used by World Wide Web browsers and Servers to exchange information. The protocol makes it possible for a user to use a Client program to enter a URL and retrieve text, graphics, sound, and other digital information from a Web Server. URLs of files on Web Servers begin with http: //.

Figure 5.8 shows how these protocols fit together, during information transfer.

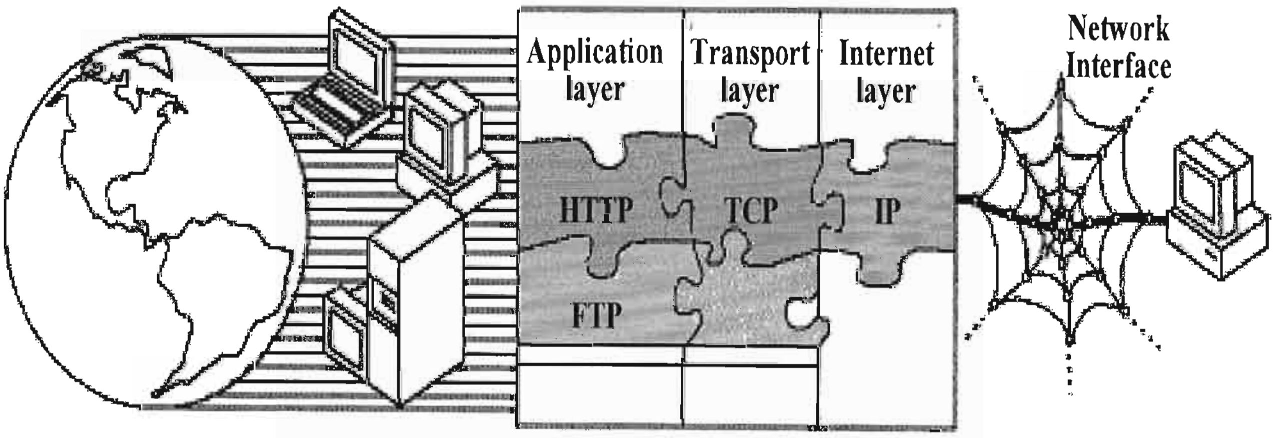


Figure 5.8. Shows how protocols fit together, during information transfer [McGee].

Using FTP and HTTP allowed the supervisory control strategy to be simplified. When a user accessed the CIM Web Server, an ActiveX executable file residing on the Server would be downloaded to the Client computer over the Internet using the FTP. The Client would then execute the ActiveX file, allowing a connection to be established between the Client and Server through the TCP/IP protocols. The CIM control program was then imbedded in the ActiveX programs. All the CIM controllers (CIM Clients) were also executing these ActiveX programs, forming a link among all the control computers in the network. These included the

Customer (Internet Client), Host Controller (Web Server), Robot Controller (Robot Client), and Conveyer Controller (Conveyer Client). The transfer of information was reduced to Client-Server communication through the sending of short text messages between each machine. Each continuously monitoring the network for the appropriate text messages in order to execute a procedure or command. This supervisory framework is modular in design and can be applied to any number of networked CIM components. Table 5.1 contains the system specifications of the CIM cell Server / Host computer.

Operating System	Microsoft Windows 2000 Server
Version	5.0.2195 Service Pack 2 Build 2195
System Name	MR2G-HOST
Processor	Genuine Intel PII~400 MHZ
BIOS Version	Award Modular BIOS v4.60PGMA
Windows Directory	C:\WINNT
Total Physical Memory	130,548 kB
Available Physical Memory	7,828 kB
Total Virtual Memory	398,516 kB
Available Virtual Memory	96,744 kB

Table 5.1. System specifications of the CIM cell Server / Host computer.

5.7 THE VISUAL FEEDBACK MODULE

The continuous and steady image stream feedback from the CIM cell was necessary for the Internet user to control the CIM components.

The front-end imaging system of the *visual feedback module* consisting of three modules, the image sensor, the image acquisition and interface card and the computer processor. The *visual* *feedback module* consisting of four overhead cameras monitoring the CIM cell. The camera image was relayed through a camera switch selector circuit. The camera selector circuit was based on the *relay power module* circuit (ref. Chapter 4) and was controlled by the Camera Selector Client. The appropriate camera view was selected by software algorithm to show the optimum set-up camera view of the CIM process being executed.

The *visual feedback module* comprised of a camera array utilising four Tedelex Universal analogue cameras. The system could use up to 8 cameras. The camera selection was controlled by the Camera selector Client controller and the image capture process was controlled by the Host Server. Each camera signal was relayed by cable to the camera selector circuit. The selector circuit was controlled by an Advantech® PCL-836 digital I/O card (ref. Chapter 6). The card was controlled using a Visual Basic software library.

Once the appropriate view was selected, the single camera view signal was then captured using a FlashPoint^{3D} frame grabber. The FlashPoint^{3D} frame grabber card is a AGP board that transfers images significantly faster than more conventional formats. For any PC system that requires quality rendering of 2-D graphics, the minimum support is double buffering for up to 800×600×16 bpp (bits per pixel). For any PC system that requires 3-D rendering in any fashion, whether using software or hardware acceleration, the minimum requirements are 800×600×16 bpp double-buffered, Z buffer, and 1.25-MB local texture cache. On AGP systems, there is no requirement for local texture cache. FlashPoint^{3D} captures and displays superior-quality 24-bit colour and monochrome video at resolutions up to 1600x1200 with simultaneous VGA and S-Video output. Microsoft® Visual Basic 6 (VB6) was used to communicate with the FlashPoint^{3D} frame grabber card, which was controlled via a library of high level functions stored in the dynamic link library (dll), *fp3d.dll*. This dll was supported by the Integral Software Developers Kit (SDK).

The *visual feedback module* hardware had to produce a real-world representation of the CIM cell under control. This visual representation had to consist of a real-time video feed. Three possible Web-casting methods were researched:

1. IIS Web-casting.
2. Videoconferencing.
3. Java applet.

5.7.1 IIS-Webcasting

The software used on the physical computer to make it a server that can speak to protocols of the Internet and respond accordingly is called Internet server software. The particular Internet server software manufactured by Microsoft is Internet Information Server (IIS). An IIS application is a Visual Basic application that resides on the Web / Image Server and responds from requests from the Client browser. An IIS application uses HTML to present its user interface and uses compiled Visual Basic code to process requests and respond to events in the browser.

To the user, an IIS application appears to be made up of a series of HTML pages. To the developer, an IIS application is made up of a special type of object called a web-class, that in turn contains a series of resources called web-items. The web-class acts as the central functional unit of the application, processing data from the browser and sending information to the users. A series of procedures is defined that determine how the web-class responds to these requests. The web-items are the HTML pages and other data the web-class can send to the browser in response to a request.

The Image Server would capture the live image stream and embed the video stream in the IIS application Web page residing on the Server. The Client would then view the live video feed upon request. This Web-casting method caused network overloading and congestion. The University of Natal firewall would also not allow this type of video transmission, due to network overloading.

5.7.2 Videoconferencing

Microsoft® NetMeeting™ lets people communicate and collaborate over the Internet and corporate intranets, holding real-time meetings where they can see, hear, and exchange information with each other. NetMeeting supports industry standards and offers rich data conferencing, audio conferencing, and video conferencing capabilities in a seamless, easy-to-use client. NetMeeting also offers a platform for real-time communication, enabling third-party vendors to easily integrate Internet conferencing into their applications and services.
Microsoft NetMeeting delivers a complete Internet-conferencing solution for real-time communication and collaboration over the Internet or corporate intranets. The new version of NetMeeting offers new features and capabilities to make the product easier to use, more powerful, faster, and more manageable. Support for the International Telecommunications Union (ITU) H.323 and T.120 standards for audio and video conferencing, and multipoint data conferencing, respectively, enables cross-vendor, cross-product, and cross-platform interoperability. NetMeeting 2.0 includes the following key components:

- Standards-based multipoint data conferencing. Users of NetMeeting can collaborate and share information with two or more conference participants in real time. During a conference, users can share applications on their computer and work together by allowing other conference participants to see the same information on their screens, thereby facilitating the review, editing, and presentation of information. In addition, users can exchange and mark up graphics and draw diagrams with the electronic whiteboard, communicate using written text with the NetMeeting chat program, and send files to other conference participants.

- Standards-based video conferencing. With a video-capture card and camera, users of NetMeeting can send and receive video images for face-to-face communication during a conference. The switchable audio and video feature of NetMeeting allows users to switch among participants they communicate with during a meeting. In addition, NetMeeting also offers support for Intel MMX technology, improving video and audio performance during a call.

- Standards-based multipoint data conferencing. Using a sound card, microphone and speakers, participants using NetMeeting can talk to friends, family, and business associates over the Internet and corporate intranets. Internet phone discussions can be enhanced using data- and video-conferencing capabilities.

NetMeeting is a novel way to transmit and receive live video streams over computer networks. NetMeeting is readily available for download from the Microsoft® Web site and is available on all the Windows® operating systems. NetMeeting 3 has enhanced features including a full software developers kit (SDK). The SDK contains the Windows® Application Program Interface (API), to embed a NetMeeting user interface ActiveX control in a Web page application. The front-end of the application consists of the active Web page, while the back-end contains the API control and image compression. For the Image to be transmitted to the Client, the Client must have NetMeeting 3 or higher as default videoconferencing software. The advantage of using NetMeeting 3 is that it is not a third party application. As with the IIS Web-casting approach, network congestion was the major drawback to using this technology as a CIM cell module. The video feed was not always available and image buffering caused image transmission delays.

5.7.3 Java Applet

The Java programming language is both a high-level programming language and operating platform. The Java programming language in unique in that a program is both compiled and interpreted. The Java compiler first translates a program into an intermediate language called Java-bytecodes. The *platform independent* codes are interpreted by the interpreter on the Java platform. The interpreter executes each Java-bytecode instruction on the computer. Compilation occurs only once and interpretation occurs each time the program is executed. Figure 5.9 illustrates how this works. Java-bytecodes make "write once, run anywhere" possible. The application program can be compiled on any platform that has a Java compiler. The bytecodes can then be run on any implementation of the Java Virtual Machine (Java VM). The Java VM is available on all Netscape Navigator™ and Microsoft® Internet Explorer Web browsers. This Web-casting technique follows the modular design approach as the Java VM allows the same program to run on Windows 2000, a Solaris workstation or on an IMAC.

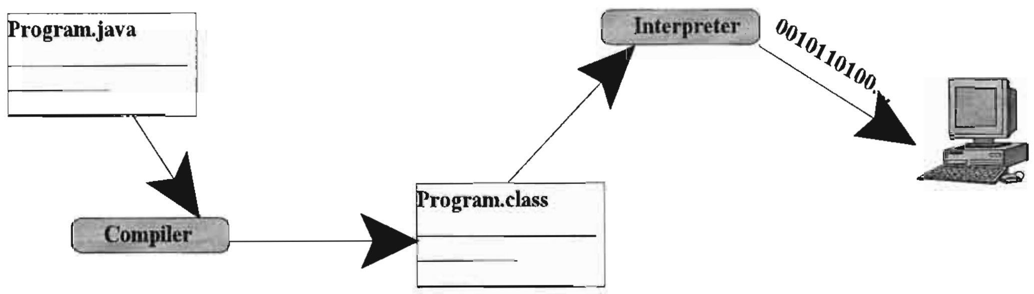


Figure 5.9. Illustrates the compilation and interpretation of executed platform independent program.

The visual feedback module used Server push technology, where the video was made up from a stream of still images, and sent by a Java program to a Java applet via a socket, and interpreted by the applet in JPEG format, as shown in Figure 5.10.

The FlashPoint^{3D} frame grabber, captured a still image every 250 milliseconds and saved this image to the image directory of the Web page on the image Server. The Java applet was imbedded in the Image Server Web page. The Java applet allowed the image to be updated at 4 frames per second. The size of the still image was 15kB, which was easily transmitted to the Client machine, without causing network congestion.

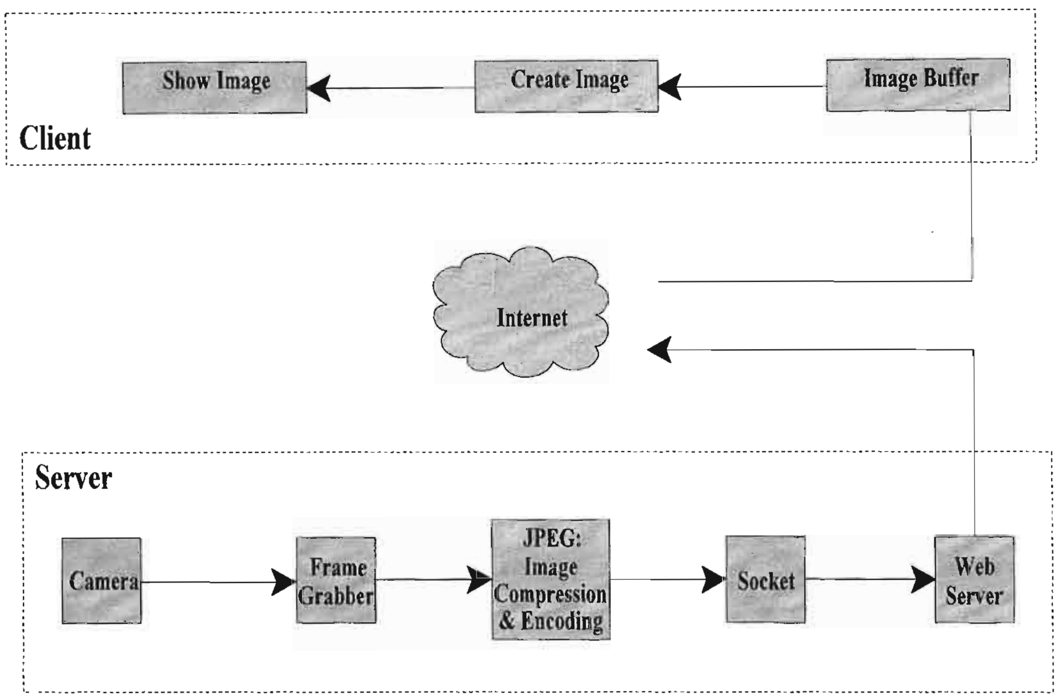


Figure 5.10. Video transmission diagram.

If the transmission of the video stream was not restricted by the University of Natal firewall, then using the NetMeeting video conferencing module would have been the best solution. NetMeeting provides a low-cost embedded web solution, which does not relay on third party software platforms.

CHAPTER 6

6 THE PC-BASED CONVEYER SYSTEM

In the design of the Computer Integrated Manufacturing systems or cells, the movement and transfer of materials were regarded as the base structure to the design of the CIM system layout. The assembly and production machines were interlinked through the chosen material handling systems. This is best achieved by the optimal arrangement of machine centres and equipment, with consideration to ergonomics, minimising the distance and trying to obtain a line of flight between machine centres [Pemberton].

Time studies in material handling systems have shown that the average workpiece spends 95%-98% held in storage or in transit, whilst only 2%-5% of the time in assembly [Allegri]. The function of material handling is a non value adding entity to the product, and it accounts for 30%-50% of the product cost [Allegri]. The key objective of an efficient material handling system is to get the right materials or parts to the right machines at the right time. If materials arrive at the incorrect time intervals, machines either become idle or backed up, excess inventory occurs.

The material handling transfer process within a CIM system can be defined as:

1. Continuous transfer.
2. Synchronous transfer.
3. Asynchronous transfer.

The transfer system implemented depends on the type of assembly operation performed. Continuos transfer systems consist of the work and the assembly tool heads moving together. Synchronous transfer systems involve all the work pieces moving together, whereas in asynchronous transfer systems the workpiece only moves once the assembly operation is completed.

The Mechatronics and Robotics Research Group (MR^{2}G) at the University of Natal School of Mechanical Engineering, has developed a CIM cell. Investigating the material handling functions within the manufacturing cell, revealed that it was the primary component which linked each of the individual manufacturing cell operations. The material handling components controlled using low-cost modular mechatronic modules were the conveyer and robot systems. Figure 6.1 shows an overview of the CIM cell in the (MR^{2}G) laboratory.

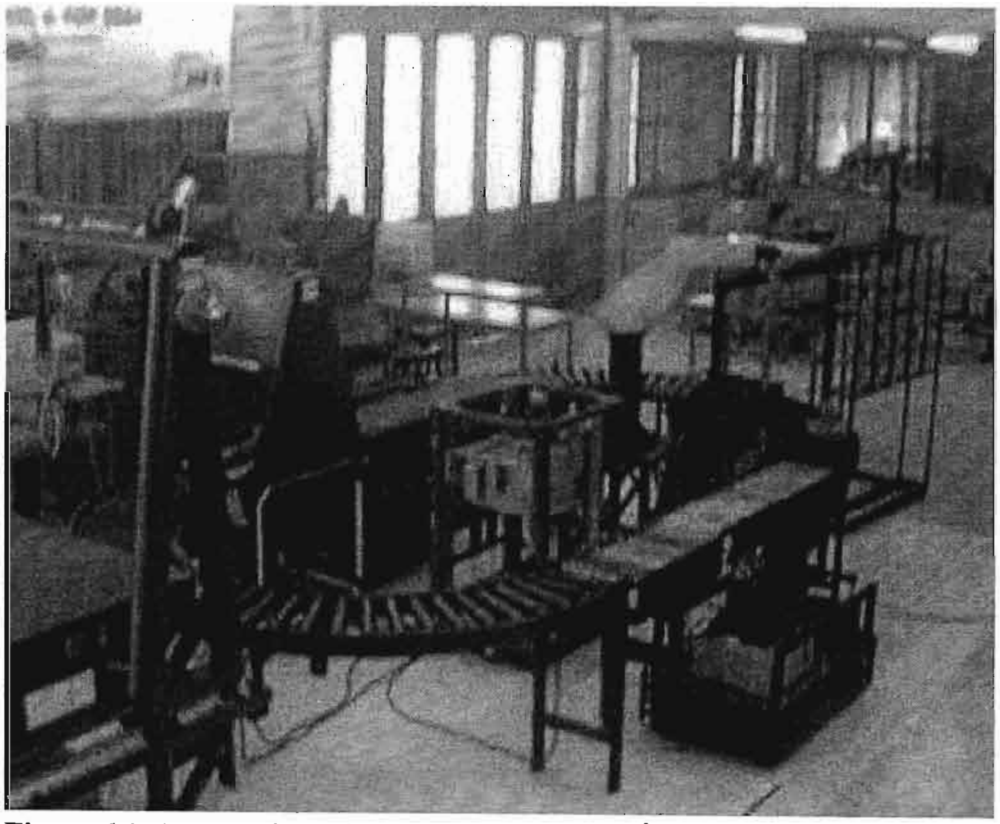


Figure 6.1. An overview of the CIM cell in the MR^{2}G laboratory. Note the oval roller conveyer system.

This chapter will detail the specific modular mechatronic sub-tasks used in the design and implementation of the PC-based conveyer system for CIM Internet control.

6.1 ROLLER CONVEYERS

Roller conveyers are line restricted devices which consist of roller elements mounted between two side members. The roller elements are of tubular design which encase roller bearings, positioned at either end. Roller conveyer systems are extremely common. Roller conveyer simplicity of design as well as the flexibility and ease at which the system can be maintained adds to the use of such systems, especially in high production industries. There are two types of roller conveyer systems available, powered and unpowered (gravitational) rollers.

The roller conveyer system in the MR^{2}G CIM cell used an interlinking drive method, where the power is transmitted from a belt, to roller, and back to another belt. By using belts as the driving medium the maintenance requirements and installation costs were reduced. This drive method also allowed for the isolation of conveyer sections for specific scheduling operations. Figure 6.2 shows the interlocking belts of the powered roller conveyer system.

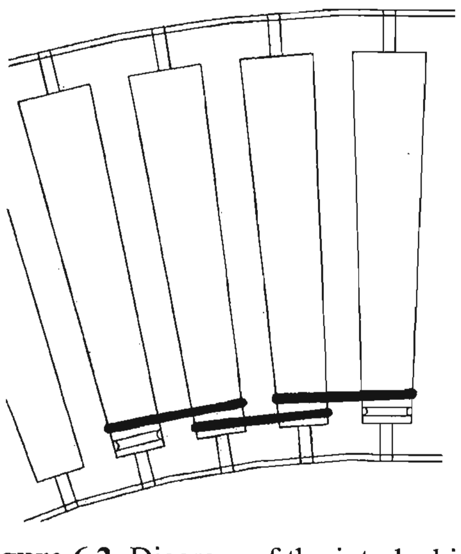


Figure 6.2. Diagram of the interlocking belts of the powered roller conveyer system.

The layout of the CIM conveyer system had been designed with modularity and flexibility in mind.

6.2 THE PC-BASED ROLLER CONVEYER SYSTEM

The primary material handling system in the CIM cell was the continuous roller conveyer system. The roller conveyers consisted of a series of tubes or rollers that were perpendicular to the direction of travel. The primary function of the roller conveyer was to transport the work pieces between assembly, machining, and inspection stations. The CIM cell material handling system consisted of an oval track conveyer type configuration. The conveyer system was divided into six different tracks, combined to form the oval configuration. The conveyer subsystem consisted of two straight sections (ref. Figure 6.3), each driven by a 12V DC servomotor, which allowed for docking by the AGV, during material transfer. The remaining four sections comprising 90° semicircular sections (ref. Figure 6.4), each driven by two identical 12V DC servomotors. In total the conveyer system consisted of ten 12V DC servomotors. The motors were required to drive inertial and frictional loads in both directions. The motor drive circuit had to incorporate a logic decoding module that was capable of accepting Transistor-Transistor Logic (TTL) input signals containing switch-on and direction information (ref. Chapter 3). The design approach to the power electronics included a high level of modularity. Rather than developing specific circuits, the design concentrated on developing robust power electronic modules that could be flexible with respect to application and implementation.

The point of entrance and exit of each conveyer was monitored by a long range infra red sensor (ref. Chapter 4). The conveyer section that was used to mate with the AGV had an additional sensor, to determine whether the pallet transfer between the AGV and the conveyer was successful. An additional sensor was placed at the section of the conveyer where the robot would place or remove the workpiece during part manufacturing processes. In total 14 infra red sensors were used.

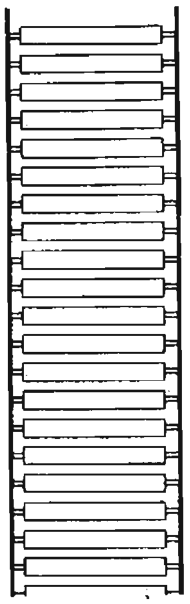


Figure 6.3. Conveyer straight section module.

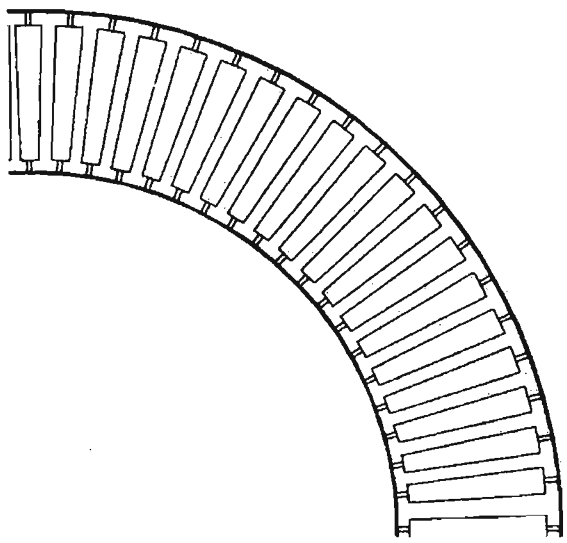


Figure 6.4. Conveyer 90° curved module.

The control structure of the modular mechatronic PC-based conveyer system was hierarchically arranged, with the Internet host controller at the top of the hierarchy. The PC controller consisted of a PI 120Mhz computer, which served as the primary conveyer controller computer. At the lower level was an Advantech® PCL-836 6-channel counter-timer and digital I/O card. From the PC controller the PCL-836 interface card was connected to an Advantech® PCLD-880 industrial terminal board. The terminal board was connected to the power electronics cabinet via an interconnecting harness. The PC controller performed two major tasks:

1. On-line user interaction and sub-task scheduling from the graphical user interface (GUI).
2. Sub-task coordination with the servo control and interface cards.

Figure 6.5 shows the hierarchical control structure of the PC-based conveyer system.

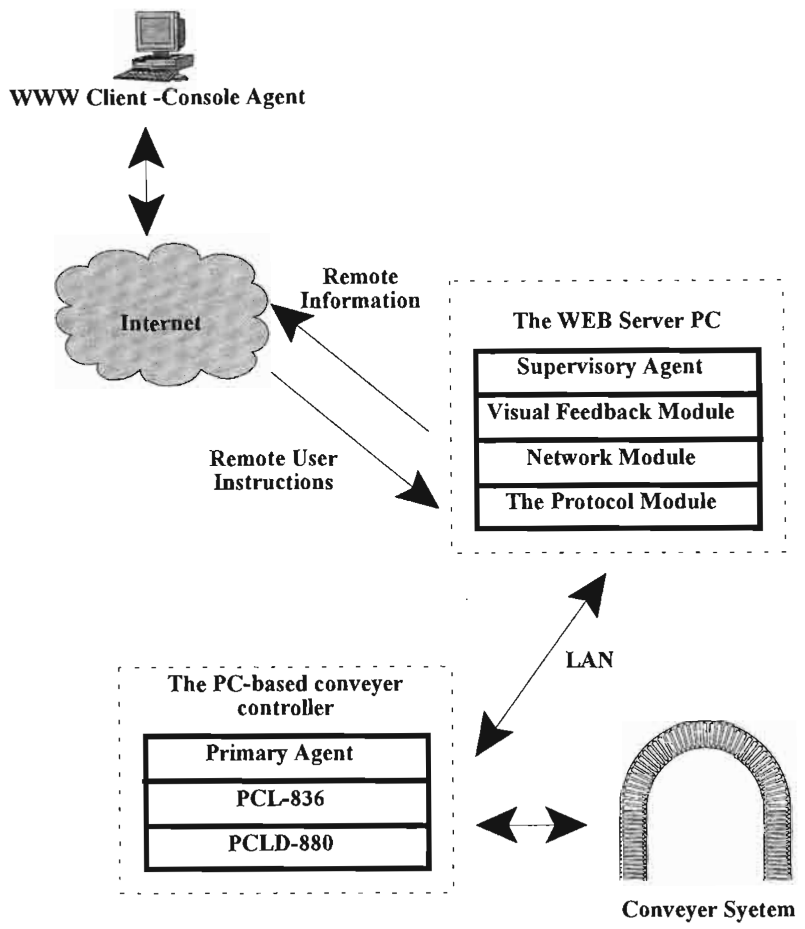


Figure 6.5. shows the hierarchical control structure of the PC-based conveyer system.

The modular mechatronic PC-based conveyer system was integrated with the CIM cell as shown in Figure 6.6. From Figure 6.6 it can be noted that the conveyer system consisted of modular interchangeable sections, which were integrated with the machining, inspection and assembly stations of the CIM. The conveyer sections, 1 to 10, were controlled and individually connected to the system cabinet with interchangeable connectors, to allow for the change of the conveyer configuration without any change of hardware.

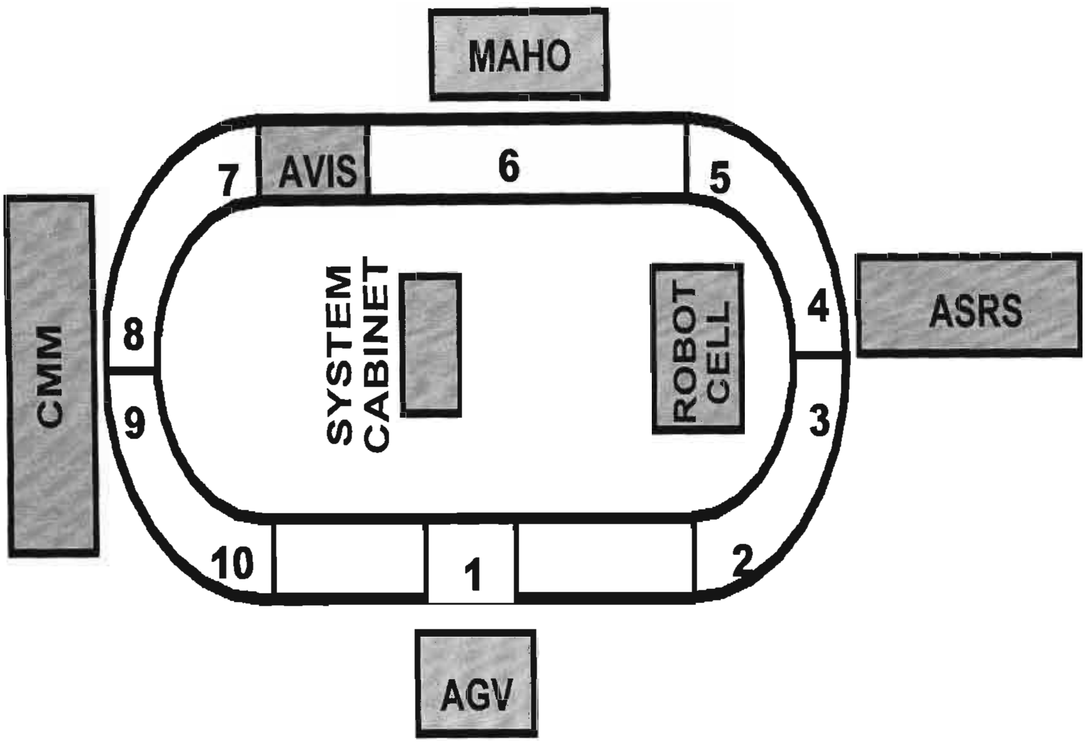


Figure 6.6. Schematic of the conveyer system configuration. Note the conveyer integration and interaction with the other CIM components.

6.3 DIGITAL I/O INTERFACING

The conveyer system implemented a digital I/O control system that was based on the transistor-transistor-level (TTL) system for the digital signal voltage ranges. The digital signals conveyed the information in a conditional manner in which the information took on two states, HIGH/LOW, TRUE/FALSE, ON/OFF etc. The two states of the digital standard were defined by the voltage ranges, a high state range where $a < V < b$ and a low state range where $c < V < d$ (ref. Figure 6.7). The precise values of the ranges varied amongst the different digital signal standards. The level of the signal voltages therefore determined the state of the information.

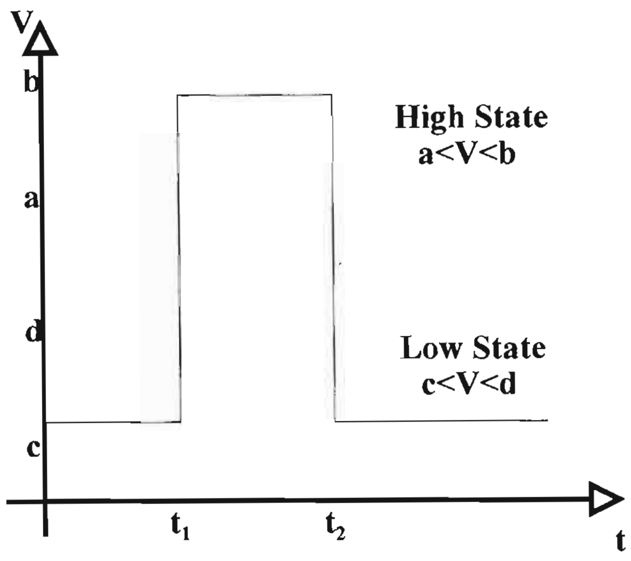


Figure 6.7. A typical digital signal conveying information on an event starting at time t_1.

6.3.1 Interfacing Using the PCL-836 Interface Card

The PC-based control system for the conveyer was implemented using an Advantech® PCL-836 digital I/O counter card. The PCL-836 was a general purpose counter/timer and digital I/O card for PC/AT compatible computers. It provided six 16-bit counter channels. It also included 16 digital outputs and 16 digital inputs. Two 8254 chips provided a variety of powerful counter/timer function modes. The PCL-836 included a unique digital filter to eliminate noise on the input signal. The frequency was adjustable to provide more stable output readings. Figure 6.8 shows the PCL-836 card.

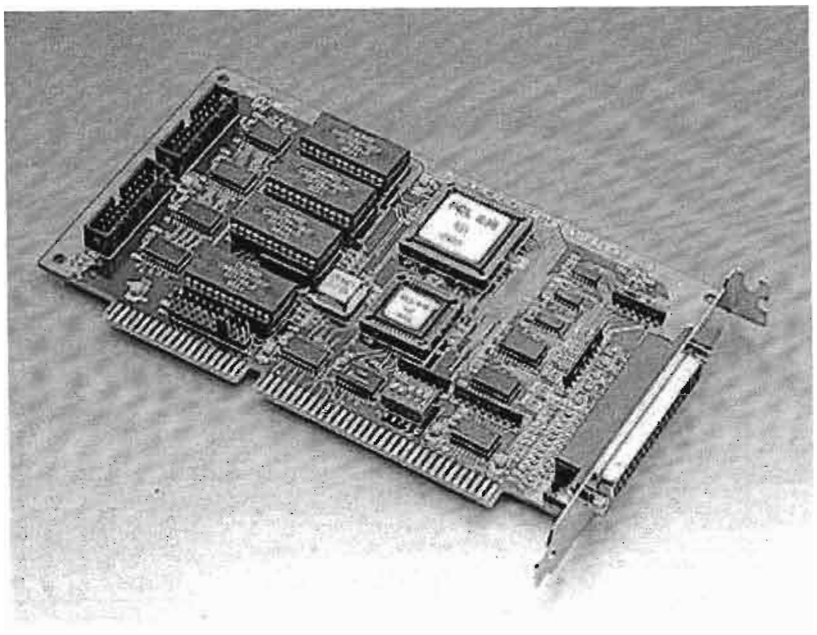


Figure 6.8. The PCL-836 I/O counter card.

The two digital ports were set as input ports, and two digital ports were set as output ports. Ports that were configured as input ports reflected the digital inputs on the port when read. Ports that were configured as outputs were set and latched to the state most recently written to the port. The bit definitions for the three ports are given in Table 6.1.

PORT	**I/O**	**BIT DEFINITION**							
		2^7	**2^6**	**2^5**	**2^4**	**2^3**	**2^2**	**2^1**	**2^0**
A	**INPUT**	A7	A6	A5	A4	A3	A2	A1	A0
B	**INPUT**	B7	B6	B5	B4	B3	B2	B1	B0
C	**OUTPUT**	C7	C6	C5	C4	C3	C2	C1	C0
D	**OUTPUT**	D7	D6	D5	D4	D3	D2	D1	D0

Table 6.1. Bit definitions for the four 8-bit ports. The data refers to the line number of the port.

During the control of the PC-based conveyer system the PCL-836 card was used to:

1. Receive the feedback information, for the long range IR (ref. Chapter 4) sensors.
2. Relay and control the actuation signals to the *relay power modules* (ref. Chapter 3) of the system cabinet.

Microsoft® Visual Basic 6 (VB6) was used to communicate with the PCL-836 card, which was controlled via a library of high level functions stored in the dynamic link library (dll), *driver.dll*. This dll was supported by the Advantech Software Developers Kit (SDK). The functions required for the control of the digital subsystem are listed in Table 6.2.

The I/O ports of the PCL-836 card a preset as input and output ports. When executing the function calls, input port A is referred to as 0 and input port B is 1. The data written and read by each port is interpreted as a decimal value.

Eg.

$\xcancel{0}$	1	2	$\xcancel{3}$	4	5	$\xcancel{6}$	7

Line 0,3,6 are high, thus

$$2^0 + 2^3 + 2^6 = 1 + 8 + 64 = \underline{73}$$

The decimal output for lines 0,3 and 6 to go high is 73.

DLL Function	Usage
DRV_DeviceOpen	Opens the I/O card.
DRV_GetAddress	Finds the HEX address of the board.
DRV_DioWritePortByte	Writes data to the output port.
DRV_DioReadPortByte	Reads the entire input port.
DRV_GetErrorMessage	Monitors the card for any errors.
DRV_DeviceClose	Closes the I/O card.

Table 6.2. The high level functions used to control the digital ports of the PCL-836 card.

6.4 DESIGN SPECIFICATIONS OF THE CONVEYER CONTROL SYSTEM

- The control system had to ensure the smooth and coordinated functioning of the conveyer system. The material handling function of the conveyer system had to ensure accurate placement of material for robot interaction.

- The control system had to be computer-based. The conveyer control system had been designed primarily as an Internet-based technology. The design of a computer-based control system was required to facilitate the interaction of the conveyer system with the other CIM components.

6.5 THE CONVEYER CONTROL MODULE

The feedback information for the conveyer material handling system was collected using the byte array feedback technique. The control module for the conveyer material handling system therefore required fourteen feedback signals to convey the position of the pallet along the conveyer track (ref. Figure 6.9). A list of feedback signals used in the control module for the conveyer system is given in Table 6.3.

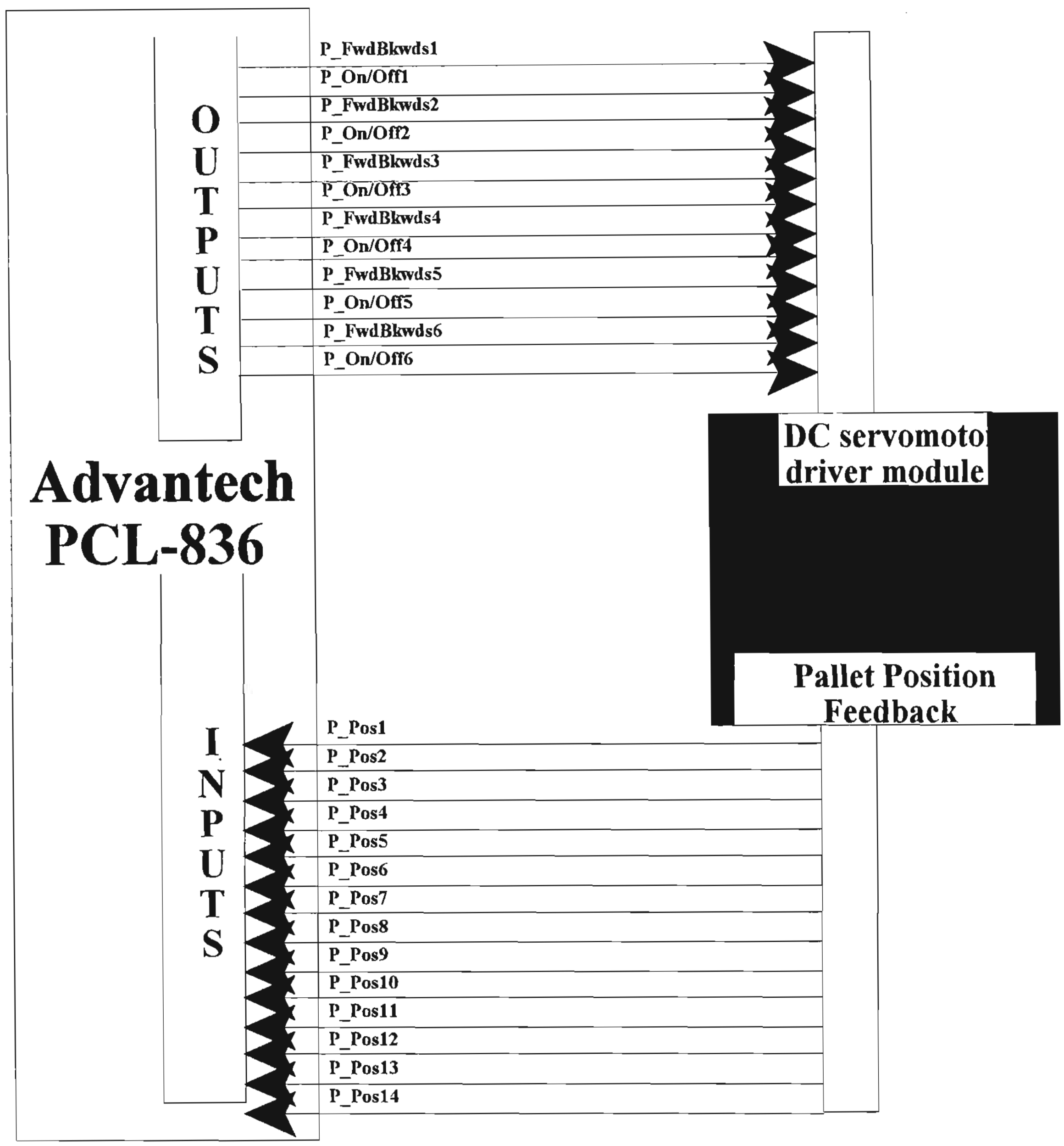


Figure 6.9. A block diagram depicting the structure of the conveyer control module.

The conveyer control module was required to generate 12 activation signals used to operate the power electronics developed to control the DC motors. The activation signals for the conveyer control module are listed in Table 6.4. A VB6 project, *ConveyerClient.vbp*, was developed to implement the conveyer control algorithm.

Name	Control Function	Port	Line
P_Pos1	Entrance/Exit conveyer 7	A	A0
P_Pos2	Entrance/Exit conveyer 8	A	A1
P_Pos3	Entrance/Exit conveyer 9	A	A2
P_Pos4	Entrance/Exit conveyer 10	A	A3
P_Pos5	Entrance/Exit conveyer 1 left	A	A4
P_Pos6	Entrance/Exit conveyer 1 right	A	A5
P_Pos7	Entrance/Exit conveyer 2	A	A6
P_Pos8	Entrance/Exit conveyer 3	A	A7
P_Pos9	Entrance/Exit conveyer 4	B	B0
P_Pos10	Entrance/Exit conveyer 5	B	B1
P_Pos11	Entrance/Exit conveyer 6 right	B	B2
P_Pos12	Entrance/Exit conveyer 6 left	B	B3
P_Pos13	AGV docking	B	B4
P_Pos14	Robot pickup station	B	B5

Table 6.3. The feedback signals used in the conveyer control module. The conveyer numbers refer to Figure 6.6.

Name	Control Function	Port	Line
P_FwdBkwds1	Reverse the direction of travel 7&8	A	A0
P_On/Off1	Start/Stop travel of conveyer 7&8	A	A1
P_FwdBkwds2	Reverse the direction of travel 9&10	A	A2
P_On/Off2	Start/Stop travel of conveyer 9&10	A	A3
P_FwdBkwds3	Reverse the direction of travel 1	A	A4
P_On/Off3	Start/Stop travel of conveyer 1	A	A5
P_FwdBkwds4	Reverse the direction of travel 2&3	A	A6
P_On/Off4	Start/Stop travel of conveyer 2&3	A	A7
P_FwdBkwds5	Reverse the direction of travel 4&5	B	B0
P_On/Off5	Start/Stop travel of conveyer 4&5	B	B1
P_FwdBkwds6	Reverse the direction of travel 6	B	B2
P_On/Off6	Start/Stop travel of conveyer 6	B	B3

Table 6.4. The activation signals used in the conveyer control module.

The control logic for the conveyer control module was based on the value of the lpDioReadPort variable which was continuously read from the input ports. The vale of lpDioReadPort was continuously cross referenced to possible conveyer control events. Once the control algorithm located the current system event the appropriate event handler was called.

6.6 THE SUPERVISORY CONVEYER CONTROL STRUCTURE

The conveyer control strategy allowed for a supervisory agent to control the conveyer system remotely, or the conveyer could be controlled from the primary PC-based controller (ref. Figure 6.5). Once the conveyer was under the control of the supervisory Internet agent, a local operator was not able to control the conveyer system.

The conveyer system was controlled using preprogrammed material handling processes. This allowed for the Internet user to be part of the open-loop control structure, while the conveyer controller was responsible for part scheduling and material handling. The scheduling strategy included the continuous monitoring of the robot state and interacting with the host controller. Chapter 9 provides a full operational review of the PC-based conveyer system.

CHAPTER 7

7 THE PC-BASED PUMA ROBOT

7.1 THE INTERDISCIPLINARITY OF MECHATRONICS AND ROBOTICS

Mechatronics is an interdisciplinary undertaking, where the knowledge from different domains has to be integrated in an optimal way (ref. Figure 7.1) [Siegwart].

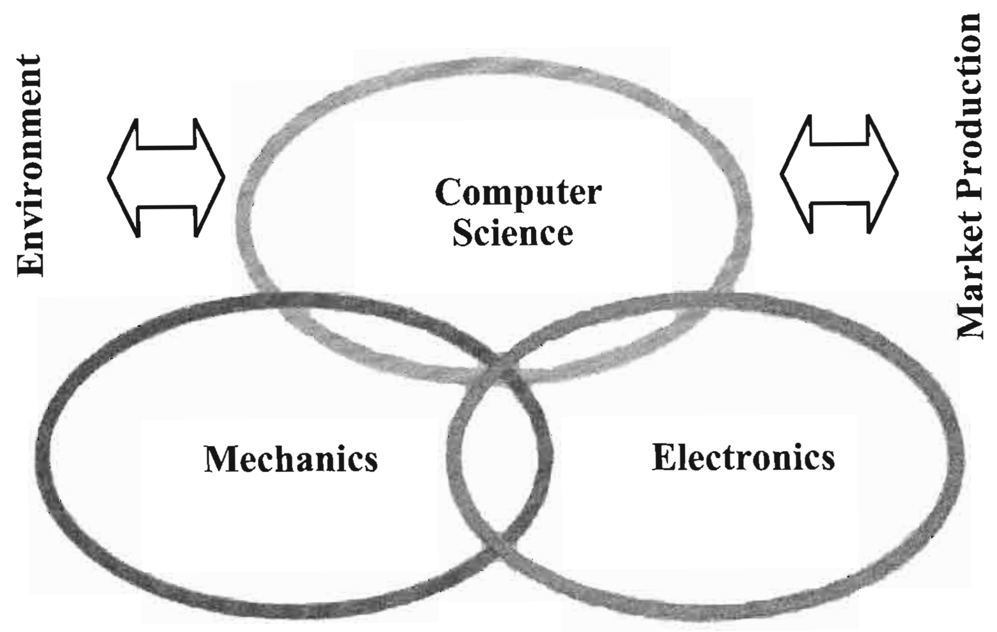


Figure 7.1. Mechatronics is the synergistic integration of precision mechanical engineering, electronics, intelligent control, and systems.

Whereas until recently many engineers have been trained in one very specific discipline, today's product designers need a more interdisciplinary education and knowledge in system design [Siegwart]. Consequent to the revolutionary progress in microelectronics and computer science of the last decades, the composition of existing products has undertaken a drastic change (ref. Table 7.1).

Products: 1960 and Today				
	1960		2000	
Product	**Mechanics**	**Electronics/ Informatics**	**Mechanics**	**Electronics/ Informatics**
Car	90%	10%	50%	50%
Calculator	100%	0%	10%	90%
Camera	100%	0%	30%	70%

Table 7.1. The revolutionary progresses in microelectronics and computer science of the last decades [Siegwart].

System integration in the *development direction* is very important since making valuable products is one of the main objectives in the engineering field. For the study of advanced modular mechatronic-based robotics, integration of existing technologies is crucial, including mechanical and electrical engineering, computer science, and human science and engineering.

In today's competitive marketplace, there is a need for flexibility in the manufacturing processes in order to respond quickly to market dynamics. Production speed, productivity and efficiency must continually improve without compromising quality, customer satisfaction or profitability. The field of computer-based automation allows for increased productivity and the delivery of end products of uniform quality. Most automated manufacturing tasks are carried out by special-purpose machines which are inflexible and programmed to predetermined tasks, and these are referred to as *hard automation* systems [FU et.al.]. The inflexibility and high cost of these machines have led to the use of robots capable of performing a variety of tasks.

In 1954 George C. Devol filed an US patent for a programmable method for transferring articles between different parts in a factory. He wrote in part:

> "*The present invention makes available for the first time a more or less general purpose machine that has universal applications to a vast diversity of applications where cyclic control is required.*"[Stadler].

With this description he could have been describing any manufacturing machine. The Robot Institute of America gives a more precise description of industrial robots:

> "*A robot is a reprogrammable multi-functional manipulator designed to move materials, part tools, or specialised devices, through variable programmed motions for the performance of a variety of tasks.*"[FU et.al.].

In the field of robotics there is a general trend toward increased robot control and flexibility. There is a need for a modular low cost industrial electronic system and software for robot controlling and interfacing. Presently there is a limitation on robot control and flexibility resulting from outdated controllers and power electronics. Standard robotic systems comprise of a robot and dedicated controller. Robot performance is directly dependant on the mechanical elements of the robot and more on the sophistication of the controller. With the advances in modem electronics and computers, controller upgrading is the logical route to follow for improved efficiency and accuracy. It then made sense to relinquish the dedicated controller and replace the control with a PC-based system. This PC-based robot was integrated and controlled in as modular mechatronic CIM cell using Internet manufacturing technologies.

This interdisciplinary integration allowed for the investigation into the field of modular mechatronics. The integration of modular mechatronic across the traditional boundaries of mechanical engineering produced more reliable, cost affective and flexible systems. When moving away from the dedicated robot controller, a few practical problems presented themselves. A thorough investigation into power electronics and robot system requirements revealed the need for driver circuits, feedback translation devices and a PC-based control package. Incorporating a PC-based control package allowed for online robot manipulation and programming, eliminating outdated programming techniques which was downtime associated. As the need for dedicated controllers were eliminated, a variety of robot systems could be upgraded by simply changing geometric and power requirements. In many instances a mere modification in the control and software would dramatically enhance the accuracy, flexibility and capabilities of the industrial robot. These factors were crucial in the design and implementation of the PC-based modular mechatronic robotic control system, described in this section. This chapter will detail the specific modular mechatronic sub-tasks used in the design and implementation of the PC-based PUMA robot for CIM Internet control.

7.2 ROBOT KINEMATICS

Most mechatronic manipulators and actuators are modelled as open-loop articulated chains with several links. In the PC-based robot application one end of the chain is the end-effector which controls and positions the tool, while the other end is the base of the robot. Kinematics is the modelling of the geometry of motion of the robot arm with respect to a fixed reference coordinate as a function of time without any regard for the forces and moments that cause the motion. Before the PC-based system could be designed and implemented a thorough investigation of the robot kinematics was required. There were two distinct kinematic modes which are dissimilar but related to each other. The first was referred to as the direct or forward kinematic problem while the second was the inverse kinematic solution. The inverse kinematic solution was used more frequently as the task is usually stated in terms of the reference coordinate frame. Figure 7.2 illustrates the relationship between the two kinematic solutions.

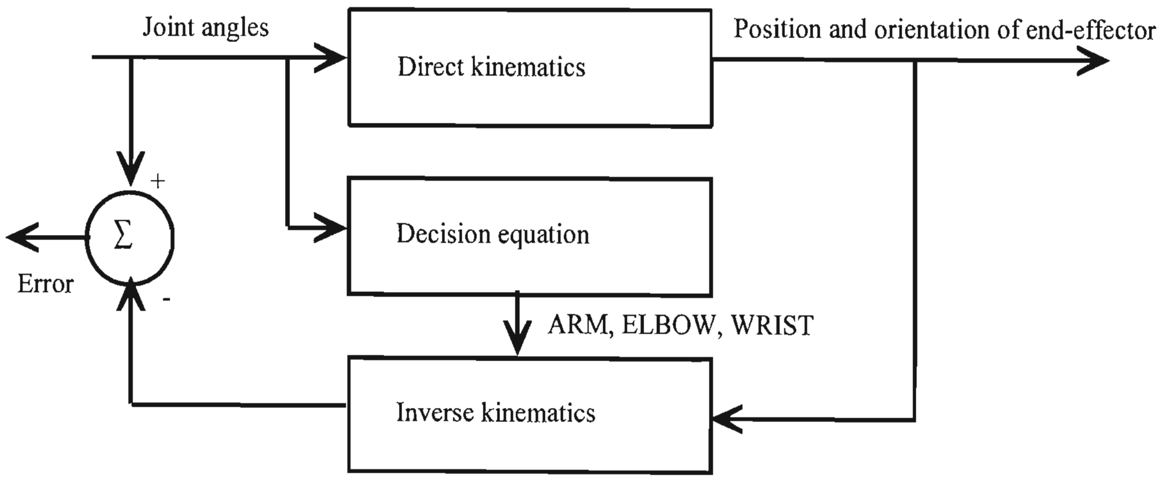


Figure 7.2. The direct and inverse kinematic transfer function solution for the end-effector position and orientation [FU et.al.].

Denavit and Hartenberg proposed a systematic approach that utilised matrix algebra to describe and represent the spatial geometry of the robot links. This method used a 4x4 homogeneous transformation matrix that related the spacial displacement of the hand coordinate frame to the reference coordinate frame. This section summarises the works of [FU et.al.] and reviews the fundamental concepts of robot kinematics.

7.3 THE DIRECT KINEMATIC SOLUTION

The development of a systematic and generalised approach to describe the location of the robot arm links with respect to the fixed reference frame was referred to as the direct kinematic solution. As the PUMA robot is a rotational robot, the direct kinematic solution was reduced to finding a transformation matrix which related the body attached coordinate frames to the reference frame. The rotation matrix was expanded to a 4x4 homogenous transformation matrix to include the transnational operations. The Denavit and Hartenberg method was used to represent a rigid mechanical link as the spacial geometry of the robot.

7.3.1 The Rotation Matrices

The 3x3 rotation matrix was defined as a transformation matrix which operates on a position vector in a three-dimensional space with coordinates expressed in a rotated coordinate system OUVW to the reference coordinate system OXYZ as in Figure 7.3.

Where $\mathbf{I}_3$ was a 3x3 identity matrix. The primary interest in developing the above transformation matrix was to determine the rotation matrices that represented the rotation of the OUVW coordinate system about each of the three principle axes of the reference coordinate system OXYZ. When the OUVW coordinate system was rotated an angle α about the OX axis to arrive at a new location, then the point p_{uvw} having coordinates $(p_u, p_v, p_w)^T$ with respect to the OUVW system would have different coordinates $(p_x, p_y, p_z)^T$ with respect to the reference system OXYZ.

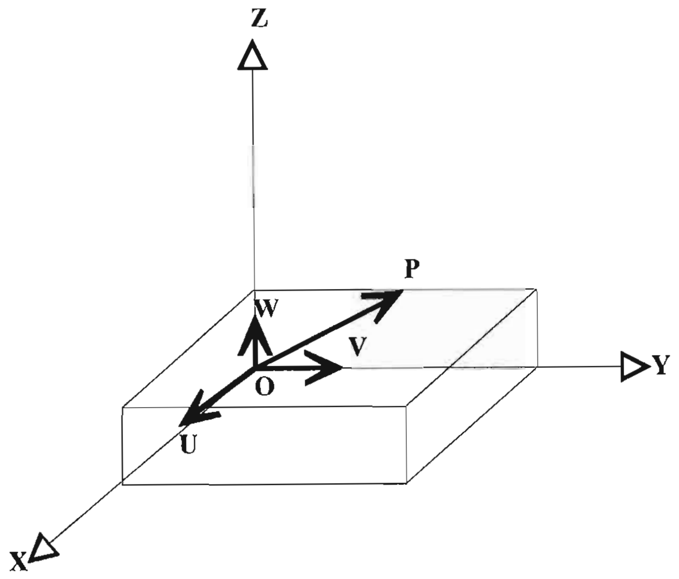


Figure 7.3. The rotation matrix reference and body-attached coordinate system

The transformation matrix $\mathbf{R}_{x,\alpha}$ was referred to as the rotation matrix about the OX axis with angle α. $\mathbf{R}_{x,\alpha}$ was derived from the above transformation matrix, with $i_x \equiv i_u$ and:

$$p_{xyz} = R_{x,\alpha} p_{uvw}$$

$$R_{x,\alpha} = \begin{bmatrix} i_x \cdot i_u & i_x \cdot j_v & i_x \cdot k_w \\ j_y \cdot i_u & j_y \cdot j_v & j_y \cdot k_w \\ k_z \cdot i_u & k_z \cdot j_v & k_z \cdot k_w \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \quad \textbf{(7.10)}$$

Similarly the 3x3 rotation matrix for rotation about the OY axis with angle Φ and about OZ axis with angle θ were respectively represented in Figure 7.4. The matrices ($R_{x,\alpha}$, $R_{y,\Phi}$, $R_{z,\theta}$) were referred to as the basic rotation matrices.

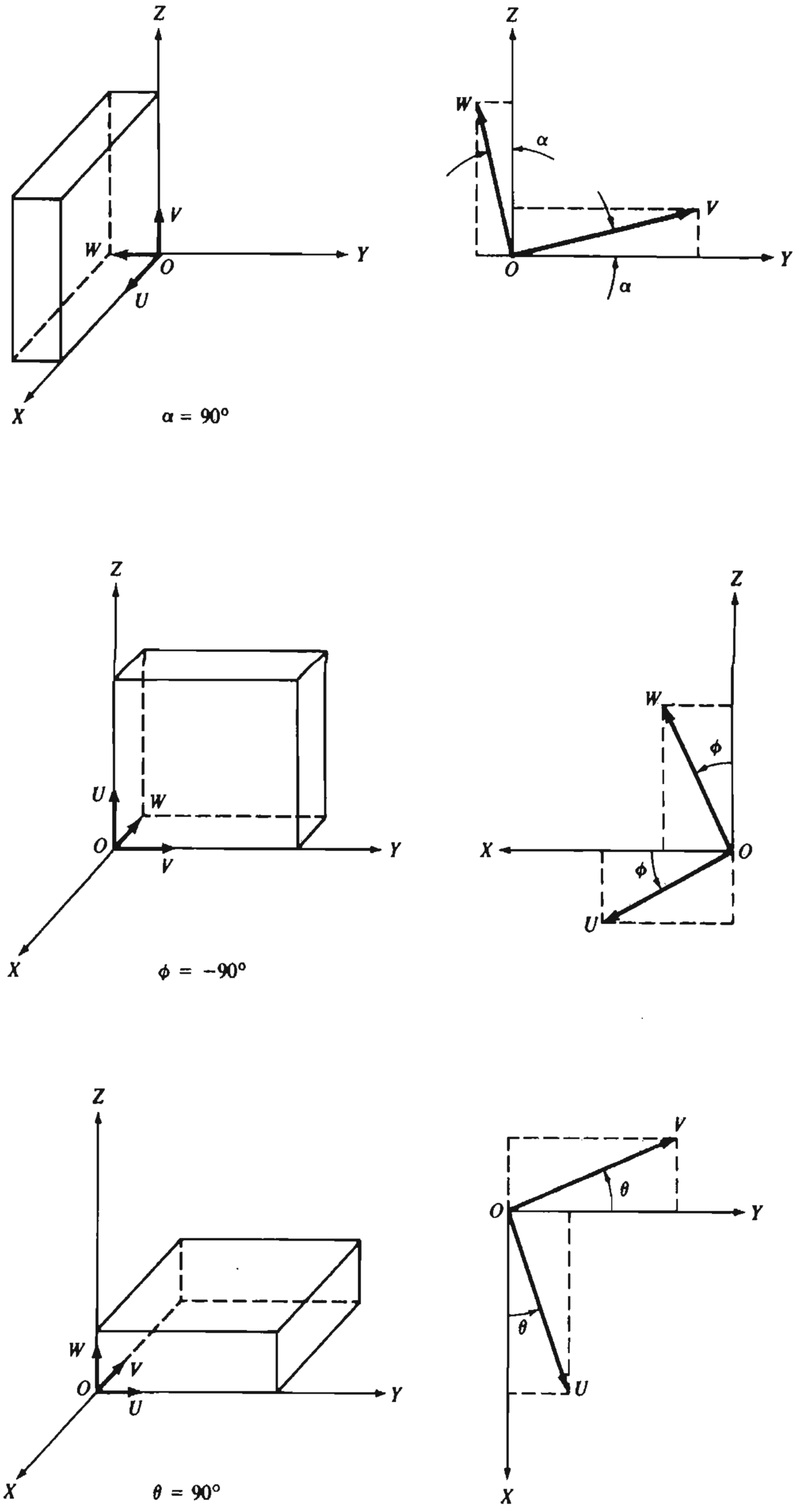


Figure 7.4. The rotating coordinate systems.

7.3.2 The Composite Rotation Matrix

The rotation matrices were multiplied together to represent a sequence of finite rotations about the principle axes of the OXYZ coordinate system. The sequence of multiplication was very important as matrix multiplications do not commute. The rotation of the angle α about the OX axis followed by a rotation angle θ about the OZ axis followed by a rotation of angle Φ about the OY axis, the resultant rotation matrix representing these rotations was:

$$R = R_{y,\phi} R_{z,\theta} R_{x,\alpha} = \begin{bmatrix} C\phi & 0 & S\phi \\ 0 & 1 & 0 \\ -S\phi & 0 & C\phi \end{bmatrix} \begin{bmatrix} C\theta & -S\theta & 0 \\ S\theta & C\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix}$$

$$R = R_{y,\phi} R_{z,\theta} R_{x,\alpha} = \begin{bmatrix} C\phi C\theta & S\phi S\alpha - C\phi S\theta C\alpha & C\phi S\theta S\alpha + S\phi C\alpha \\ S\theta & C\theta C\alpha & -C\theta S\alpha \\ -S\phi C\theta & S\phi S\theta C\alpha + C\phi C\alpha & C\phi C\alpha - S\phi S\theta S\alpha \end{bmatrix} \qquad \textbf{(7.12)}$$

$C\phi \equiv \cos\phi; \quad S\phi \equiv \sin\phi; \quad C\theta \equiv \cos\theta; \quad S\theta \equiv \sin\theta \quad C\alpha \equiv \cos\alpha \quad S\alpha \equiv \sin\alpha$

7.3.3 Link and Joint Parameters

A mechanical manipulator consists of a sequence of rigid bodies, called links. Figure 7.5 illustrates these links for the Unimate PUMA robot.

Each joint link pair constituted 1 degree of freedom. For an N degree of freedom manipulator, there were N joint-link pairs with link 0 attached to a supporting base where an internal coordinate frame was established for the dynamic system, and the last link was attached to a tool. The joints and links were numbered outwardly from the base. Joint 1 was the point of connection between link 1 and the supporting base. Each link was connected to, at most, two others so that no closed loops were formed.

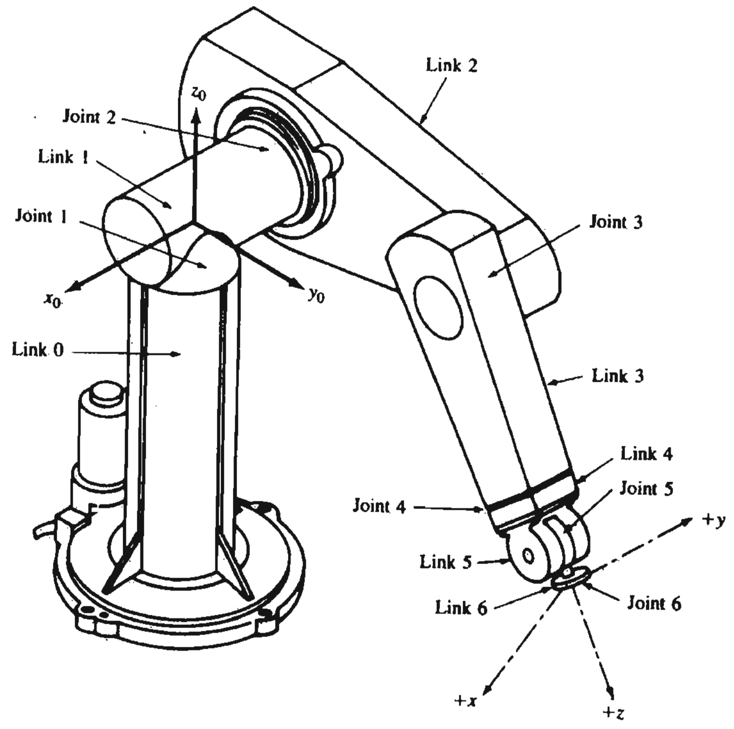


Figure 7.5. The PUMA 560 series robot arm, showing the joints and links.

A joint axis was

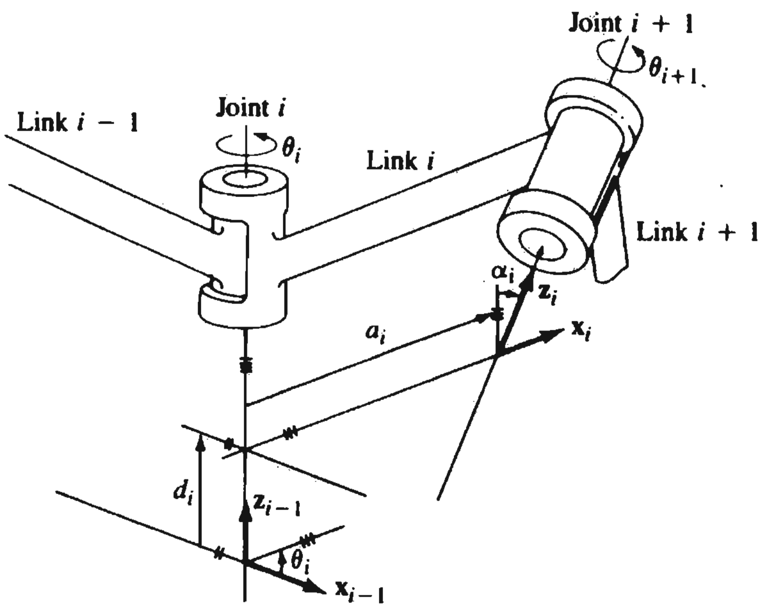


Figure 7.6. The link coordinate system and parameters.

This joint axis had two normals connected to it, one from each of the links. The relative position of two such connected links was given by d_i which was the distance measured along the joint axis between the normals. The joint angle θ_i between the normals was measured in a plane normal to the joint axis. The variables d_i and θ_i were called the distance and the angle between the adjacent links respectively. They determined the relative position of neighbouring links. The significance of the links is, that they maintained a fixed configuration between their joints which could be characterised by two parameters, a_i and α_i. The parameter a_i was the shortest distance measured along the common normal between the joint axes, and α_i was the angle between the joint axis measured in a plane perpendicular to a_i. Thus a_i and α_i were called the length and the twist angle of the link i, respectively. These parameters determined the structure of link i.

7.3.4 The Denavit-Hartenberg Representation

To describe the transnational and rotational relationship between adjacent links, Denavit and Hartenberg proposed a matrix method of systematically establishing a coordinate system to each link of an articulated chain. The Denavit-Hartenberg representation resulted in a 4x4 homogeneous transformation matrix representing each link's coordinate system at the joint with respect to the previous links coordinate system. Figure 7.7 illustrates the axes configuration of the PUMA robot with six degrees of freedom. Table 7.2 displays the PUMA robot arm coordinate parameters.

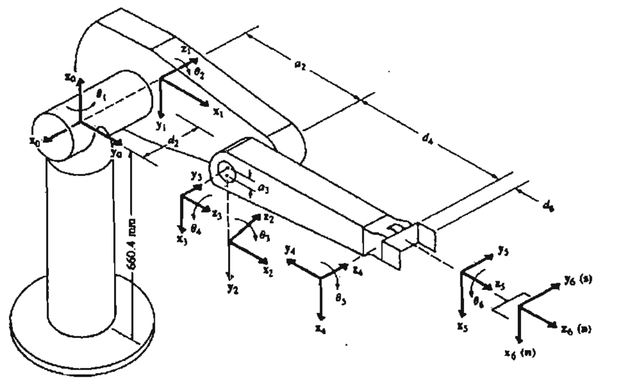


Figure 7.7. The link coordinate system for the PUMA 560 series robot.

PUMA robot arm link parameters					
Joint i	θ_i	α_i	a_i	d_i	Joint range
1	90	-90	0	0	-160 to 160
2	0	0	431.8mm	149.09mm	-225 to 45
3	90	90	20.32	0	-45 to 225
4	0	-90	0	433.07mm	-110 to170
5	0	90	0	0	-100 to 100
6	0	0	0	56.25mm	-266 to 266

Table 7.2. The link coordinate and parameter table of the PUMA 560 robot.

7.4 THE INVERSE KINEMATIC SOLUTION

In order to control the position and orientation of the end-effector of the six degree of freedom robot, the inverse kinematic solution was more important. In other words, given the position and orientation of the end-effector of a six-axis robot arm as ${}^{0}T_6$ and its joint and link parameters, the corresponding joint angles $q=(q_1,q_2,q_3,q_4,q_5,q_6)^T$ of the robot had to be calculated, so that the end-effector could be positioned as desired.

7.4.1 The Geometric Approach

This section presents the geometric approach to solving the inverse kinematic problem of the PUMA robot. Similar as to human anatomy the robot was classified into a mechanical manipulator having an ARM, ELBOW and WRIST. For the six-axis PUMA robot there was four possible solutions for the first three joints and two for the last three joints. These solutions were dependant on various arm configurations and their definitions. Definitions and various arm configurations for the PUMA robot are shown in Figure 7.8. The configuration definitions used for the PC-based PUMA robot were based on the configuration in Figure 7.9.

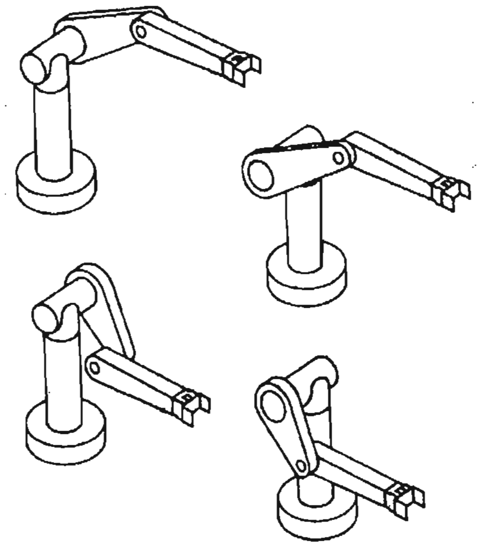


Figure 7.8. The various arm definitions and configurations of the PUMA 500 series robot.

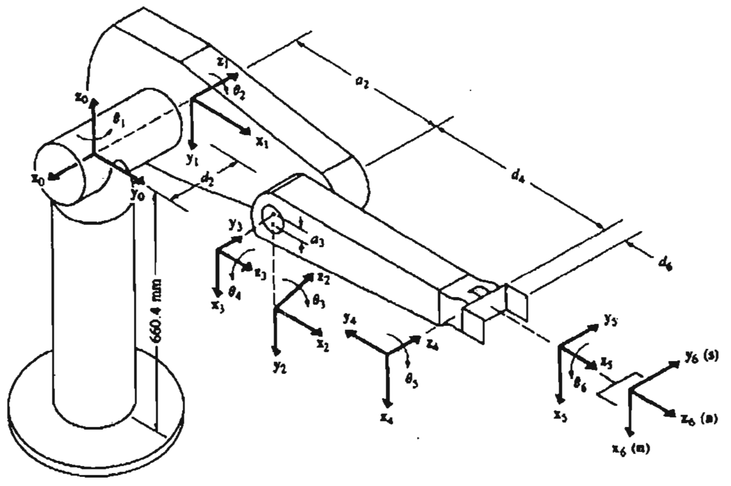


Figure 7.9. The configuration definition used for the PC-based PUMA 560 series robot.

- RIGHT (shoulder) ARM: Positive θ_2 moved the wrist in the positive z_0 direction while joint 3 was not activated.
- LEFT (shoulder) ARM: Positive θ_2 moved the wrist in the negative z_0 direction while joint 3 was not activated.
- ABOVE ARM (elbow above wrist): Position of wrist of the RIGHT/LEFT arm with respect to the shoulder coordinate had a NEGATIVE/POSITIVE coordinate value along the y_2 axis.
- BELOW ARM (elbow below wrist): Position of wrist of the RIGHT/LEFT arm with respect to the shoulder coordinate had a POSITIVE/NEGATIVE coordinate value along the y_2 axis.
- WRIST DOWN: The s unit vector of the hand coordinate system and the y_5 unit vector of the (x_5,y_5,z_5) coordinate system had a positive dot product.
- WRIST UP: The s unit vector of the hand coordinate system and the y_5 unit vector of the (x_5,y_5,z_5) coordinate system had a negative dot product.

With respect of the above definitions of the arm configurations, two arm configuration indicators were defined for each arm configuration. The indicators were combined to give one possible solution for the four arm configurations as in Figure 7.9.

These indicators were defined as:

$$ARM = \begin{Bmatrix} +1 & RIGHT & ARM \\ -1 & LEFT & ARM \end{Bmatrix}$$

$$ELBOW = \begin{Bmatrix} +1 & ABOVE & ARM \\ -1 & BELOW & ARM \end{Bmatrix}$$

$$WRIST = \begin{Bmatrix} +1 & WRIST & DOWN \\ -1 & WRIST & UP \end{Bmatrix}$$

The inverse kinematic arm solutions for the respective joints were defined using this simple geometric approach.

JOINT 1 solution: If the vector **p** was on the x_0,y_0 plane, a solution for θ_1 was calculated.

$$\theta_1^L = \phi - \alpha$$
$$\theta_1^R = \pi + \phi + \alpha$$
$$r = AB = \sqrt{p_x^2 + p_y^2 - d_2^2}$$
$$R = OB = \sqrt{p_x^2 + p_y^2}$$
$$OA = d_2$$

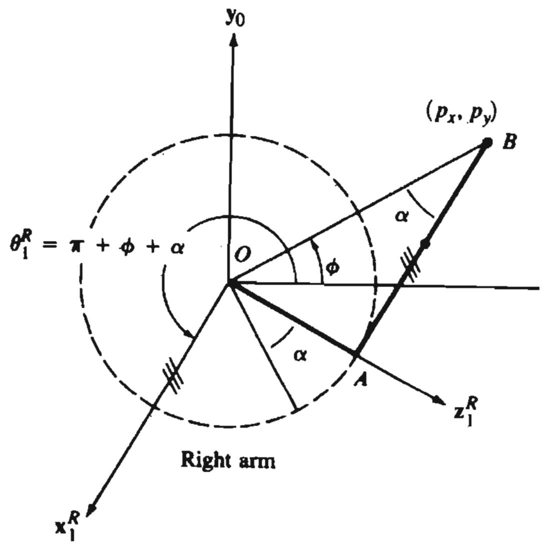


Figure 7.10. The geometric solution of joint 1.

$$\theta_1 = \tan^{-1}\left[\frac{-ARMp_y\sqrt{p_x^2 + p_y^2 - d_2^2} - p_x d_2}{-ARMp_x\sqrt{p_x^2 + p_y^2 - d_2^2} + p_y d_2}\right]$$

For the right arm PUMA robot ARM=+1

$$\therefore \theta_1 = \tan^{-1}\left[\frac{-p_y\sqrt{p_x^2 + p_y^2 - d_2^2} - p_x d_2}{-p_x\sqrt{p_x^2 + p_y^2 - d_2^2} + p_y d_2}\right]$$

$$-\pi \le \theta_1 \le \pi \qquad \textbf{(7.17)}$$

JOINT 2 solution: For joint 2 project vector **p** on to the x_1,y_1 plane. For the RIGHT ARM ABOVE the WRIST, $\theta_2=\alpha+\beta$.

ARM=+1 and ELBOW=+1

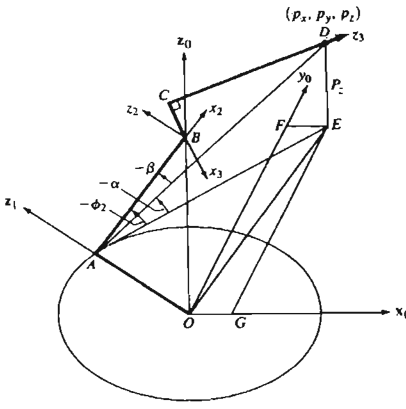


Figure 7.11. The geometric solution of joint 2.

$$OA = d_2 \quad AB = a_2 \quad BC = a_3 \quad EF = p_x$$

$$EG = p_y \quad DE = p_z \quad CD = d_4$$

$$AD = R = \sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2} \quad AE = r = \sqrt{p_x^2 + p_y^2 - d_2^2}$$

$$\sin\alpha = -\frac{p_z}{R} = -\frac{p_z}{\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}}$$

$$\cos\alpha = -\frac{ARM \cdot r}{R} = -\frac{1 \cdot \sqrt{p_x^2 + p_y^2 - d_2^2}}{\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}} \tag{7.18}$$

$$\cos\beta = \frac{a_2^2 + R^2 - (d_4^2 + a_3^2)}{2a_2 R} \quad \sin\beta = \sqrt{1 - \cos^2\beta}$$

$$\therefore \sin\theta_2 = \sin\alpha\cos\beta + (ARM \cdot ELBOW)\cos\alpha\sin\beta$$

$$\cos\theta_2 = \sin\alpha\cos\beta - (ARM \cdot ELBOW)\sin\alpha\sin\beta$$

$$\theta_2 = \tan^{-1}\left[\frac{\sin\theta_2}{\cos\theta_2}\right] \quad -\pi \le \theta_2 \le \pi$$

JOINT 3 solution: For joint 3 the position vector **p** was projected onto the x_2,y_2 plane.

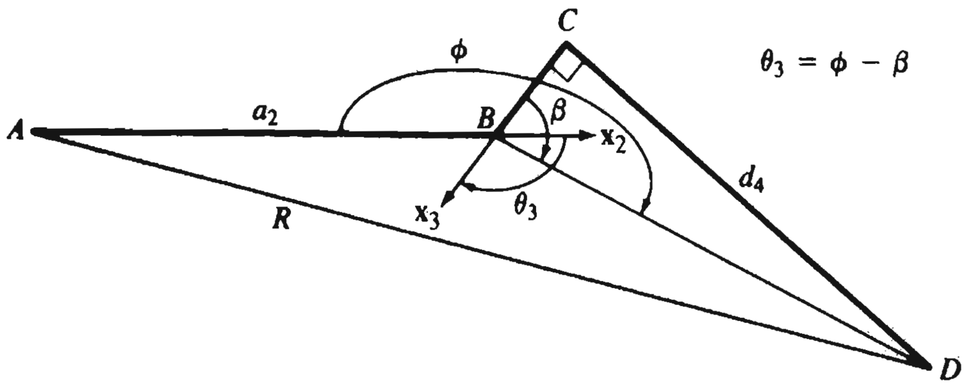


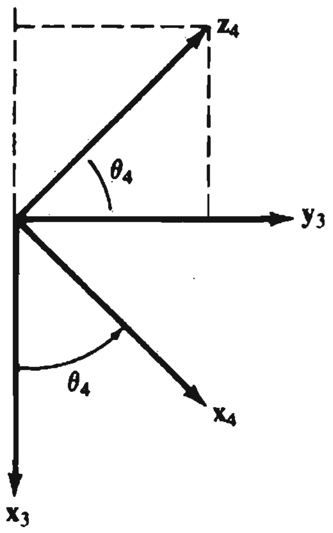
Figure 7.12. The geometric solution of joint 3.

$$AB = a_2 \quad BC = a_3 \quad CD = d_4 \quad BD = \sqrt{a_3^2 + d_4^2}$$
$$AD = R = \sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}$$
$$\theta_3 = \phi - \beta$$
$$\sin\theta_3 = \sin(\phi - \beta) = \sin\phi\cos\beta - \cos\phi\sin\beta \tag{7.19}$$
$$\cos\theta_3 = \cos(\phi - \beta) = \cos\phi\cos\beta + \sin\phi\sin\beta$$
$$\theta_3 = \left[\frac{\sin\phi_3}{\cos\phi_3}\right] \quad -\pi \leq \theta_3 \leq \pi$$

Knowing the first three joint angles the evaluation for the ${}^0T_3$ matrix was used extensively to find the solution for the last three joints. The solution of the last three joints of the PUMA robot was calculated by setting these joints to meet the following criteria:

1. Set joint 4 such that a rotation about joint 5 would align the axis of motion of joint 6 with the given approach vector (**a** of **T**).
2. Set joint 5 to align the axis of motion of joint 6 with the approach vector.
3. Set joint 6 to align the given orientation vector and normal vector.

Mechanically the criteria related to: **JOINT 4**



$$Z_4 = \frac{\pm (z_3 \times a)}{\|z_3 \times a\|} \qquad a = (a_x, a_y, a_z)^T$$

$$a = z_5 \quad s = y_5 \quad s = (s_x, s_y, s_z)^T \tag{7.20}$$

$$\sin\theta_4 = -(z_4 \cdot x_3)$$

$$\cos\theta_4 = z_4 \cdot y_3$$

Figure 7.13. The geometric solution of joint 4

$$\theta_4 = \tan^{-1}\left[\frac{C_1 a_y - S_1 a_x}{C_1 C_{23} a_x + S_1 C_{23} a_y - S_{23} a_z}\right] \qquad -\pi \le \theta_4 \le \pi \tag{7.21}$$

JOINT 5 solution: To find θ_5, using the criterion that aligned the axis of rotation of joint 6 with respect to the approach vector:

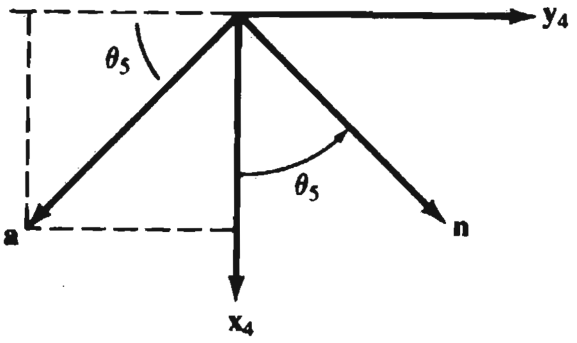


Figure 7.14. The geometric solution of joint 5.

$\sin\theta_5 = a \cdot x_4 \qquad \cos\theta_5 = -(a \cdot y_4)$

Where x_4 and y_4 were the x and y column vectors of ${}^0T_4$ respectively and **a** was the approach vector.

$$\theta_5 = \tan^{-1}\left[\frac{(C_1C_{23}C_4 - S_1S_4)a_x + (S_1C_{23}C_4 + C_1S_4)a_y - C_4S_{23}a_y}{C_1S_{23}a_x + S_1S_{23}a_y + C_{23}a_z}\right] \qquad \textbf{(7.22)}$$

$-\pi \leq \theta_4 \leq \pi$

JOINT 6 solution: The gripper had to be aligned with the object to ease picking up of the object. Projecting the hand coordinate onto the (x_5,y_5) plane.

$\sin\theta_6 = n \cdot y_5 \qquad \cos\theta_6 = s \cdot y_5$

Where y_5 was the y column vector of ${}^0T_5$, **n** and **s** are the normal and sliding vectors of ${}^0T_6$ respectively.

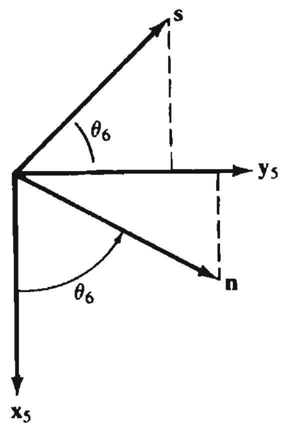


Figure 7.15. The geometric solution of joint 6.

$$\theta_6 = \tan^{-1}\left[\frac{(-S_1C_4 - C_1C_{23}S_4)n_x + (C_1C_4 - S_1C_{23}S_4)n_y + (S_4S_{23})n_z}{(-S_1C_4 - C_1C_{23}S_4)s_x + (C_1C_4 - S_1C_{23}S_4)s_y + (S_4S_{23})s_z}\right] \quad \textbf{(7.23)}$$

$$-\pi \le \theta_6 \le \pi$$

Deriving the equations for the inverse kinematic solution, solutions for the joint orientations $(\theta_1, \theta_2, \theta_3, \theta_4, \theta_5, \theta_6)$ were calculated. Comparing the forward and inverse kinematic solutions to an orientation, allowed for the accurate position control of the PUMA robot. The difference between the calculated forward kinematic solution and the comparative inverse kinematic solution was known as the actuation position error as in Figure 7.16.

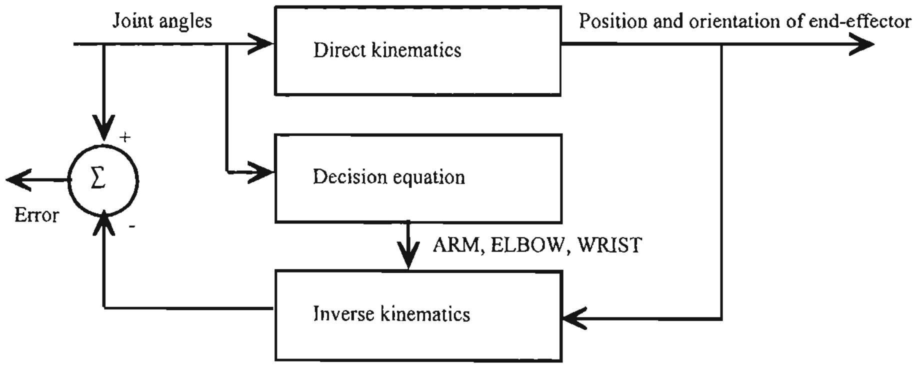


Figure 7.16. Illustrates how the difference between the calculated forward kinematic solution and the comparative inverse kinematic solution produced the actuation position error.

7.5 DYNAMIC ROBOT CONTROL

Determining the kinematic solution of the PC-based PUMA robot required large calculations and processing speeds. The forward and inverse kinematic solutions had to be compared and error adjustments had to be made. The error determined the mechanical actuation required for the final arm position during which the P controller provided the dynamic control of the robot arm. This section details the derivation of the transfer function of single joint of the PUMA robot, from which a proportional controller was obtained. The servo control cards used for the robot control were equipped with P controllers. The control strategy was initially developed for a single joint of the PC-based robot and later applied to all six robot joints.

7.6 THE PC-BASED PUMA 560 SERIES ROBOT

For the PUMA 560 series robot arm, each joint was treated as a simple servomechanism, each with a 40V permanent magnet DC servomotor equipped with an electromagnetic brake (ref. Chapter 3). Each DC servomotor was monitored by a bidirectional encoder for incremental position feedback and a 5V potentiometer for absolute position feedback during startup (ref. Chapter 4). The robot was interconnected to the PC-based controller by a signal harness and a power harness. The signal harness was responsible for all the low power signals, while the power harness relayed all the high power signals between the controller and the robot arm.

The control structure was hierarchically arranged, with the Internet host controller at the top of the hierarchy. For the modified modular mechatronic PC-based robot, the controller consisted of a PII 500MHz computer, which served as the primary robot controller computer, while the Internet control formed part of the supervisory control. At the lower level were the two Advantech® PCL-832 3-axis servomotor control cards and one Eagle® PC-30GA interface card. The PII 500MHz computer performed two major functions:

1. On-line user interaction and sub-task scheduling from the user, graphical user interface (GUI) commands.
2. Sub-task coordination with the servo control and interface cards.

Figure 7.17 shows the modular mechatronic hierarchical control structure of the PC-based PUMA 560 series robot.

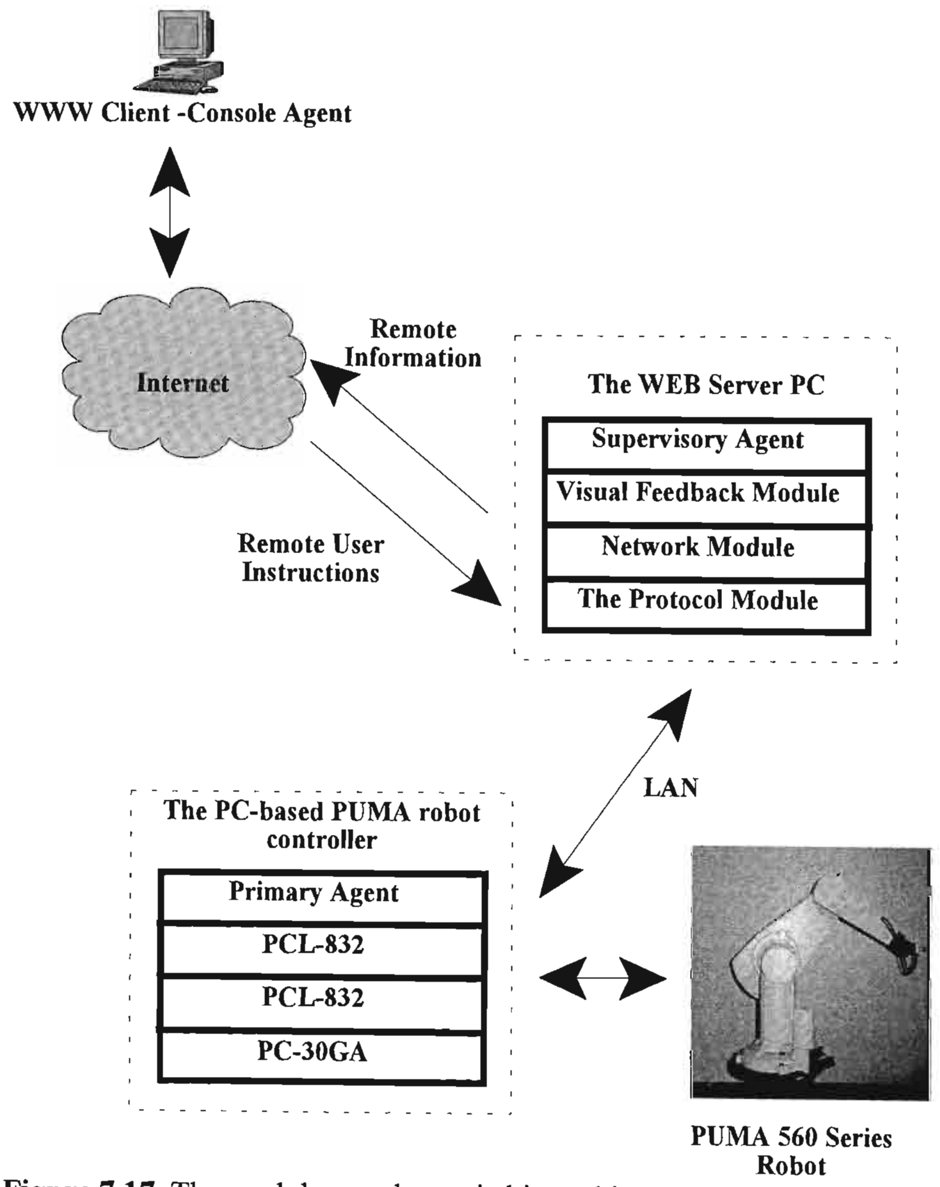


Figure 7.17. The modular mechatronic hierarchical control structure of the PC-based PUMA 560 series robot.

The on-line user interactions included passing, interpreting, and decoding the GUI commands, in addition to reporting appropriate error messages to the user. Once the GUI command had been decoded, various internal routines were called to perform scheduling and coordination functions. These functions included:

1. Coordinate system transformations.
2. Joint-interpolated position updates to correspond to each set point to each joint every 25ms (programmable DDA cycle time).
3. Acknowledging from each servo control card that each axis of motion has completed its required incremental position.

The PC-based robot control scheme was a proportional control method (P controller). It did not have the capability of updating the feedback gains under varying payloads. This is an area for further research. Since an industrial robot was a highly nonlinear system, the inertial loading, the coupling between joints and the gravity effects were all either position-dependant or position- and velocity-dependant terms.

7.7 THE ROBOT / COMPUTER INTERFACE

The ability of standard desktop computers to interact with an environment was dependent on the presence of some form of interfacing hardware [Demas]. The function of the hardware was to obtain information from the environment with which the computer was interfacing and then present this information to the central processing unit (CPU) of the controlling computer. The PC-based control system adopted some fundamental interfacing principles to ensure the effective interaction of the system with its environment. The transducers collected the information from the environment. Modular mechatronic signal conditioning systems were used to produce signals the PC-based controller could process to make control decisions. The signal conditioned feedback information was passed as arguments to software coded decision algorithms that generated the appropriate activation signals to effect system changes that were required to maintain system stability.

The computer central processing unit interpreted the feedback information and generated activation signals that controlled the system actuation circuitry. The actuation signals were analogue and digital in nature.

7.7.1 Control Using the PC-30GA Card

The PC-30GA series card was a low cost, high accuracy analogue and digital I/O card for the IBM PC. The PC-30GA card had a throughput of 100 kHz, featuring four digital-to-analogue converters (DAC). The card had 16 single ended or 8 differential analogue-to-digital inputs, with programmable gains of 1,10,100 and 1000. The card also had three digital I/O ports with 8 lines per port. Figure 7.18 shows the PC-30GA card.

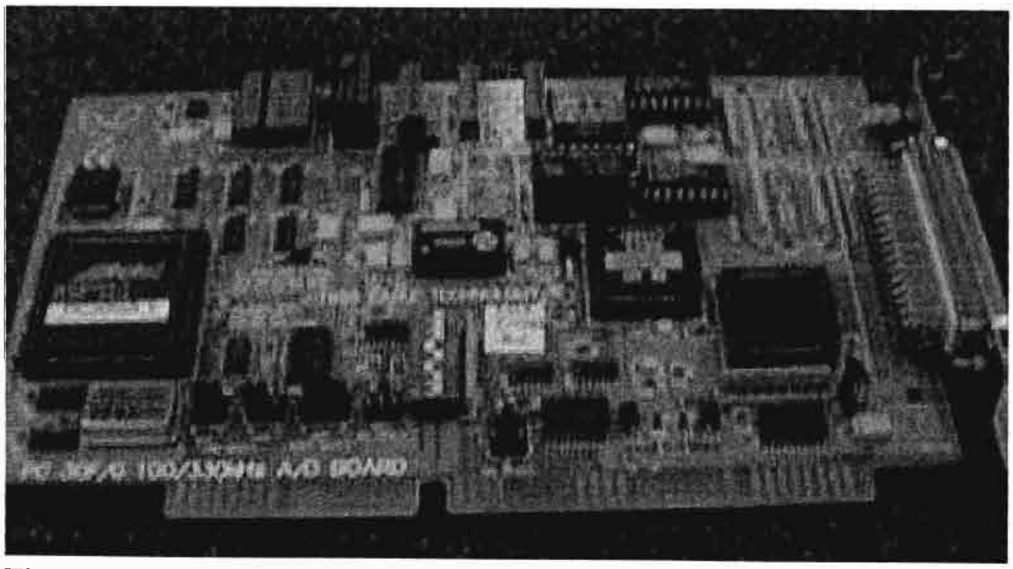


Figure 7.18. The Eagle PC-30GA interface card.

The 3 digital ports could each be set as either input or output ports. Ports that were configured as input ports reflected the digital inputs on the port when read. Ports that were configured as outputs were set and latched to the state most recently written to the port. The bit definitions for the three ports are given in Table 7.3.

PORT	BIT DEFINITIONS							
	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
A	A7	A6	A5	A4	A3	A2	A1	A0
B	B7	B6	B5	B4	B3	B2	B1	B0
C	C7	C6	C5	C4	C3	C2	C1	C0

Table 7.3. Bit definitions for the three 8-bit ports. The data refers to the line number of the port.

During the control of the PC-based PUMA robot the PC-30GA card was used to:

1. Receive the feedback information, for the angular position of each joint, from the potentiometers during startup.
2. Release the electromagnetic brakes of the PUMA robot.

The PC-based PUMA robot implemented the simple analogue-to-digital (AD) ports of the PC-30GA card, to sample the angular position states of each joint. Six AD ports were used, one for each joint of the robot. Each joint returned a voltage between 0.5V and 5V corresponding to its angular position. Each joints angular position to voltage ratios was different due to the gearing ratios of each joint. During software coding each joint was calibrated as to afford the appropriate voltage to an angular position ratio. Figure 7.19, 7.20, 7.21 and 7.22 show the internal gearing mechanisms of the PUMA 560 series robot.

The PC-30GA interface card was also used to release the electromagnetic brakes of the PUMA robot. All six the robot brakes were released at the same time as the brakes were all interconnected. The brakes were released, from software, just before the robot operation commenced.

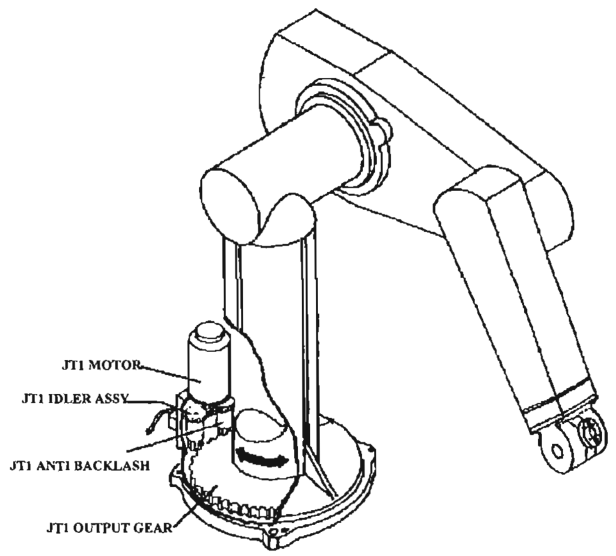


Figure 7.19. The gear mechanism of joint 1.

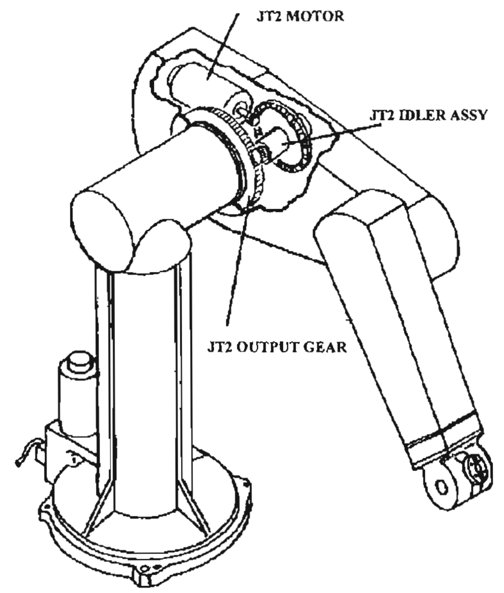


Figure 7.20. The gear mechanism of joint 2.

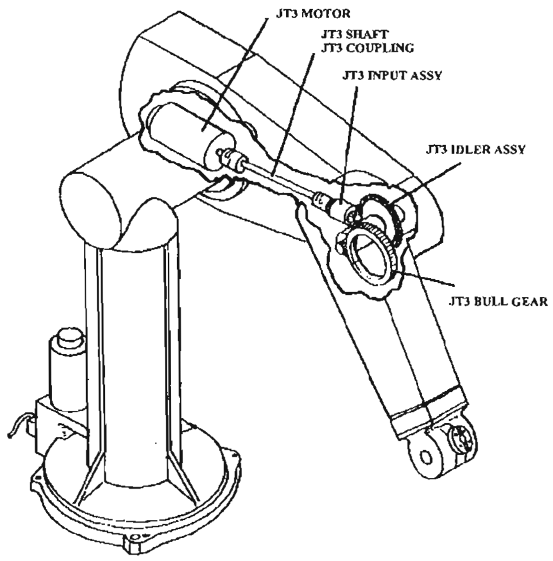


Figure 7.21. The gear mechanism of joint 3

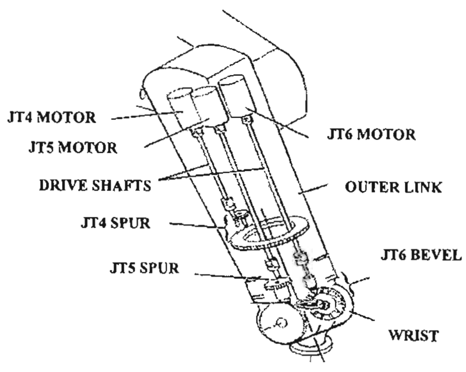


Figure 7.22. The gear mechanism of joint 4,5, and 6.

The brake release circuitry used a transistor-relay circuit that allowed a low power TTL control input signal to a high power relay to energise the electromagnetic brakes from the external power supply. The relay solenoid was connected to a collector-emitter circuit of a low cost, low power bipolar NPN-transistor. The TTL signals were applied to the base of the transistor, setting the voltage bias of the base. When the PC-based control signal was a digital high 2.8V dc $< V_b <$ 5V dc the base was biassed such that the current flowed in the collector-emitter circuit, energising the relay coil. Similarly if the control signal was a logic low 2.1V dc $> V_b >$ 0.8V dc the biassing of the base would not allow current to flow through the collector-emitter circuit, not allowing the coil to be energised. The switching state of the relay was controlled by setting the appropriate control signal TTL levels to either high or low. Figure 7.23 shows the robot brake release relay circuit diagram. During the robot shutdown procedure the brakes are activated, preventing the robot from collapsing under its own load.

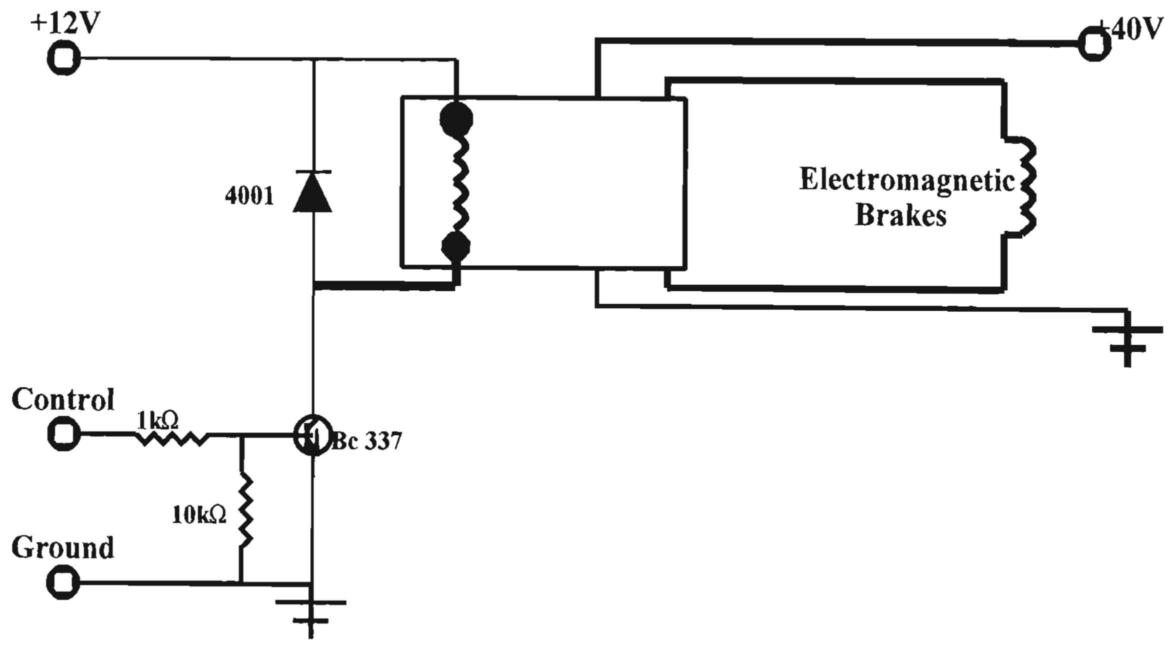


Figure 7.23. The circuit diagram for the brake release control module. The control signal was applied to the base of the transistor, which energised the relay coil with the additional low power 12V DC supply.

Microsoft® Visual Basic 6 (VB6) was used to communicate with the PC-30GA card, which was controlled via a library of high level functions stored in the dynamic link library (dll), *edr32.dll.* This dll was supported by the Eagle EDR Software Developers Kit (SDK).

The PC-30GA interface card formed one part of the PC-based robot control strategy. The PC-30GA card was responsible for two aspects of the control strategy. As the card was equipped with a variety of control functions, it was ideal to use not only for the PC-based robot control, but also as a *modular mechatronic interface* device. The decision logic of the PC-30GA interface card with respect to the PC-based robot control is shown in Figure 7.24.

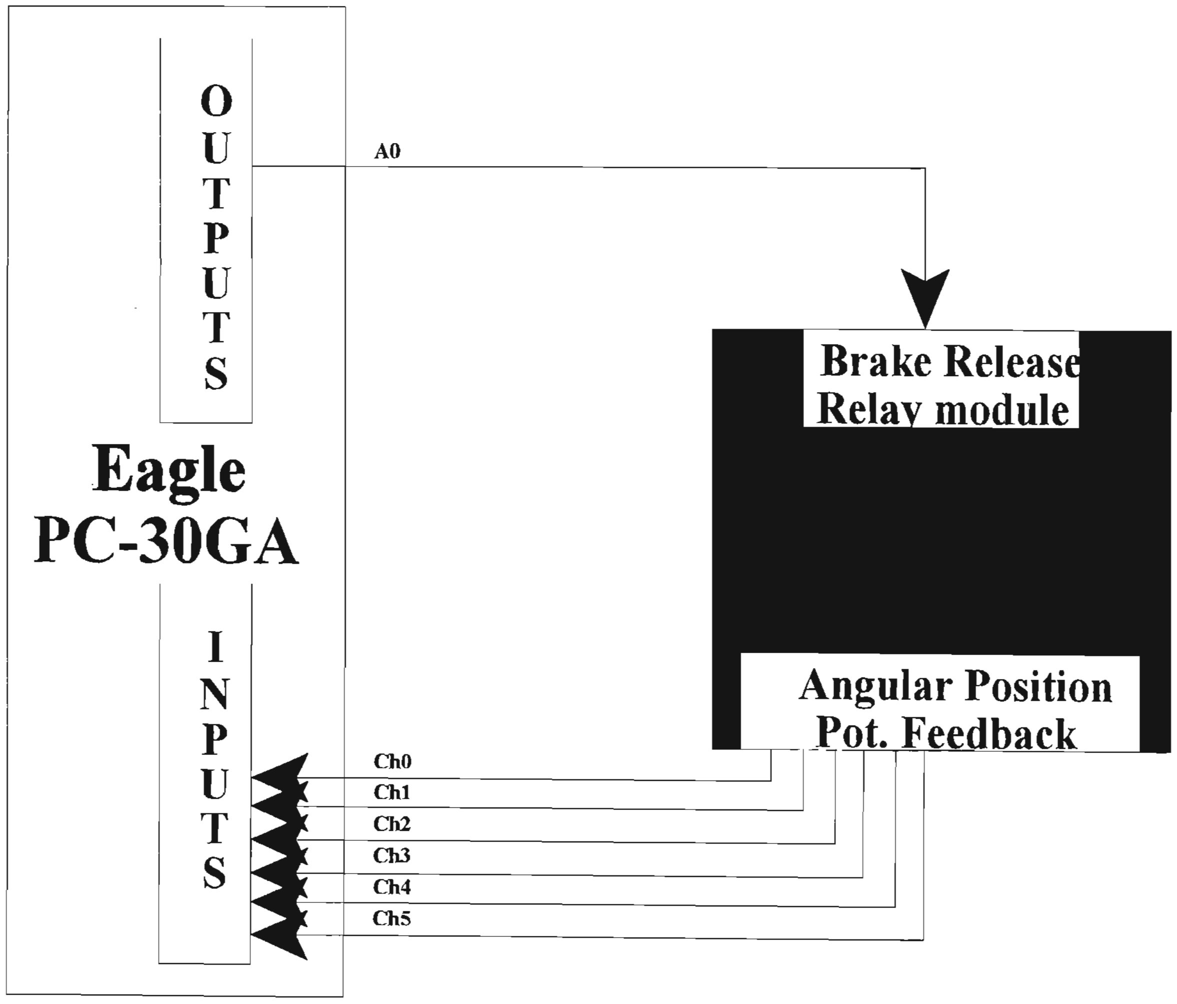


Figure 7.24. The decision logic of the PC-30GA interface card.

The angular position feedback information was sampled via Ch0-Ch5. The brake release relay module was controlled from the digital line A0.

7.7.2 The PCL-832 Servo Control Card

The PCL-832 3-axis servomotor control card was used as a sophisticated position controller. The card's implementation provided high performance at an affordable price. The PCL-832 used digital differential analysis techniques to implement position control. Each axis had its own position control chip, allowing completely independent control of up to three servo motors.

The PCL-832's programming library supported high-level commands and functions, making control easy. The library included commands to set the DDA cycle time and acceleration/deceleration curve as well as functions for linear interpolation, circular interpolation, return home and jog. Each PCL-832 3-axis servomotor control card was capable of independent 3-axis servo control, affording control to the six independent robot joints. The cards had fully continues closed-loop P offset controllers, with a differential position encoder interface. Each card had three independent 12-bit digital-to-analogue (DAC) outputs with ±10V ranges. Figure 7.25 shows the Advantech® PCL-832 3-axis servomotor control card.

Microsoft® Visual Basic 6 (VB6) was used to communicate with the PCL-832 card, which was controlled via a library of high level functions stored in the dynamic link library (dll), *PCL832.dll.*

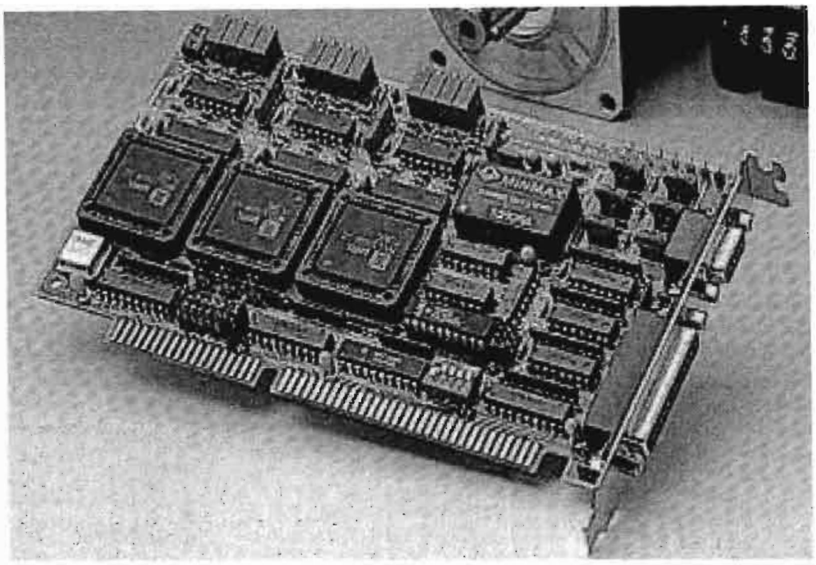


Figure 7.25. The PCL-832 3-axis servomotor control card.

The PCL-832 servo control card was used as the servo motor control module of the PC-based PUMA robot. Two cards were required for the overall robot control. Each robot joint had a bidirectional encoder, which was connected to the *encoder signal conditioning modules* (ref. Chapter 4) via the robot signal harness. The *encoder signal conditioning modules* were interfaced to the PCL-832 servo control cards. The PCL-832 servo control signals from the PC, were interfaced to the *LM12 buffer circuit module* (ref. Chapter 4). The *LM12 buffer circuit module* was connected to the *LM12 drive module* (ref. Chapter 3), which produced the high power servomotor control voltages. The PCL-832 card had two connectors. Table 7.4 shows the line definitions of the PCL-832 servo control card used. The decision logic of the PCL-832 interface cards for the PC-based robot control is shown in Figure 7.26.

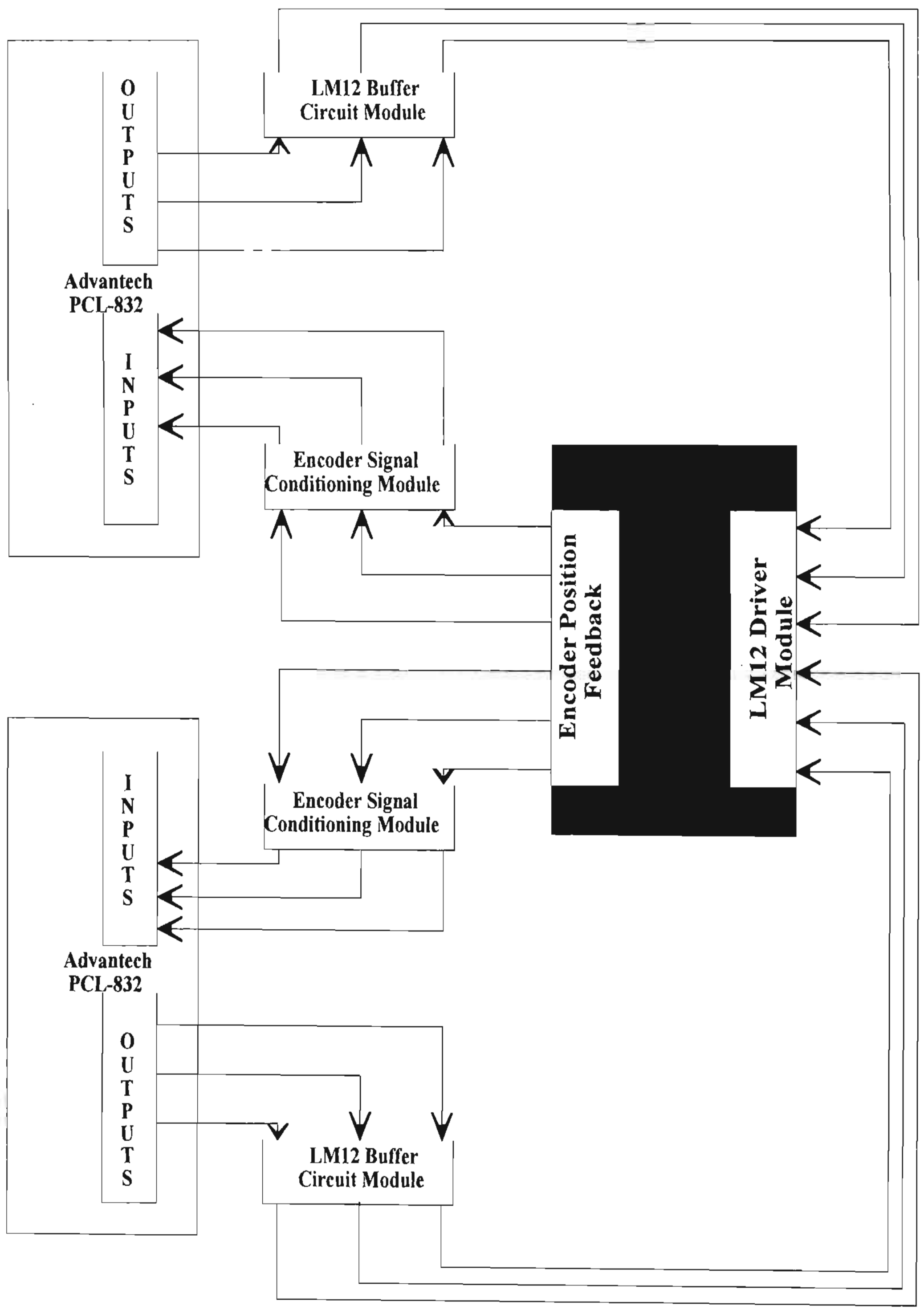

OUTPUTS
Advantech PCL-832
INPUTS
LM12 Buffer Circuit Module
Encoder Signal Conditioning Module
Encoder Position Feedback
LM12 Driver Module
Encoder Signal Conditioning Module
INPUTS
Advantech PCL-832
OUTPUTS
LM12 Buffer Circuit Module

Figure 7.26. The decision logic of the PCL-832 interface cards for the PC-based PUMA robot.

I/O	**LINE**			**FUNCTION**
INPUT - PCL-832 (1)				
CN2	CH1 Ain+	CH1Bin+	CH1 INDEX	Joint 1
	CH2 Ain+	CH2 Bin+	CH2 INDEX	Joint 2
	CH3 Ain+	CH3 Bin+	CH3 INDEX	Joint 3
	DGND			Digital Ground
INPUT - PCL-832 (2)				
CN2	CH1 Ain+	CH1 Bin+	CH1 INDEX	Joint 4
	CH2 Ain+	CH2 Bin+	CH2 INDEX	Joint 5
	CH3 Ain+	CH3 Bin+	CH3 INDEX	Joint 6
	DGND			Digital Ground
OUTPUT - PCL-832 (1)				
CN1	CH1VCMD			Joint 1
	CH2VCMD			Joint 2
	CH3VCMD			Joint 3
	AGND			Analogue Ground
OUTPUT - PCL-832 (2)				
CN1	CH1VCMD			Joint 4
	CH2VCMD			Joint 5
	CH3VCMD			Joint 6
	AGND			Analogue Ground

Table 7.4. The line definitions of the PCL-832 servo control cards.

The decision logic and hardware system layout of the PC-based PUMA controller is shown in Figure 7.27.

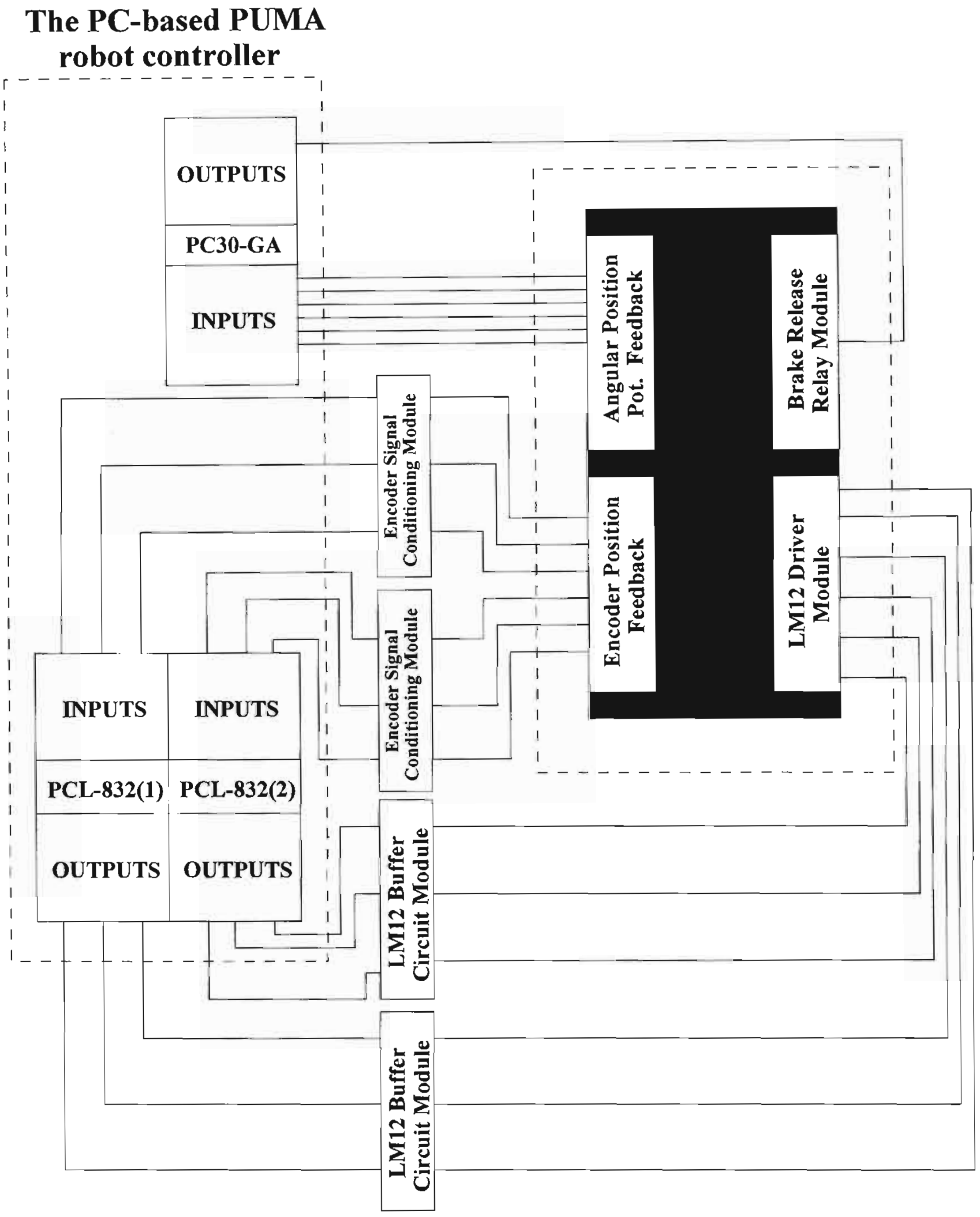

The PC-based PUMA robot controller
OUTPUTS
PC30-GA
INPUTS
Angular Position Pot. Feedback
Brake Release Relay Module
Encoder Signal Conditioning Module
Encoder Position Feedback
LM12 Driver Module
Encoder Signal Conditioning Module
INPUTS
INPUTS
PCL-832(1)
PCL-832(2)
OUTPUTS
OUTPUTS
LM12 Buffer Circuit Module
LM12 Buffer Circuit Module

Figure 7.27. The decision logic and control modules of the PC-based PUMA controller.

7.8 ROBOT CONTROL SYSTEM

The final control system for the PUMA robot was successfully developed as a PC-based system that implemented digital and analogue interfacing techniques to control and coordinate the functioning of the PUMA robot arm. The system was implemented using two control strategies. The PUMA robot arm was controlled either by a remote supervisory agent or locally, using the primary PC-based controller.

In keeping with the modular mechatronics design technique, the final control system of the PUMA robot was developed by integrating the previously established control modules for each of the motion systems. The control software of the PC-based PUMA robot included the sequencing algorithms of the two control strategies. Each control strategy allowed for either the robot joints to be controlled individually or the calling of a part manufacturing process from memory. The operation of the PUMA robot control software package, *RobotClient.vbp*, is reviewed in more detail in Chapter 9.

Incorporating a PC-based control package allowed for online robot manipulation and programming, eliminating outdated programming techniques which was downtime associated. As the need for dedicated controllers are eliminated, a variety of robot systems can be upgraded by simply changing geometric and power requirements. The modification in the control and software dramatically enhanced the accuracy, flexibility and capabilities of the industrial robot.

CHAPTER 8

8 PERFORMANCE ANALYSIS OF THE INTERNET-BASED CIM SYSTEM

The functioning of the Internet-based CIM cell is dependant on the integrity of the networked devices, and the efficiency of the communications system. This chapter presents the network performance and data transmission results of the Internet-based control system.

The efficient supervisory control of the CIM components is dependant upon the network performance and connectivity. The performance testing of each CIM Client with respect to the Server has been reviewed separately, this was required as each CIM Client communicates with the Server using different control commands. The rate of network transmission is dependant on the length of the control word transmitted over the network. When transmitting data across an open network the following communication requirements exist:

- The network must have low latency.
- The network must have a high throughput.
- Any signals being transmitted over the network must be authenticated. The receiver must know who sent the information.
- The transmission of the data must be secure. The sender must be aware of who receives the data.
- The network must be fault-resilient.

The overall time optimised operation of the Internet control system is largely dependant on the speed of data transmission and data integrity.

8.1 MEASURING NETWORK PERFORMANCE

The performance of the Network is dependant on the transmission rate of the Network adapters used by each CIM Client controller and the CIM Server. The laboratory standard, of the Network adapters, used in the MR^{2}G CIM cell is 10 Mbps (Mega bits per second). The rating of the Network adapter is an indication of the operational limits of the adapter. The adapter is able to receive data at 10Mbps, but cannot transmit at the same rate. The continuous collisions on the network prevents this from happening. Bits per second is a measure of the number of data bits (0 or 1) transmitted each second in a communication channel. This is commonly referred to as bit rate. Individual characters (letters, numbers, etc.), also referred to as bytes, are composed of several bits. The speed at which the Intra-connected machines communicate is also known as the *line speed* of the network.

The speed at which data is transmitted across the Internet is dependant upon the bandwidth of the transmitting device, either a modem or ISDN (Integrated Services Digital Network) line used for Internet transmission. The bandwidth expresses the number of discrete signals in one second. Note that both bandwidth and bps (bits per second) refer to the rate at which the bits within a single frame are transmitted. The gaps between the frames can be variable length.

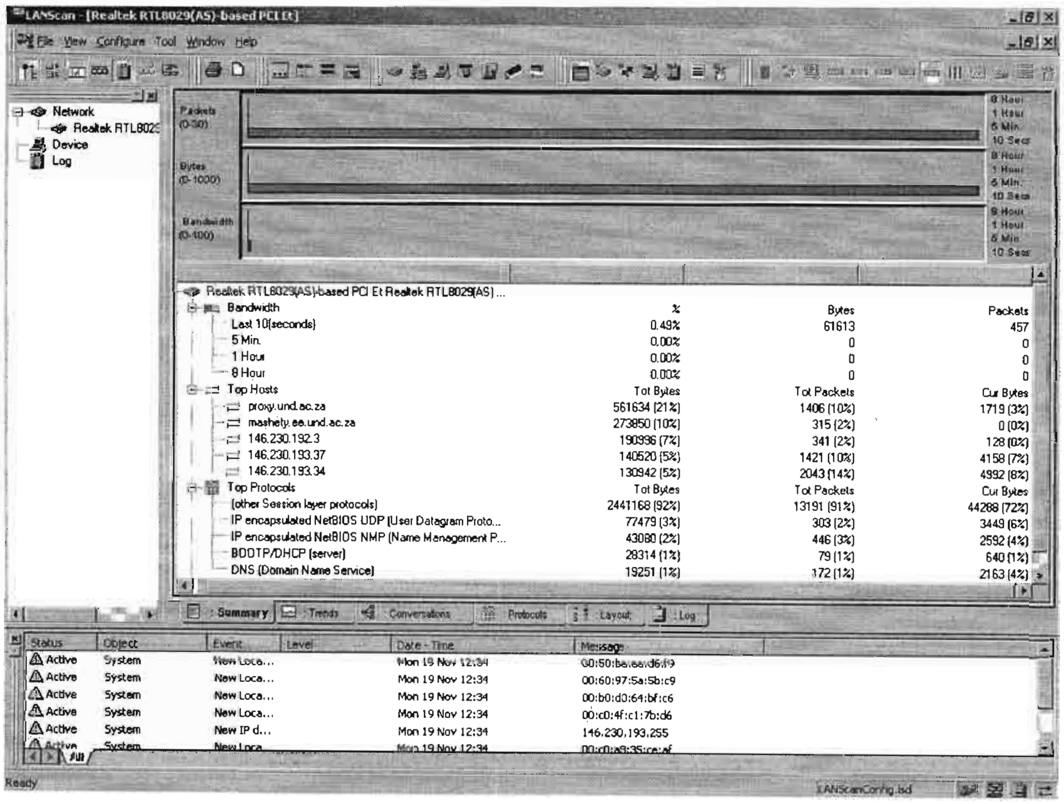
For the performance analysis of the CIM Internet control system, the performance of the network must be monitored and measured. A network analyser software package, called LANScan, was used to monitor and measure Network performance. LANScan is a powerful network monitoring and diagnostic tool, used to gain valuable information regarding network performance, utilisation and faults.

LANScan applications depending on the network activity include:

- Monitoring router performance.
- Measuring network bandwidth utilisation.
- Reporting Internet access.
- Identifying applications that use network resources.

- Locating faulty components, like broken cables, bad interface cards or failed network ports.
- Identifying software problems, including incorrectly configured network drivers, duplicate or missing network addresses or unexpected activity.
- Measuring network responses for local and wide area applications.
- Record long term network activity.
- Remote network monitoring.

Figure 8.1 shows the LANScan graphical user interfaced (GUI), launched on the CIM Server computer.



Realtek RTL8029(AS)-based PCI Et Realtek RTL8029(AS) ...

Bandwidth	%	Bytes	Packets
Last 10(seconds)	0.49%	61613	457
5 Min.	0.00%	0	0
1 Hour	0.00%	0	0
8 Hour	0.00%	0	0
Top Hosts	Tot Bytes	Tot Packets	Cur Bytes
proxy.und.ac.za	561634 (21%)	1406 (10%)	1719 (3%)
mashely.ee.und.ac.za	273850 (10%)	315 (2%)	0 (0%)
146.230.192.3	190996 (7%)	341 (2%)	128 (0%)
146.230.193.37	140520 (5%)	1421 (10%)	4158 (7%)
146.230.193.34	130942 (5%)	2043 (14%)	4992 (8%)
Top Protocols	Tot Bytes	Tot Packets	Cur Bytes
(other Session layer protocols)	2441168 (92%)	13191 (91%)	44288 (72%)
IP encapsulated NetBIOS UDP (User Datagram Proto...	77479 (3%)	303 (2%)	3449 (6%)
IP encapsulated NetBIOS NMP (Name Management P...	43080 (2%)	446 (3%)	2592 (4%)
BOOTP/DHCP (server)	28314 (1%)	79 (1%)	640 (1%)
DNS (Domain Name Service)	19251 (1%)	172 (1%)	2163 (4%)

Status	Object	Event	Level	Date - Time	Message
Active	System	New Loca...		Mon 19 Nov 12:34	00:50:ba:ea:d6:f9
Active	System	New Loca...		Mon 19 Nov 12:34	00:60:97:5a:5b:c9
Active	System	New Loca...		Mon 19 Nov 12:34	00:b0:d0:64:bf:c6
Active	System	New Loca...		Mon 19 Nov 12:34	00:c0:4f:c1:7b:d6
Active	System	New IP d...		Mon 19 Nov 12:34	146.230.193.255
Active	System	New Loca...		Mon 19 Nov 12:34	00:c0:a8:35:ce:af

Figure 8.1. The LANScan graphical user interface (GUI).

Referring to Figure 8.1, the upper left corner of the GUI form displays the Network adapter used. The Network adapter for the CIM Server is a Realtek RTL8029 PCI Network card. The centre of the form displays the key parameters of the network. A tree displays real time information about network bandwidth, top hosts, top protocols and events.

8.2 NETWORK PERFORMANCE MEASUREMENTS

The performance of computer network is also dependant on the packet size of the information being transmitted, and the network congestion at a specific time of the day. Table 8.1 shows the network bandwidth available for data transmission at different times of the day. During the measuring of these values the CIM Client controllers were logged in to the CIM Host controller. The averaged network performance was sampled over a period of three weeks. The Network performance values were obtained using LANScan. Figure 8.2 shows a graph of the network traffic performance for different times of the day.

Reading #	**Time (CAT)**	**Bandwidth (Bytes/Second)**
1	08:00	73982
2	08:30	31273
3	09:00	28359
4	09:30	17362
5	10:00	10229
6	10:30	9872
7	11:00	12312
8	11:30	11099
9	12:00	8977
10	12:30	7306
11	13:00	4182
12	13:30	4774
13	14:00	5637
14	14:30	8648
15	15:00	10224
16	15:30	11239
17	16:00	13098
18	16:30	43897
19	17:00	65209

Table 8.1. The performance of the Network during the day.

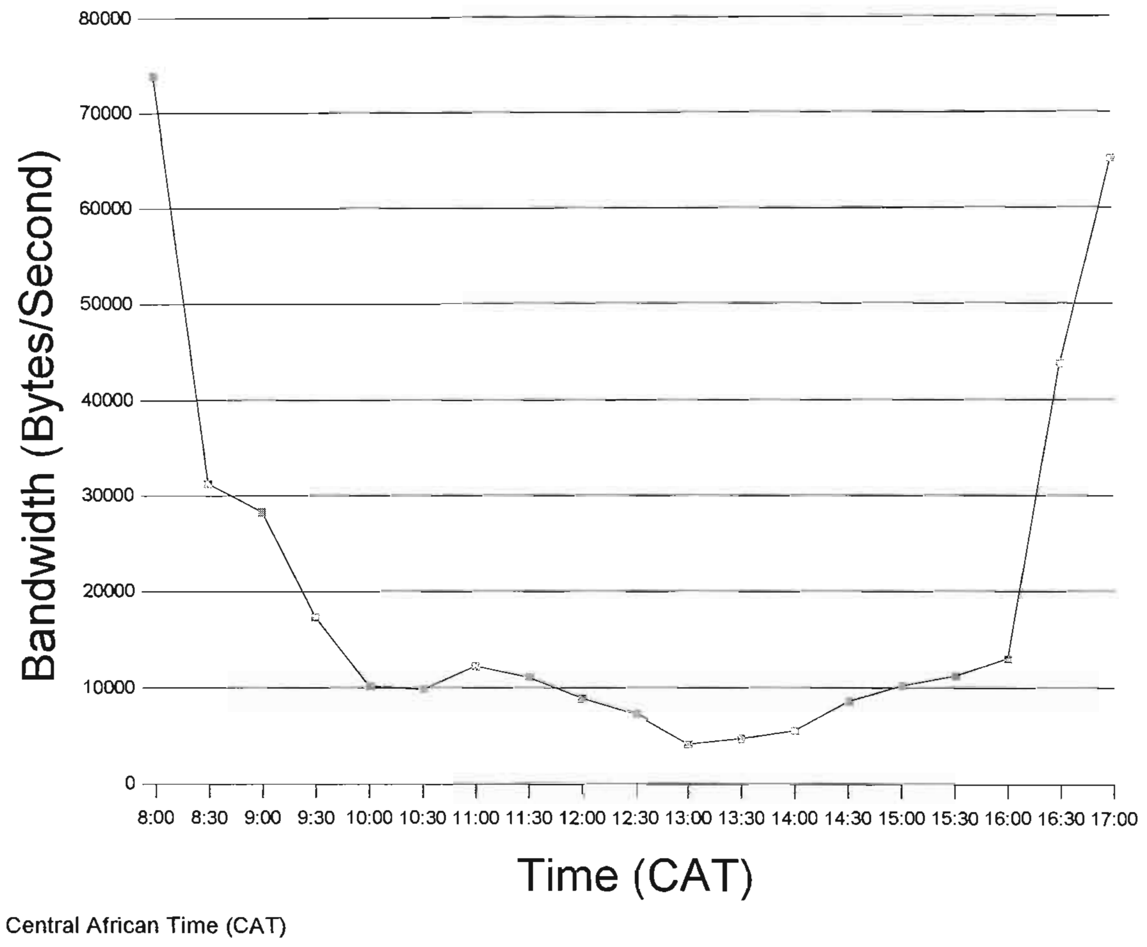


Figure 8.2. Graph of the network traffic performance for an average day on the University of Natal Network.

Figure 8.2 shows how the network activity changes during the day. The significant change in Network bandwidth affects the performance of the CIM Internet control system. Using equation 8.1, the data transmission speed across the Network can be determined.

$$\frac{Bytes}{Bandwidth} = Time \tag{8.1}$$

eg. If the CIM Host controller transmits a 26 Byte packet to the Robot Client at 15:30 (CAT) then the transmission time would be:

$$\frac{26Bytes}{11239Bytes/s} = \underline{2.313x10^{-3}s}$$

The 26 Byte packet reaches the Robot Client 2.313 milliseconds after transmission from the CIM Host controller. Chapter 9 describes the operational review of the Modular Mechatronic CIM Internet control system, including the tabulated commands for Client/Server interactions.

8.3 CIM COMMUNICATION PERFORMANCE

This section displays the performance of the data transfer between the CIM Host controller and each Client controller. These calculations are based on the Network bandwidth results described in Table 8.1. Each data transmission event across the network has an 18 Byte header and trailer attached to the data packet. The header and trailer simply identifies the beginning and the end of a transmitted data packet, including information about the origin of the information.

8.3.1 The Camera Selector Client

Table 8.2 shows the command interaction between the CIM Host controller and the Camera Selector Client. The time transmission performance of the commands are listed Table 8.3.

Command Number	CIM Host → Camera Selector Client	Camera Selector Client → CIM Host
1	ClientCAM-CAMERA1	
2	ClientCAM-CAMERA2	
3	ClientCAM-CAMERA3	
4	ClientCAM-CAMERA4	
5	ClientCAM-EXIT	
6		HostCAM-CAMERA1
7		HostCAM-CAMERA2
8		HostCAM-CAMERA3
9		HostCAM-CAMERA4
10		HostCAM-EXIT

Table 8.2. The command interactions between the CIM Host and the Camera Selector Client.

TIME (CAT)	CIM Host → Camera Selector Client					Camera Selector Client → CIM Host				
	Commands					Commands				
	1	2	3	4	5	6	7	8	9	10
	Number of Bytes (Including header and trailer)					Number of Bytes (Including header and trailer)				
	35	35	35	35	32	33	33	33	33	30
	Transmission Time (ms)					Transmission Time (ms)				
8:00	0.473	0.473	0.473	0.473	0.432	0.446	0.446	0.446	0.446	0.405
8:30	1.119	1.119	1.119	1.119	1.023	1.055	1.055	1.055	1.055	0.959
9:00	1.234	1.234	1.234	1.234	1.128	1.164	1.164	1.164	1.164	1.058
9:30	2.016	2.016	2.016	2.016	1.843	1.900	1.900	1.900	1.900	1.728
10:00	3.422	3.422	3.422	3.422	3.128	3.226	3.226	3.226	3.226	2.933
10:30	3.545	3.545	3.545	3.545	3.241	3.343	3.343	3.343	3.343	3.039
11:00	2.843	2.843	2.843	2.843	2.599	2.680	2.680	2.680	2.680	2.437
11:30	3.153	3.153	3.153	3.153	2.883	2.973	2.973	2.973	2.973	2.703
12:00	3.899	3.899	3.899	3.899	3.565	3.676	3.676	3.676	3.676	3.342
12:30	4.791	4.791	4.791	4.791	4.379	4.517	4.517	4.517	4.517	4.106
13:00	8.369	8.369	8.369	8.369	7.651	7.891	7.891	7.891	7.891	7.174
13:30	7.331	7.331	7.331	7.331	6.703	6.912	6.912	6.912	6.912	6.284
14:00	6.208	6.208	6.208	6.208	5.677	5.854	5.854	5.854	5.854	5.322
14:30	4.047	4.047	4.047	4.047	3.700	3.816	3.816	3.816	3.816	3.469
15:00	3.423	3.423	3.423	3.423	3.130	3.228	3.228	3.228	3.228	2.934
15:30	3.114	3.114	3.114	3.114	2.847	2.936	2.936	2.936	2.936	2.669
16:00	2.672	2.672	2.672	2.672	2.443	2.519	2.519	2.519	2.519	2.290
16:30	0.797	0.797	0.797	0.797	0.729	0.752	0.752	0.752	0.752	0.683
17:00	0.537	0.537	0.537	0.537	0.490	0.506	0.506	0.506	0.506	0.460

Table 8.3. The transmission time performance between the CIM Host controller and the Camera Selector Client in milliseconds.

The transmission time performance for each command between the CIM Host and Camera Selector Client is plotted in Figure 8.3.

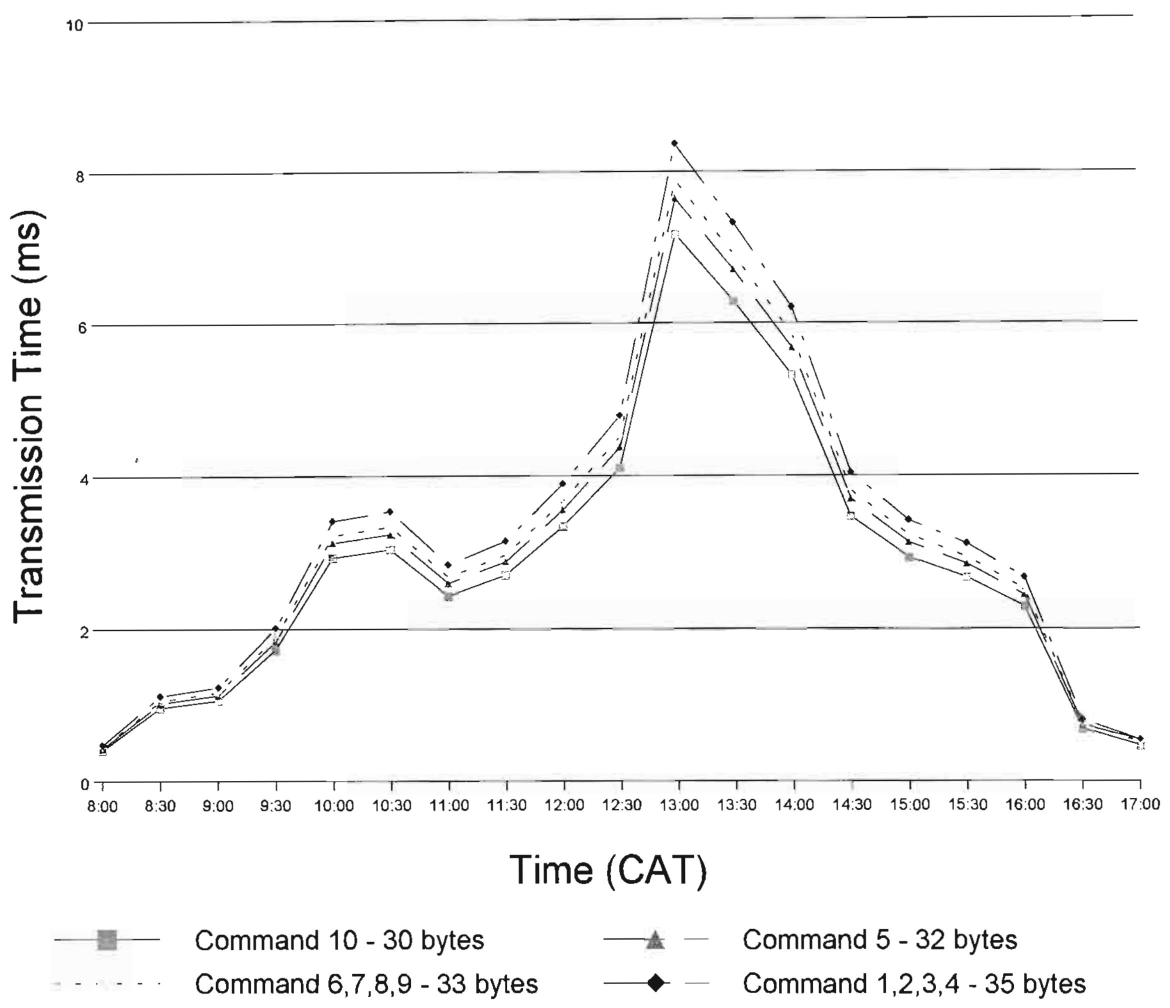


Figure 8.3. Graph of the transmission times between the CIM Host and the Camera Selector Client.

From Figure 8.3. the maximum transmission time occurs as predicted during the times when the network bandwidth is the lowest. The longest transmission time interval is 8.369 milliseconds, for a 35 byte data packet at 13:00 hours (CAT). The Camera Selector Client receives the command to select a camera from the CIM Host in less than 9 milliseconds from transmission. During off-peak times, 17:00 hours (CAT), this value drops below 600 microseconds.

8.3.2 The Conveyer Client

Table 8.4 shows the command interaction between the CIM Host controller and the Conveyer Client. The time transmission performance of the commands are listed Table 8.5.

Command Number	CIM Host - Conveyer Client	Conveyerr Client - CIM Host
1	ClientCON-START	
2	ClientCON-AGVAVIS	
3	ClientCON-AVISAGV	
4	ClientCON-AGVROBOT	
5	ClientCON-ROBOTAGV	
6	ClientCON-AVISROBOT	
7	ClientCON-ROBOTAVIS	
8	ClientCON-EXIT	
9		HostCON-AGV
10		HostCON-ROBOT
11		HostCON-AVIS
12		HostCON-RUN
13		HostCON-EXIT

Table 8.4. The command interactions between the CIM Host and the Conveyer Client.

The transmission time performance for each command between the CIM Host and Camera Selector Client is plotted in Figure 8.4.

TIME (CAT)	CIM Host - Conveyer Client Commands								Conveyer Client - CIM Host Commands				
	1	2	3	4	5	6	7	8	9	10	11	12	13
	Number of Bytes (Including header and trailer)								Number of Bytes(Including header and trailer)				
	33	35	35	36	36	37	37	32	29	31	30	29	30
	Transmission Time (ms)								Transmission Time (ms)				
8:00	0.446	0.473	0.473	0.486	0.486	0.500	0.500	0.432	0.392	0.419	0.405	0.392	0.405
8:30	1.055	1.119	1.119	1.151	1.151	1.183	1.183	1.023	0.927	0.991	0.959	0.927	0.959
9:00	1.163	1.234	1.234	1.269	1.269	1.304	1.304	1.128	1.022	1.093	1.058	1.022	1.058
9:30	1.901	2.016	2.016	2.073	2.073	2.131	2.131	1.843	1.670	1.786	1.728	1.670	1.728
10:00	3.226	3.422	3.422	3.519	3.519	3.617	3.617	3.128	2.835	3.03	2.933	2.835	2.933
10:30	3.342	3.545	3.545	3.647	3.647	3.748	3.748	3.241	2.938	3.140	3.039	2.938	3.039
11:00	2.680	2.843	2.843	2.923	2.923	3.005	3.005	2.599	2.355	2.518	2.437	2.355	2.437
11:30	2.973	3.153	3.153	3.243	3.243	3.333	3.333	2.883	2.613	2.793	2.703	2.613	2.703
12:00	3.676	3.899	3.899	4.010	4.010	4.122	4.122	3.565	3.230	3.453	3.342	3.230	3.342
12:30	4.517	4.791	4.791	4.927	4.927	5.064	5.064	4.379	3.969	4.243	4.106	3.969	4.106
13:00	7.891	8.369	8.369	8.608	8.608	8.847	8.847	7.651	6.934	7.413	7.174	6.934	7.174
13:30	6.912	7.331	7.331	7.540	7.540	7.75	7.75	6.703	6.075	6.493	6.284	6.075	6.284
14:00	5.854	6.208	6.208	6.386	6.386	6.563	6.563	5.677	5.144	5.499	5.322	5.144	5.322
14:30	3.816	4.047	4.047	4.163	4.163	4.278	4.278	3.700	3.353	3.585	3.469	3.353	3.469
15:00	3.228	3.423	3.423	3.521	3.521	3.619	3.619	3.130	2.836	3.032	2.934	2.836	2.934
15:30	2.936	3.114	3.114	3.203	3.203	3.292	3.292	2.847	2.580	2.759	2.669	2.580	2.669
16:00	2.519	2.672	2.672	2.748	2.748	2.825	2.825	2.443	2.214	2.367	2.290	2.214	2.290
16:30	0.752	0.797	0.797	0.820	0.820	0.843	0.843	0.729	0.661	0.706	0.683	0.661	0.683
17:00	0.506	0.537	0.537	0.552	0.552	0.567	0.567	0.490	0.445	0.475	0.460	0.445	0.460

Table 8.5. The transmission time performance between the CIM Host controller and the Conveyer Client in milliseconds.

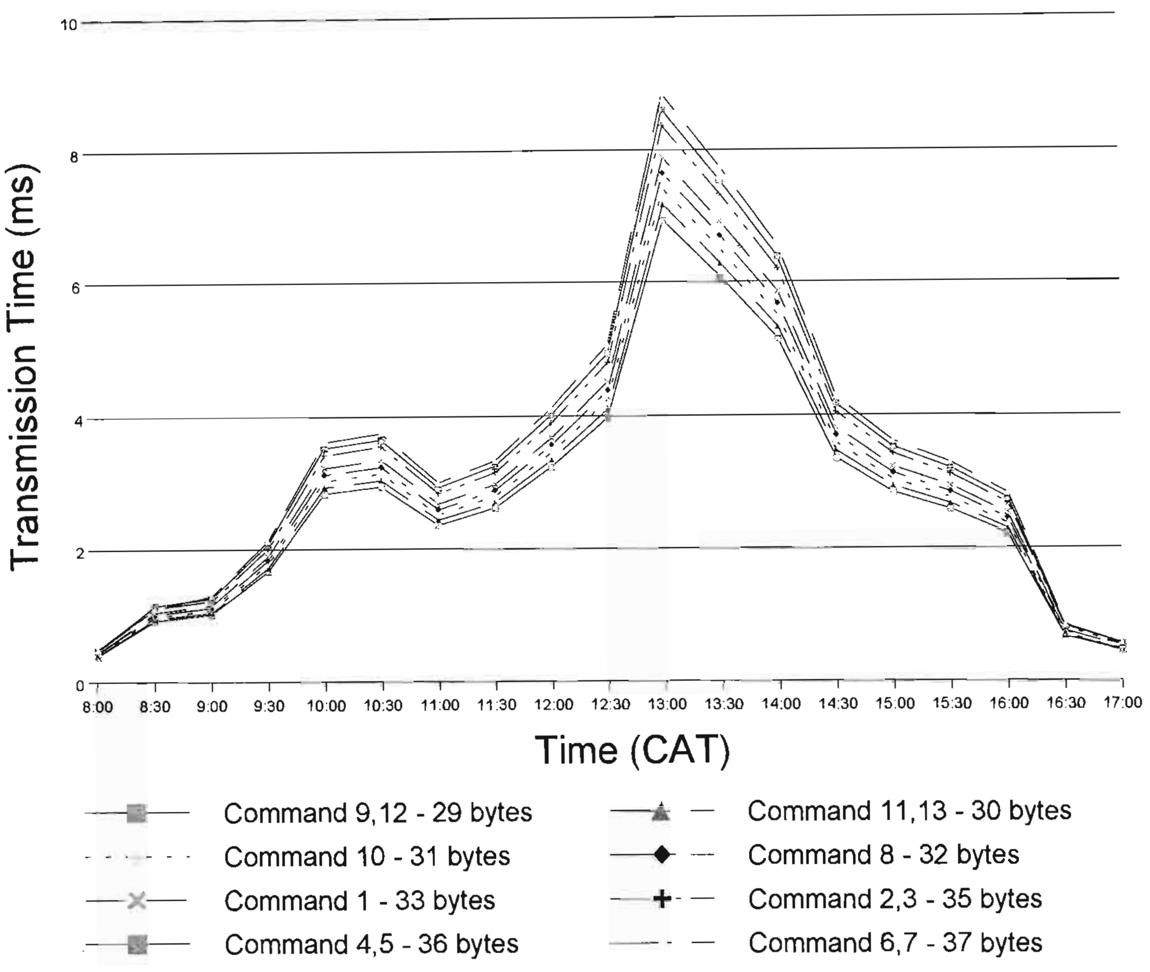


Figure 8.4. Graph of the transmission times between the CIM Host and the Conveyer Client.

From Figure 8.4. the longest transmission time interval is 8.847 milliseconds, for a 37 byte data packet at 13:00 hours (CAT). The Conveyer Client receives the command to start a material handling routine from the CIM Host in less than 9 milliseconds from transmission. During off-peak times, 17:00 hours (CAT), this value drops below 600 microseconds. Figure 8.4 and Figure 8.3 show the same characteristic graphical trends, as the bandwidth decreases the transmission times increase.

8.3.3 The Robot Client

Table 8.6 shows the command interaction between the CIM Host controller and the Robot Client. The time transmission performance of the commands are listed Table 8.7.

Command Number	CIM Host - Robot Client	Robot Client - CIM Host
1	ClientROB-START	
2	ClientROB-XAYBZC	
3	ClientROB-U-V-W-X-Y-Z	
4	ClientROB-EXIT	
5		HostROB-RUN
6		HostROB-XAYBZC
7		HostROB-U-V-W-X-Y-Z
8		HostROB-EXIT

Table 8.6. The command interaction between the CIM Host controller and the Robot Client.

The transmission time performance for each command between the CIM Host and Robot Client is plotted in Figure 8.5.

TIME (CAT)	CIM Host - Robot Client Commands				Robot Client - CIM Host Commands			
	1	2	3	4	5	6	7	8
	Number of Bytes (Including header and trailer)				Number of Bytes(Including header and trailer)			
	33	40	51	32	29	38	49	30
	Transmission Time (ms)				Transmission Time (ms)			
8:00	0.446	0.541	0.689	0.432	0.392	0.514	0.662	0.405
8:30	1.055	1.28	1.631	1.023	0.927	1.215	1.567	0.959
9:00	1.163	1.410	1.798	1.128	1.022	1.34	1.728	1.058
9:30	1.901	2.304	2.937	1.843	1.670	2.189	2.822	1.728
10:00	3.226	3.910	4.986	3.128	2.835	3.715	4.79	2.933
10:30	3.342	4.052	5.166	3.241	2.938	3.849	4.964	3.039
11:00	2.680	3.249	4.1423	2.599	2.355	3.086	3.98	2.437
11:30	2.973	3.604	4.595	2.883	2.613	3.424	4.415	2.703
12:00	3.676	4.456	5.681	3.565	3.230	4.233	5.458	3.342
12:30	4.517	5.475	6.981	4.379	3.969	5.201	6.707	4.106
13:00	7.891	9.565	12.195	7.651	6.934	9.087	11.717	7.174
13:30	6.912	8.379	10.682	6.703	6.075	7.96	10.263	6.284
14:00	5.854	7.096	9.047	5.677	5.144	6.741	8.693	5.322
14:30	3.816	4.625	5.897	3.700	3.353	4.39	5.666	3.469
15:00	3.228	3.912	4.988	3.130	2.836	3.72	4.793	2.934
15:30	2.936	3.559	4.538	2.847	2.580	3.38	4.36	2.669
16:00	2.519	3.054	3.894	2.443	2.214	2.901	3.741	2.290
16:30	0.752	0.911	1.162	0.729	0.661	0.866	1.116	0.683
17:00	0.506	0.613	0.782	0.490	0.445	0.583	0.751	0.460

Table 8.7. The transmission time performance between the CIM Host controller and the Robot Client in milliseconds.

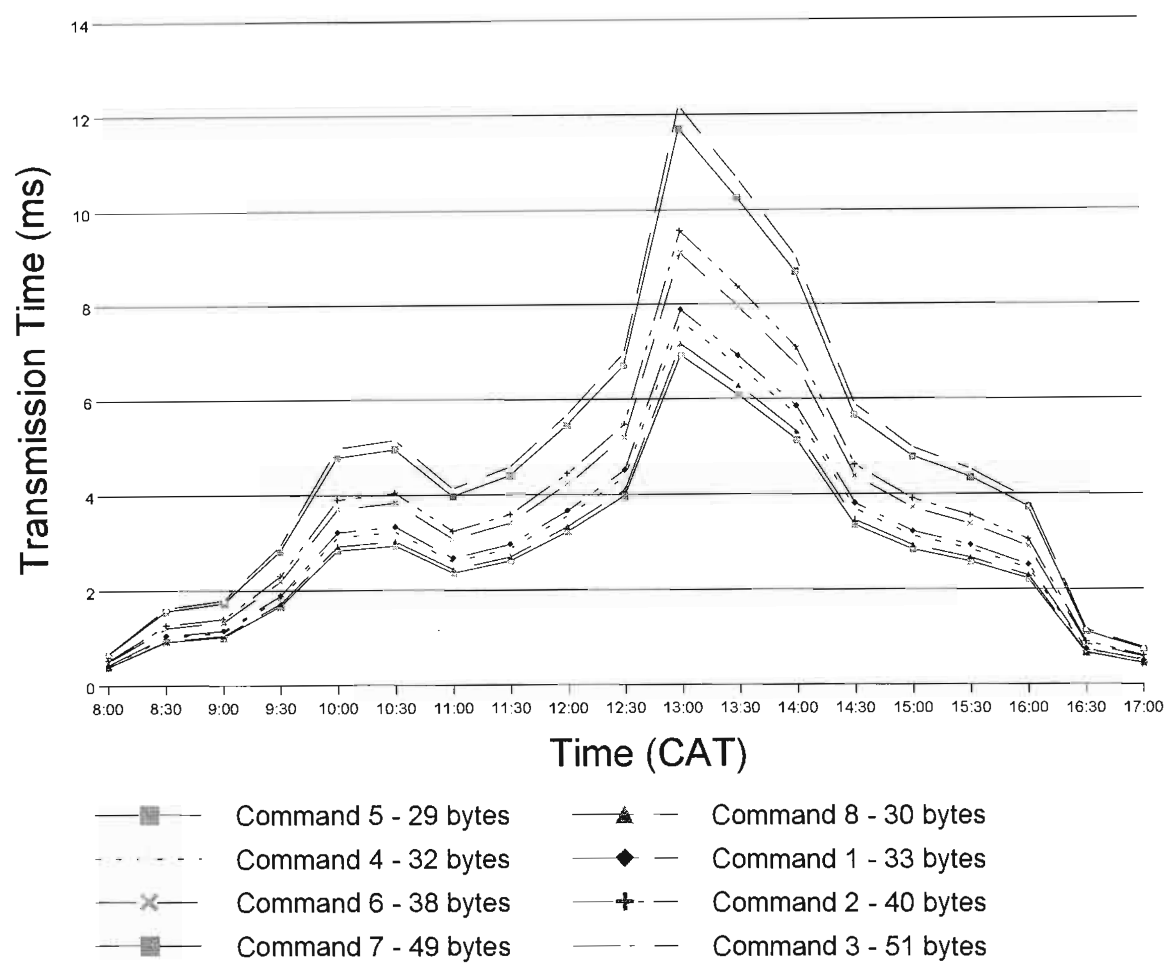


Figure 8.5. Graph of the transmission times between the CIM Host and the Robot Client.

From Figure 8.5. the longest transmission time interval is 12.195 milliseconds, for a 51 byte data packet at 13:00 hours (CAT). The Robot Client receives the command to position each robot joint at a specific angle from the CIM Host in less than 13 milliseconds from transmission. During off-peak times, 17:00 hours (CAT), this value decreases below 800 microseconds. Figure 8.5 clearly shows how increase between the transmission times per byte for low bandwidth (13:00) is greater than the increase at high bandwidth (08:00) times. This is due to the fact that the relationship between data packet sizes are not linear due to the addition of the header and trailer to each packet.

8.3.4 The WEB Client

Command Number	CIM Host - WEB Client	WEB Client - CIM Host
A	ClientWEB-START	
B	ClientWEB-AGVAVIS	
B	ClientWEB-AVISAGV	
C	ClientWEB-AGVROBOT	
C	ClientWEB-ROBOTAGV	
D	ClientWEB-AVISROBOT	
D	ClientWEB-ROBOTAVIS	
E	ClientWEB-CAM1	
E	ClientWEB-CAM2	
E	ClientWEB-CAM3	
E	ClientWEB-CAM4	
F	ClientWEB-XAYBZC	
G	ClientWEB-U-V-W-X-Y-Z	
E	ClientWEB-EXIT	
H		HostWEB-RUN
H		HostWEB-AGV
I		HostWEB-ROBOT
J		HostWEB-AVIS
H		HostWEB-CAM1
H		HostWEB-CAM2
H		HostWEB-CAM3
H		HostWEB-CAM4
K		HostWEB-XAYBZC
L		HostWEB-U-V-W-X-Y-Z
J		HostWEB-EXIT

Table 8.8. The command interactions between the CIM Host controller and the Web Client.

Table 8.8 shows the command interaction between the CIM Host controller and the remote WEB Client.. The time transmission performance of the commands are listed Table 8.9.

TIME (CAT)	CIM Host - WEB Client Commands							WEB Client - CIM Host Commands				
	A	B	C	D	E	F	G	H	I	J	K	L
	Number of Bytes (Including header and trailer)							Number of Bytes(Including header and trailer)				
	33	35	36	37	32	40	51	29	31	30	38	49
	Transmission Time (ms)							Transmission Time (ms)				
8:00	0.446	0.473	0.486	0.500	0.432	0.541	0.689	0.392	0.419	0.405	0.514	0.662
8:30	1.055	1.119	1.151	1.183	1.023	1.28	1.631	0.927	0.991	0.959	1.215	1.567
9:00	1.163	1.234	1.269	1.304	1.128	1.410	1.798	1.022	1.093	1.058	1.34	1.728
9:30	1.901	2.016	2.073	2.131	1.843	2.304	2.937	1.670	1.786	1.728	2.189	2.822
10:00	3.226	3.422	3.519	3.617	3.128	3.910	4.986	2.835	3.03	2.933	3.715	4.79
10:30	3.342	3.545	3.647	3.748	3.241	4.052	5.166	2.938	3.140	3.039	3.849	4.964
11:00	2.680	2.843	2.923	3.005	2.599	3.249	4.1423	2.355	2.518	2.437	3.086	3.98
11:30	2.973	3.153	3.243	3.333	2.883	3.604	4.595	2.613	2.793	2.703	3.424	4.415
12:00	3.676	3.899	4.010	4.122	3.565	4.456	5.681	3.230	3.453	3.342	4.233	5.458
12:30	4.517	4.791	4.927	5.064	4.379	5.475	6.981	3.969	4.243	4.106	5.201	6.707
13:00	7.891	8.369	8.608	8.847	7.651	9.565	12.195	6.934	7.413	7.174	9.087	11.717
13:30	6.912	7.331	7.540	7.75	6.703	8.379	10.682	6.075	6.493	6.284	7.96	10.263
14:00	5.854	6.208	6.386	6.563	5.677	7.096	9.047	5.144	5.499	5.322	6.741	8.693
14:30	3.816	4.047	4.163	4.278	3.700	4.625	5.897	3.353	3.585	3.469	4.39	5.666
15:00	3.228	3.423	3.521	3.619	3.130	3.912	4.988	2.836	3.032	2.934	3.72	4.793
15:30	2.936	3.114	3.203	3.292	2.847	3.559	4.538	2.580	2.759	2.669	3.38	4.36
16:00	2.519	2.672	2.748	2.825	2.443	3.054	3.894	2.214	2.367	2.290	2.901	3.741
16:30	0.752	0.797	0.820	0.843	0.729	0.911	1.162	0.661	0.706	0.683	0.866	1.116
17:00	0.506	0.537	0.552	0.567	0.490	0.613	0.782	0.445	0.475	0.460	0.583	0.751

Table 8.9. The transmission time performance between the CIM Host controller and the remote WEB Client in milliseconds.

Figure 8.6 shows the transmission times.

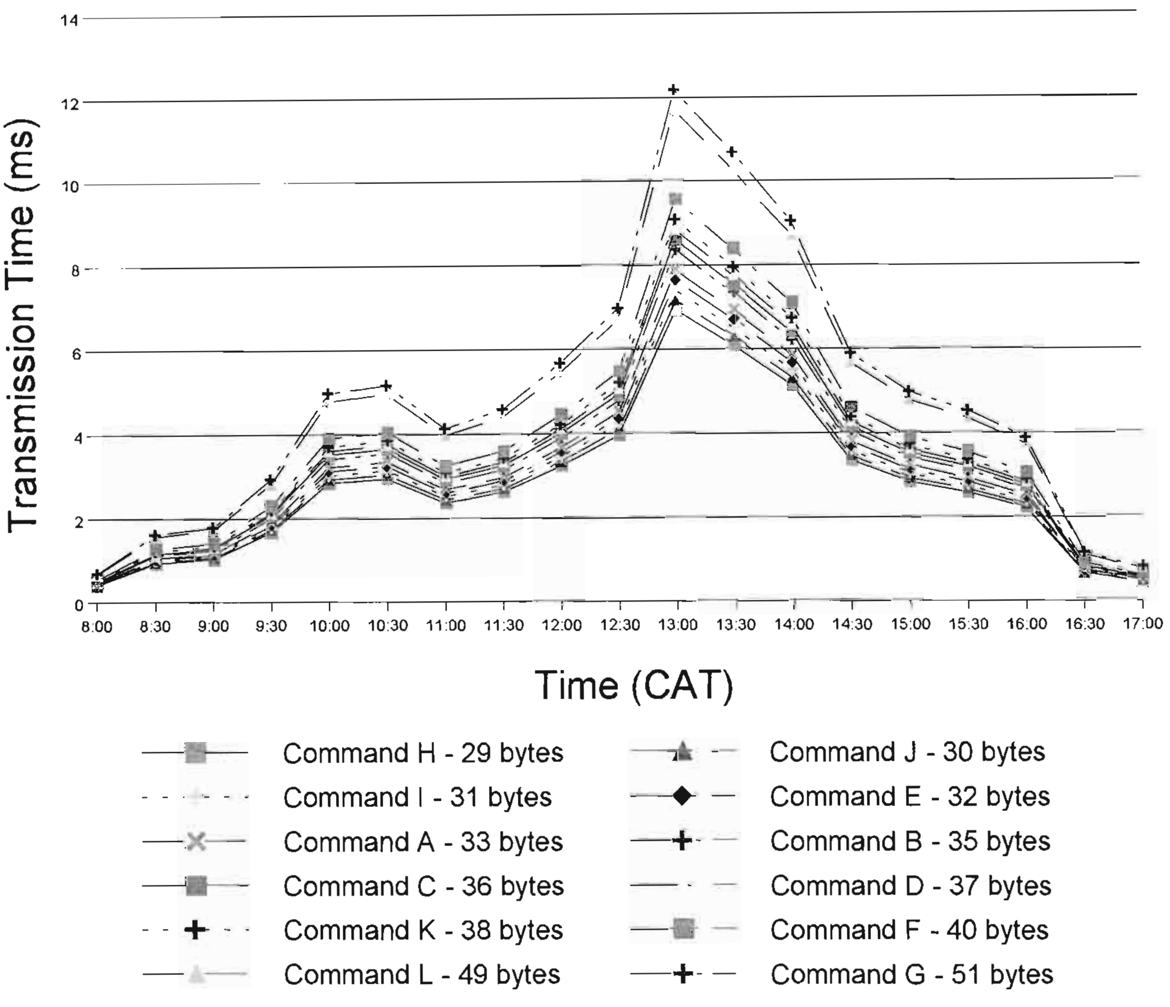


Figure 8.6. Graph of the transmission times between the CIM Host and the WEB Client.

From Figure 8.6. the longest transmission time interval is 12.195 milliseconds, for a 51 byte data packet at 13:00 hours (CAT), this value decreases below 800 microseconds. Figure 8.6 clearly shows how increase between the transmission times per byte for low bandwidth (13:00) is greater than the increase at high bandwidth (08:00) times.

This phenomenon is illustrated in Figure 8.7. The gradient of each time interval plot increases, as the bandwidth value decreases. This result illustrates the constraints bandwidth restrictions can place on Internet controlled manufacturing processes.

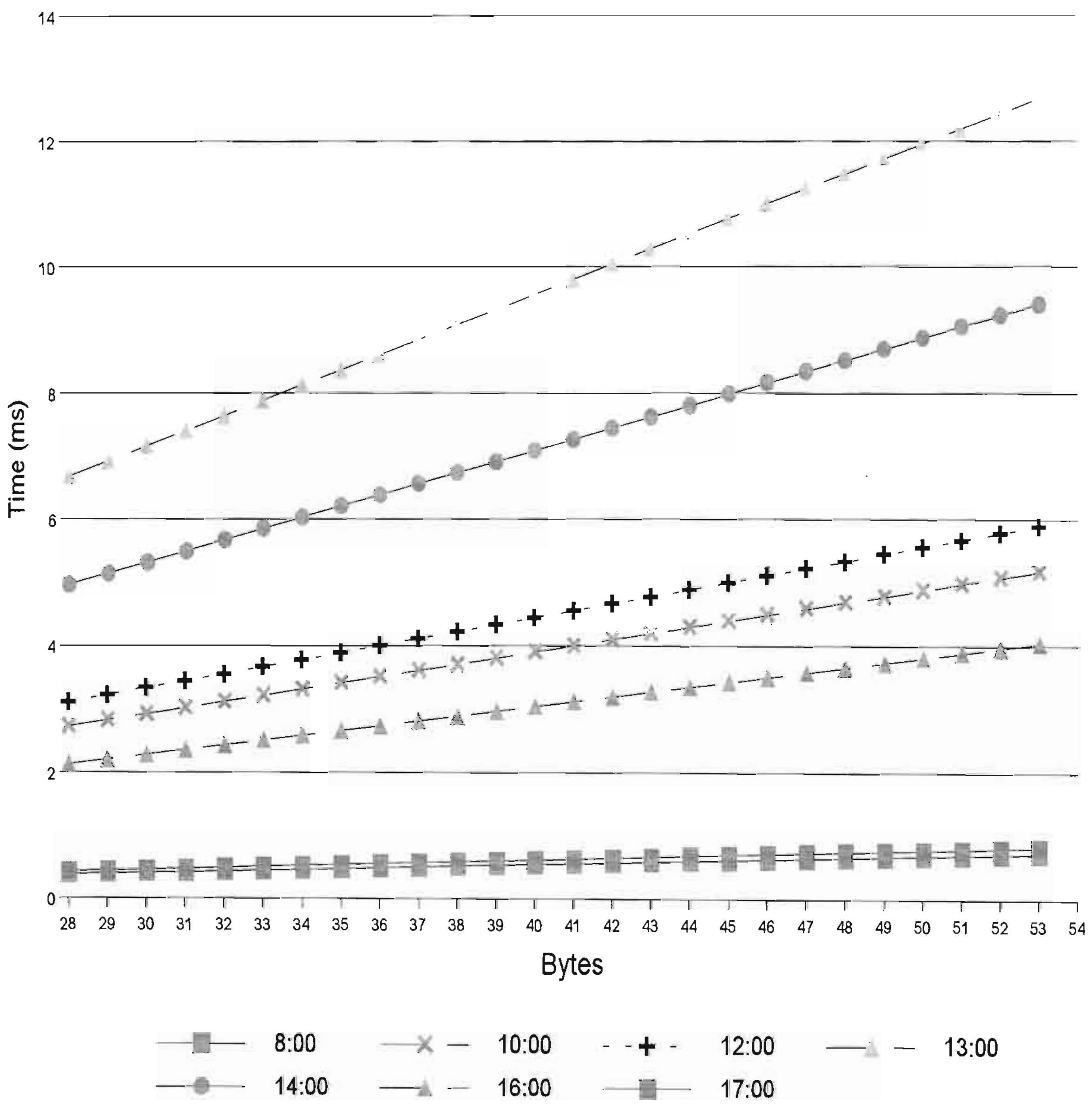


Figure 8.7. Shows the gradient increase for lower bandwidth values.

8.4 THE VISUAL FEEDBACK PERFORMANCE

Referring to Table 8.3, the performance of the computer network is dependant on the packet size of the information being transmitted. The *visual feedback* module of the CIM Internet control system relies on the available bandwidth of the network to transmit the live CIM cell image over the Internet. During development of the system, the image was pushed across the network at four frames per second.

The size of the image was 320 x 240 pixels totalling 18.2 KB (18699 Bytes). Figure 8.8 shows the performance graph of the visual feedback system.

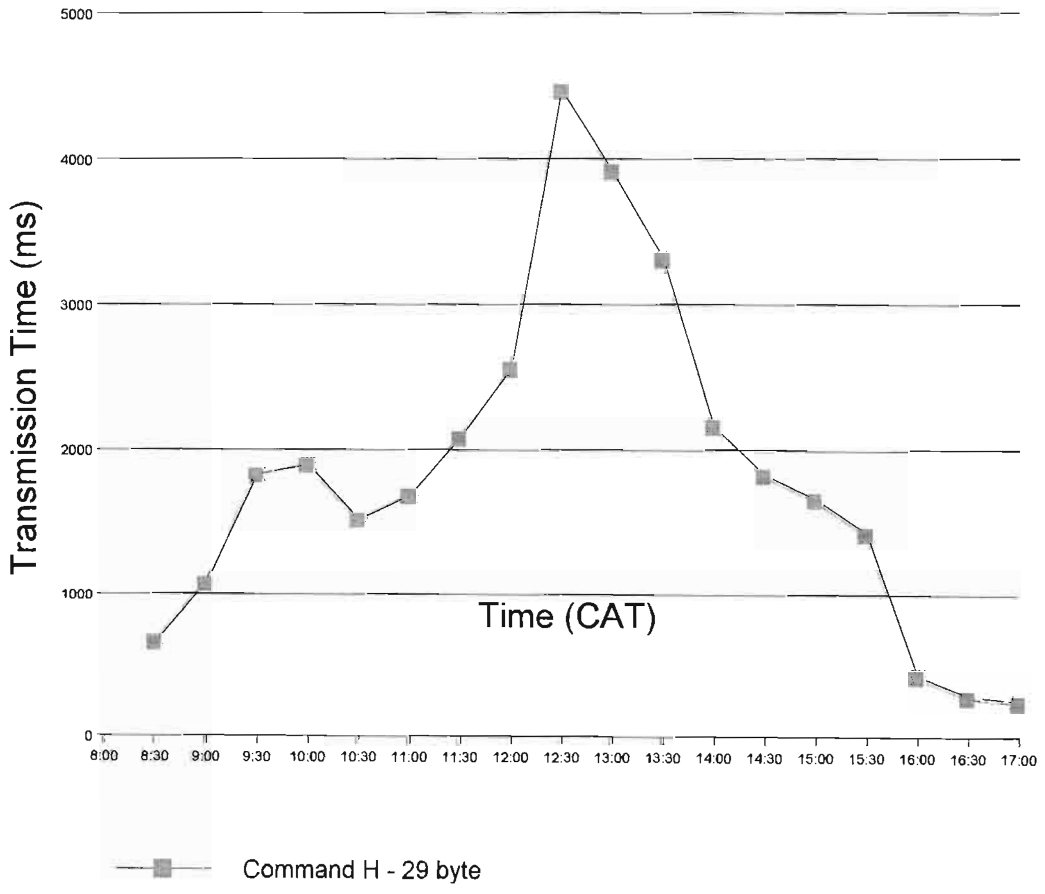


Figure 8.8. The visual feedback performance graph.

Referring to Figure 8.8, at 8:00 hours the Internet Image is updated every 252 ms. This result is satisfactory as the Web image is updated every 250 ms. However during peak times the image is updated every 4.47 seconds. The bandwidth required to update the live image feed is not sufficient to update the image every four seconds. The image overhead on the system is 366 times greater than the largest control signal to the CIM cell. The immediate solution to this problem lies in slower update speeds and reducing the size of the image.

CHAPTER 9

9 OPERATIONAL REVIEW OF THE MODULAR MECHATRONIC CIM INTERNET CONTROL SYSTEM

This chapter discusses the implementation of the modular mechatronic Internet control system in the CIM environment. The modular mechatronic Internet control system was implemented in a research-based computer integrated manufacturing cell in the School of Mechanical Engineering at the University of Natal.

The chapter also presents two other applications of the operational procedures adopted by the modular mechatronic Internet control system. The aim of this chapter is to enhance the reader's understanding of the overall function of the Internet control system, and to highlight the practical applications of the research presented.

9.1 IMPLEMENTATION OF THE MODULAR MECHATRONIC INTERNET CONTROL SYSTEM IN THE CIM ENVIRONMENT

This section discusses the implementation of the modular mechatronic CIM Internet control system in the computer integrated manufacturing environment. The Internet monitoring and control system was implemented using a modular mechatronic design methodology. It was fundamental that the computer-based technology could be integrated into a CIM environment. The Internet control system was successfully integrated and used to control and monitor various CIM systems online.

The communication between the various CIM Clients and CIM Host was achieved using the TCP/IP communications protocol. The system used low-level control commands to instruct the various CIM controllers to perform a task. The CIM controllers relied on primary machine intelligence to perform the task. This control structure eliminated Internet lag problems associated with the control of machines over vast network distances. The standardisation of Mechatronic components simplified the control and monitoring of the various CIM components. Figure 9.1 shows the command interactions between the various CIM Client components and the CIM Host controller. Table 9.1 contains the control commands as graphically illustrated in Figure 9.1.

The CIM Host controller communicates with the CIM Clients using one WinSock controller for each Client. To connect to a WinSock controller requires an IP address and a port. On the CIM Host controller the IP address stays fixed and each Client connects to a different Port number. Simply adding more Winsock Controls allows for more Clients to connect to the Host controller, which provides the system with modularity. If the connecting client tries to connect using a conflicting port number, the Host controller will forcefully reject the connection.

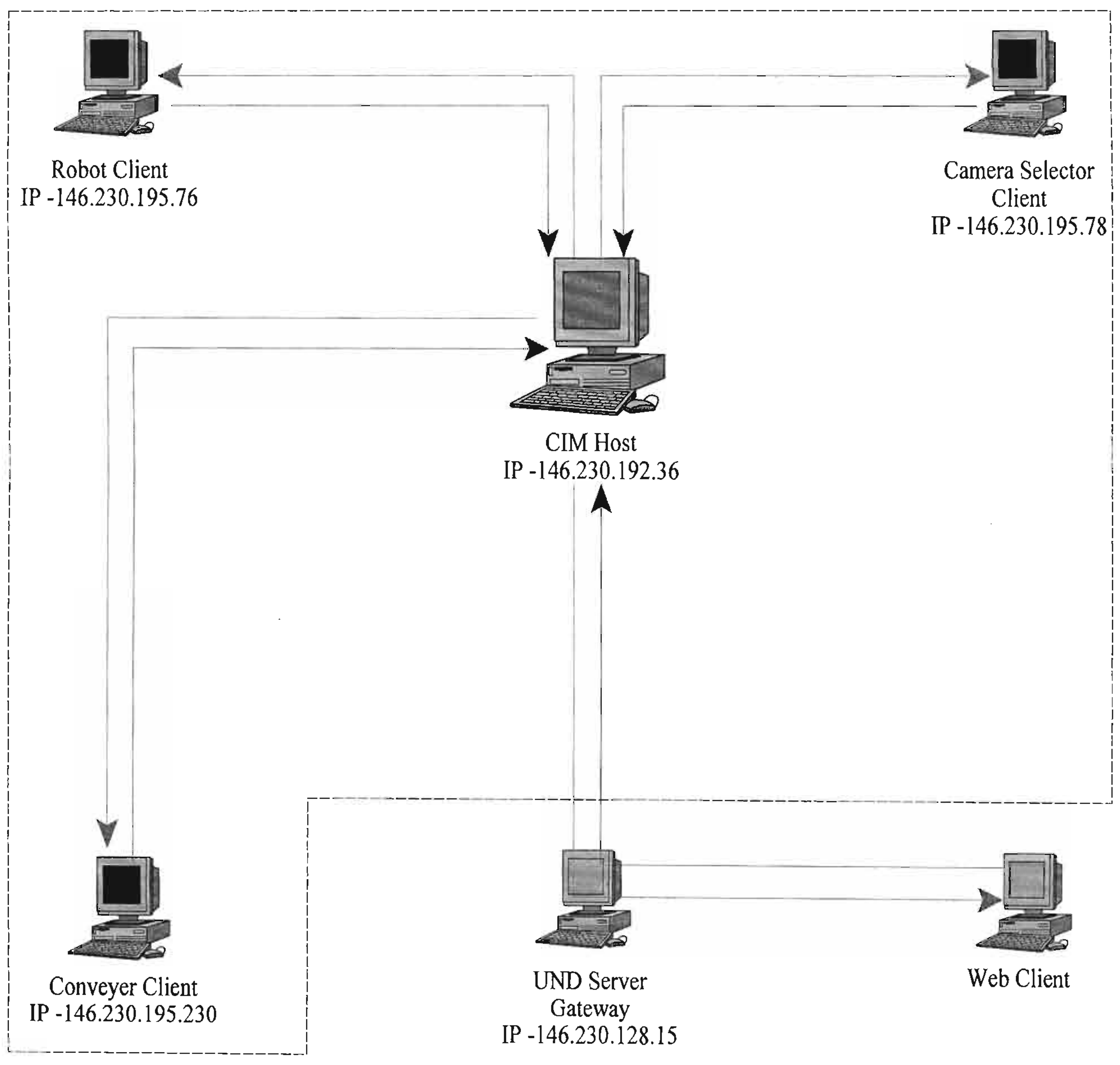


Figure 9.1. The interactions between the various CIM Client components and the CIM Host controller

Network Commands			
Camera Selector Client	**Robot Client**	**Conveyer Client**	**Web Client**
ClientCAM-CAMERA1	ClientROB-START	ClientCON-START	ClientWEB-START
ClientCAM-CAMERA2	ClientROB-XAYBZC	ClientCON-AGVAVIS	ClientWEB-AGVAVIS
ClientCAM-CAMERA3	ClientROB-U-V-W-X-Y-Z	ClientCON-AVISAGV	ClientWEB-AVISAGV
ClientCAM-CAMERA4	ClientROB-EXIT	ClientCON-AGVROBOT	ClientWEB-AGVROBOT
ClientCAM-EXIT	HostROB-RUN	ClientCON-ROBOTAGV	ClientWEB-ROBOTAGV
HostCAM-CAMERA1	HostROB-XAYBZC	ClientCON-AVISROBOT	ClientWEB-AVISROBOT
HostCAM-CAMERA2	HostROB-U-V-W-X-Y-Z	ClientCON-ROBOTAVIS	ClientWEB-ROBOTAVIS
HostCAM-CAMERA3	HostROB-EXIT	ClientCON-EXIT	ClientWEB-CAM1
HostCAM-CAMERA4		HostCON-AGV	ClientWEB-CAM2
HostCAM-EXIT		HostCON-ROBOT	ClientWEB-CAM3
		HostCON-AVIS	ClientWEB-CAM4
		HostCON-RUN	ClientWEB-XAYBZC
		HostCON-EXIT	ClientWEB-U-V-W-X-Y-Z
			ClientWEB-EXIT
			HostWEB-RUN
			HostWEB-AGV
			HostWEB-ROBOT
			HostWEB-AVIS
			HostWEB-CAM1
			HostWEB-CAM2
			HostWEB-CAM3
			HostWEB-CAM4
			HostWEB-XAYBZC
			HostWEB-U-V-W-X-Y-Z
			HostWEB-EXIT

Table 9.1. The control commands as graphically illustrated in Figure 9.1.

9.2 OPERATIONAL REVIEW OF THE SUPERVISORY CONTROL SYSTEM

9.2.1 The Remote Client Website

The CIM Internet control system is accessed via a Website hosted on the University of Natal shrike Server. The URL contains the hyperlink to CIM cell Host controller. When the URL is invoked, the remote user loads the CIM home page to their browser. Figure 9.2 shows the CIM Internet Control System home page.

The home page displays an overall picture of the CIM cell in the MR^{2}G laboratory. The remote user has the option to use any one of the four navigation bars on the home page. Each navigation bar contains a hyperlink to another page on the Website. Each page contains the relevant information of the system and the applications needed to operate the CIM cell remotely. These applications are distributed and installed on the remote Client machine as self extracting executable applications. Figure 9.3 shows the CIM Video Feed Web page with a hyperlink to the *Video.html* Web page on the CIM Host Controller. Figure 9.4 shows the live video feed page, which is dynamically updated at four frames per second.

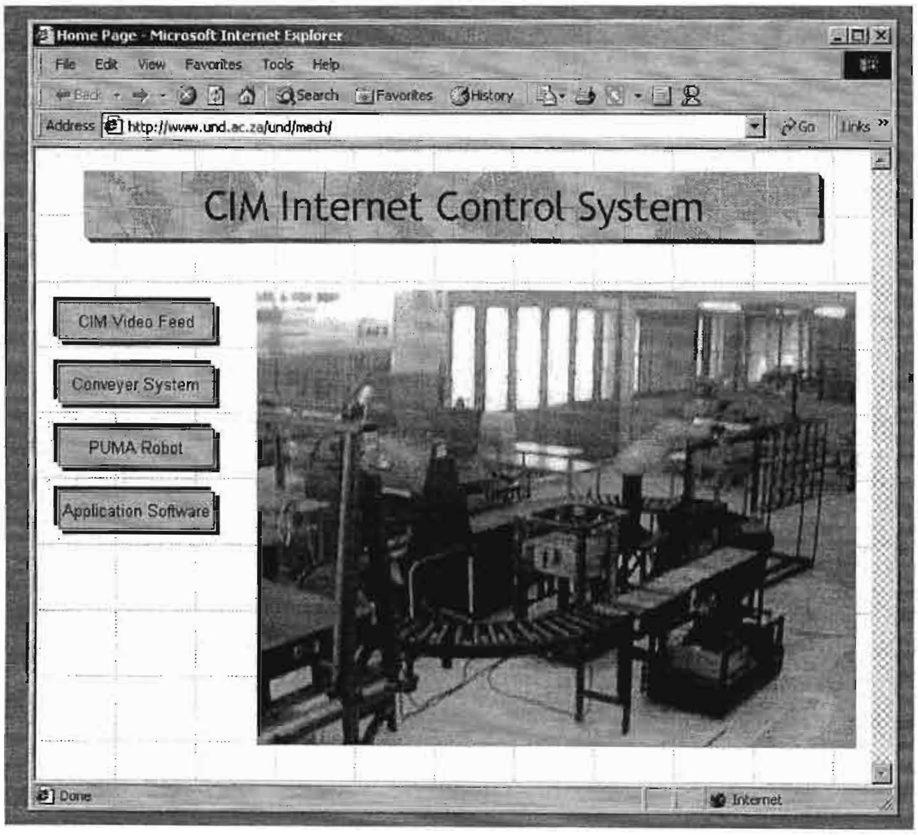
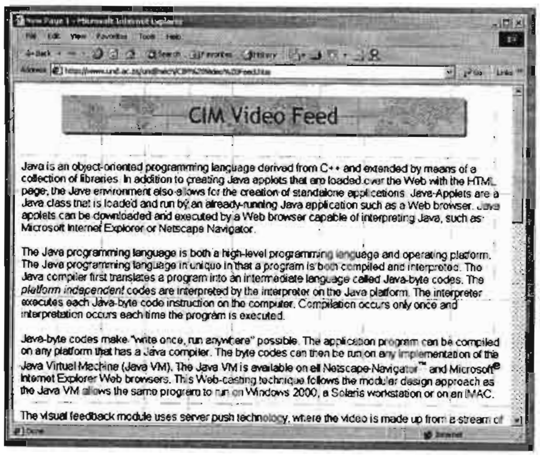

Home Page - Microsoft Internet Explorer
http://www.und.ac.za/und/mech/
CIM Internet Control System
CIM Video Feed
Conveyer System
PUMA Robot
Application Software

Figure 9.2. The CIM Control System home page.


CIM Video Feed

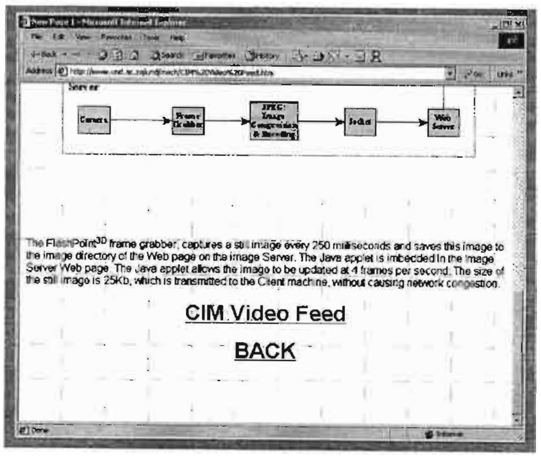
Java is an object-oriented programming language derived from C++ and extended by means of a collection of libraries. In addition to creating Java applets that are loaded over the Web with the HTML page, the Java environment also allows for the creation of standalone applications. Java-Applets are a Java class that is loaded and run by an already-running Java application such as a Web browser. Java applets can be downloaded and executed by a Web browser capable of interpreting Java, such as Microsoft Internet Explorer or Netscape Navigator.

The Java programming language is both a high-level programming language and operating platform. The Java programming language in unique in that a program is both compiled and interpreted. The Java compiler first translates a program into an intermediate language called Java-byte codes. The *platform independent* codes are interpreted by the interpretor on the Java platform. The interpreter executes each Java-byte code instruction on the computer. Compilation occurs only once and interpretation occurs each time the program is executed.

Java-byte codes make "write once, run anywhere" possible. The application program can be compiled on any platform that has a Java compiler. The byte codes can then be run on any implementation of the Java Virtual Machine (Java VM). The Java VM is available on all Netscape-Navigator™ and Microsoft® Internet Explorer Web browsers. This Web-casting technique follows the modular design approach as the Java VM allows the same program to run on Windows 2000, a Solaris workstation or on an IMAC.

The visual feedback module uses server push technology, where the video is made up from a stream of

Figure 9.3(a). The CIM Video Feed Web page 1.



The FlashPoint3D frame grabber, captures a still image every 250 milliseconds and saves this image to the image directory of the Web page on the image Server. The Java applet is imbedded in the image Server Web page. The Java applet allows the image to be updated at 4 frames per second. The size of the still image is 25Kb, which is transmitted to the Client machine, without causing network congestion.

CIM Video Feed

BACK

Figure 9.3(b). The CIM Video Feed Web page 2.

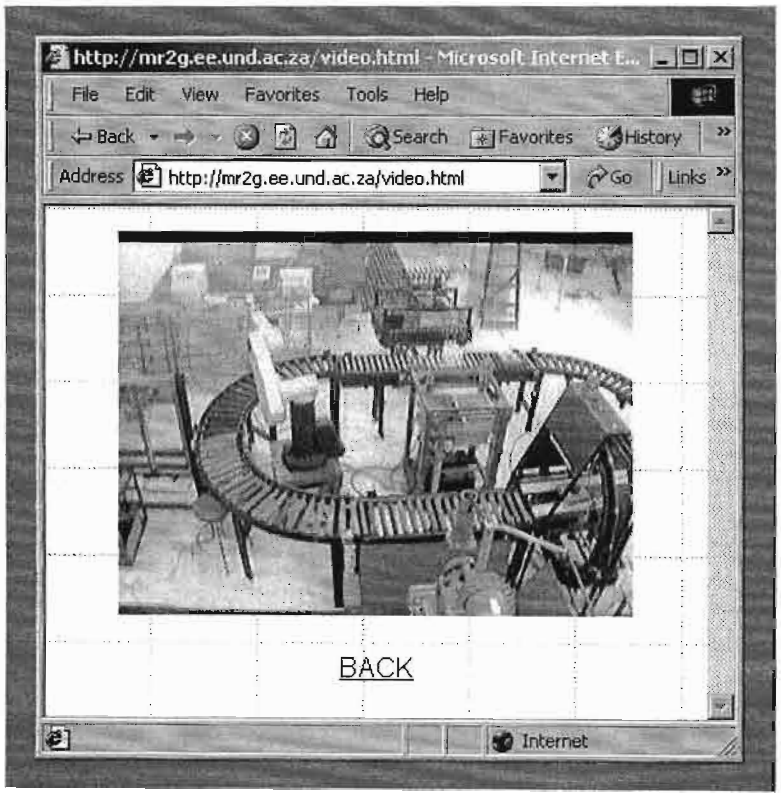
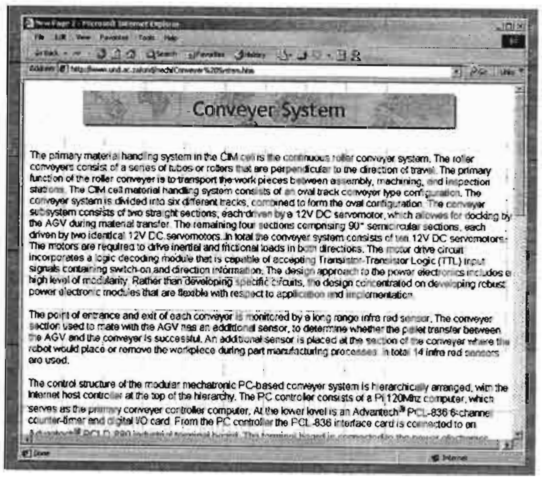

http://mr2g.ee.und.ac.za/video.html - Microsoft Internet E...
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://mr2g.ee.und.ac.za/video.html Go Links
BACK
Internet

Figure 9.4. The live video feed Web page.

Figure 9.4 shows the image of camera 1. This camera view is transmitted over the Internet using the server push technology as discussed in Chapter 5. Figure 9.5 shows the Conveyer System Web page. This page contains the information regarding the workstation assignments of the CIM conveyer system.



The primary material handling system in the CIM cell is the continuous roller conveyer system. The roller conveyers consist of a series of tubes or rollers that are perpendicular to the direction of travel. The primary function of the roller conveyer is to transport the work pieces between assembly, machining, and inspection stations. The CIM cell material handling system consists of an oval track conveyor type configuration. The conveyer system is divided into six different tracks, combined to form the oval configuration. The conveyer subsystem consists of two straight sections, each driven by a 12V DC servomotor, which allows for docking by the AGV during material transfer. The remaining four sections comprising 90° semicircular sections, each driven by two identical 12V DC servomotors. In total the conveyer system consists of ten 12V DC servomotors. The motors are required to drive inertial and frictional loads in both directions. The motor drive circuit incorporates a logic decoding module that is capable of accepting Transistor-Transistor Logic (TTL) input signals containing switch-on and direction information. The design approach to the power electronics includes a high level of modularity. Rather than developing specific circuits, the design concentrated on developing robust power electronic modules that are flexible with respect to application and implementation.

The point of entrance and exit of each conveyor is monitored by a long range infra red sensor. The conveyer section used to mate with the AGV has an additional sensor, to determine whether the pallet transfer between the AGV and the conveyer is successful. An additional sensor is placed at the section of the conveyer where the robot would place or remove the workpiece during part manufacturing processes. In total 14 infra red sensors are used.

The control structure of the modular mechatronic PC-based conveyer system is hierarchically arranged, with the Internet host controller at the top of the hierarchy. The PC controller consists of a PI 120Mhz computer, which serves as the primary conveyer controller computer. At the lower level is an Advantech® PCL-836 6-channel counter-timer and digital I/O card. From the PC controller the PCL-836 interface card is connected to an [illegible]

Figure 9.5(a). The Conveyer System Web page 1.

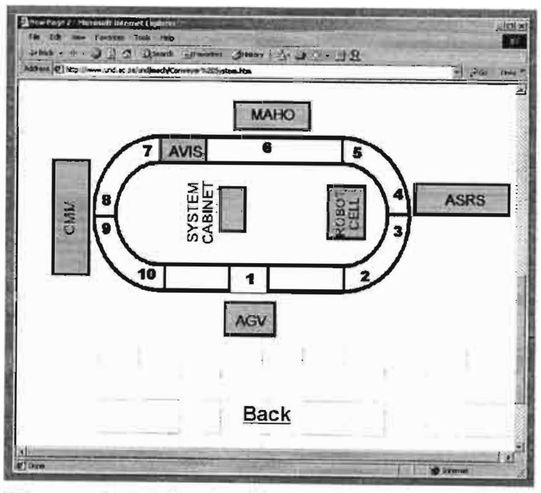
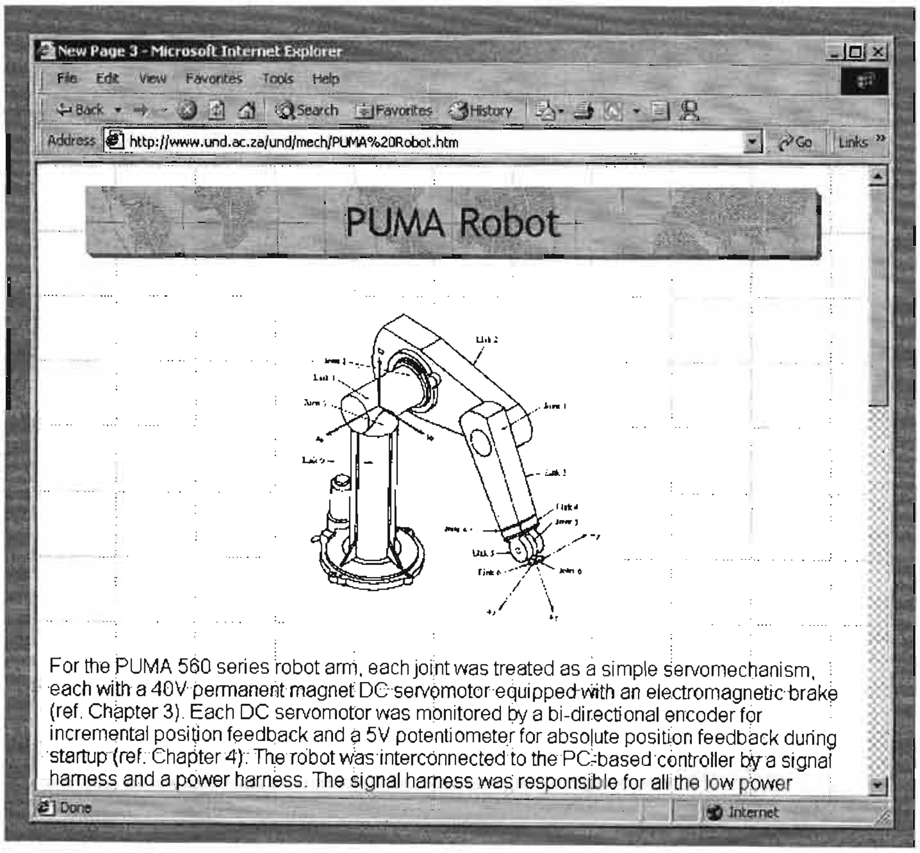


Figure 9.5(b). The Conveyer System Web page 2.

The position and movement of the pallet is controlled using preprogrammed algorithms. The remote Client makes use of the supervisory control strategy, while the primary control tasks are executed by the local conveyer Client. Figure 9.6 shows the PUMA Robot Web Page. This page contains the information regarding the PUMA Robot.


New Page 3 - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://www.und.ac.za/und/mech/PUMA%20Robot.htm
Go Links
PUMA Robot
Link 2
Link 1
Link 3
Link 4
Link 5
Link 6
Done
Internet

For the PUMA 560 series robot arm, each joint was treated as a simple servomechanism, each with a 40V permanent magnet DC servomotor equipped with an electromagnetic brake (ref. Chapter 3). Each DC servomotor was monitored by a bi-directional encoder for incremental position feedback and a 5V potentiometer for absolute position feedback during startup (ref. Chapter 4). The robot was interconnected to the PC-based controller by a signal harness and a power harness. The signal harness was responsible for all the low power

Figure 9.6. The PUMA Robot Web page.

The Application Software Web page contains the remote Client software required to control the CIM components over the Web. The software is downloaded to the remote Client machine via the CIM Host ftp:// server. The application software was written in VB6 and uses ActiveX and TCP/IP technology for Internet connectivity and control.

9.2.2 The Visual Feedback Module

The visual feedback module was designed as a robust online monitoring system and as a result the operation of the module is greatly simplified. The Host Controller (Server) interacts and controls the visual feedback module, using preprogrammed control algorithms. The Host Controller interacts with the high level graphical user interface (GUI), *MainServer.vbp* and *Video.java*, software installed on the host controller. The Host Controller also interacts and communicates with the camera selector controller, to select and activate the appropriate camera, using the *CAMClient.vbp*, software. The various visual feedback software modules were developed as Windows 95/98/2000® applications. The software could be invoked from the start menu of the Host and Camera Selector Controllers. The *ServerMain.vbp* software also contained the TCP/IP communication backbone between the Host (Server) Controller and the CIM cell controllers.

When the Visual Feedback Module is launched, the GUI [*fflashpt.frm*] is shown on the screen. The GUI displays the visual feedback information on the screen allowing manual control of the visual feedback process and configures the FlashPoint^{3D} frame grabber card (ref. Figure 9.7). Once the configuration functions have been completed, the [*Server.frm*] is loaded, but kept hidden from the desktop display. The *Server.frm* control software monitors the entire network for control signals from the Web Client, Robot Client, Conveyer Client and Camera Selector Client controllers.

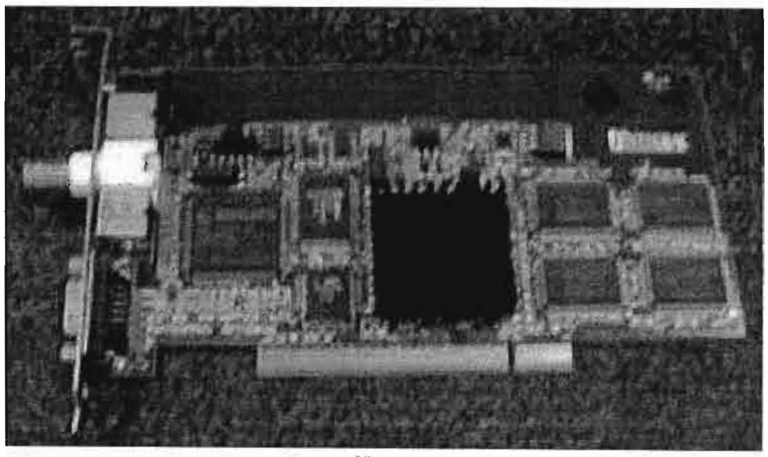
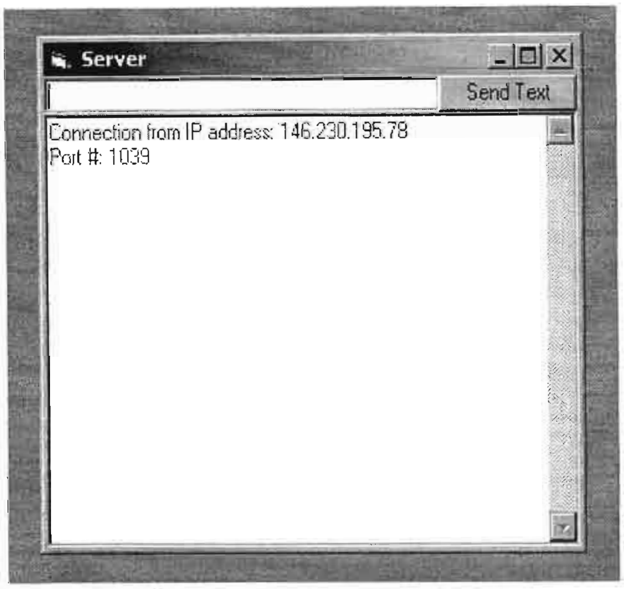


Figure 9.7. The FlashPoint^{3D} frame grabber card.

This process continuos for the entire duration of the manufacturing process. The *fflashpt.frm* is contained in the *flashptb.vbp* project, which controls the *SaveImg.frm*. The *SaveImg.frm* captures and saves the live video image to the c:/Inetpub/wwwroot/webcam32.jpg file on the Web Server. This file is updated every 250ms, updating the live video feed at four frames per second. The *Server.frm* communicates with the Video Selector Client, which in turn selects the appropriate camera as per a preprogrammed algorithm. Figure 9.8 shows the *Server.frm*.



Connection from IP address: 146.230.195.78
Port #: 1039

Figure 9.8. The Server.frm form. Note the server has connected to IP address 146.230.195.78, which is the Camera Selector Client.

As the *webcam32.jpg* file is updated every 250ms, the image is loaded to the *Video.java* applet which is imbedded in the *Video.html* web page that resides on the Web Server. The Java applet has been programmed to update the Web browser on the client side every 250ms, allowing the remote Web Client to view the CIM process at four frames per second. Figure 9.9 shows the *Video.html* page and the four different camera views of the CIM cell. The performance of the Visual Feedback module was dependant on the network bandwidth limitations at different operating times. Figures 9.9 (a)(b)(c)(d) shows the transmitted camera views at 11:00 hours. This produced an image update every second. This image update time was sufficient to monitor the CIM cell while using the supervisory control structure.

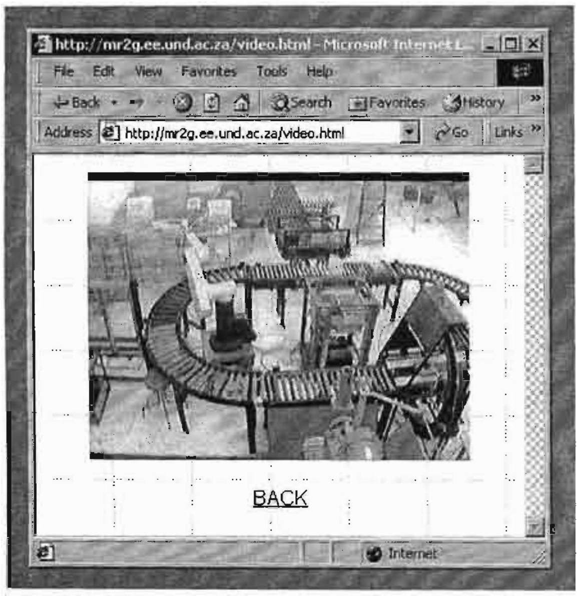

http://mr2g.ee.und.ac.za/video.html - Microsoft Internet E...
File Edit View Favorites Tools Help
Back Search Favorites History
Address http://mr2g.ee.und.ac.za/video.html Go Links
BACK
Internet

Figure 9.9(a). Camera view 1.

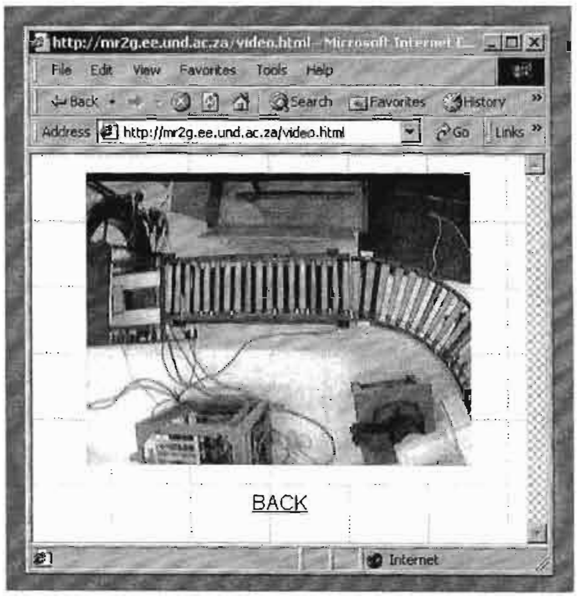

http://mr2g.ee.und.ac.za/video.html - Microsoft Internet E...
Address http://mr2g.ee.und.ac.za/video.html
BACK
Internet

Figure 9.9(b). Camera view 2.

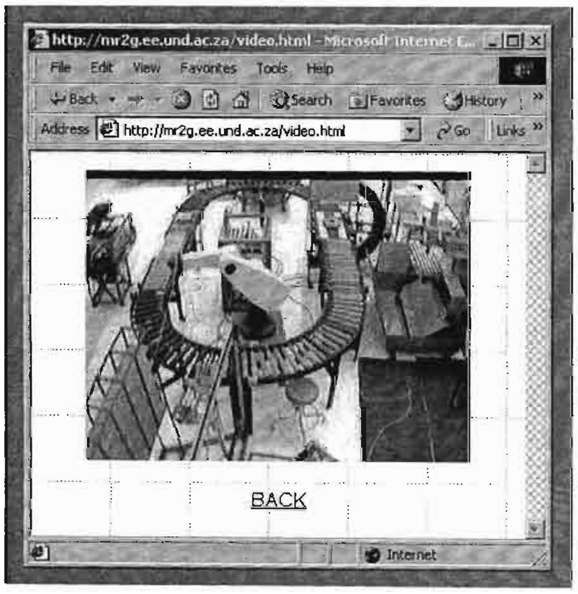

http://mr2g.ee.und.ac.za/video.html - Microsoft Internet E...
Address http://mr2g.ee.und.ac.za/video.html
BACK
Internet

Figure 9.9(c). Camera view 3.

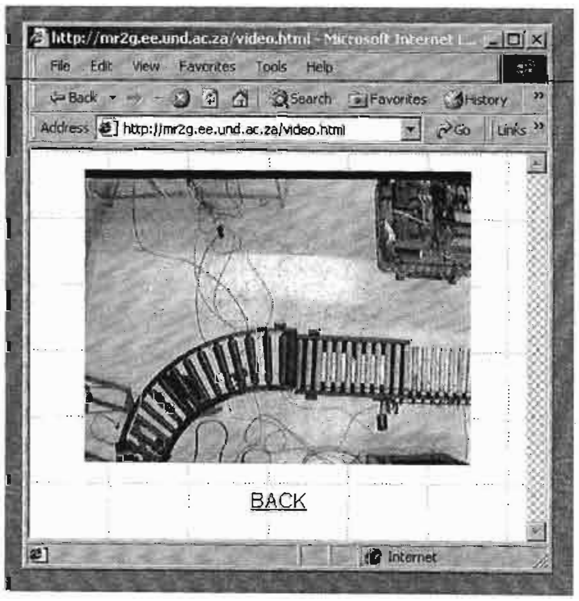

http://mr2g.ee.und.ac.za/video.html - Microsoft Internet E...
Address http://mr2g.ee.und.ac.za/video.html
BACK
Internet

Figure 9.9(d). Camera view 4.

The Host Controller communicates with the Camera Selector Client using five different commands. These commands are shown in Table 9.1.

When the Host Controller transmits any messages across the network which does not match the commands from Table 9.1, the Camera Selector Client ignores the erroneous messages. The cameras are selected using the camera selector circuit, which uses the *relay power module* circuit to switch between the four installed cameras. Figure 9.10 shows the camera selector circuit.

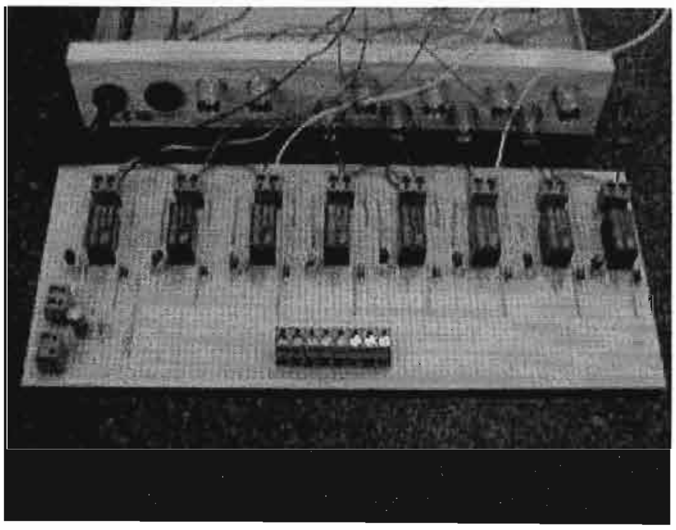


Figure 9.10. The camera selector circuit.

The circuit uses modular functionality as the circuit is capable of switching eight cameras.

9.2.3 The Web Client Control Software

The control software was written in VB6. As the modular Mechatronic design methodology was used, the Internet control is achieved using the TCP/IP network backbone (ref. Figure 9.11).

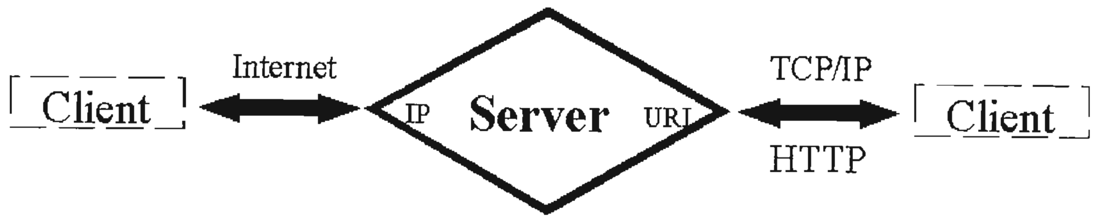


Figure 9.11. The TCP/IP network backbone for CIM hardware control and monitoring.

The CIM Host Controller (Server) receives the supervisory control signals from the remote Web Client and processes and relays the appropriate control signals to the robot Client, conveyer client and camera selector client controllers. The Host Controller transmits the control information across the entire CIM network, but only the appropriate CIM components receive and use the information. The Client controller setup allows the Web Client to only communicate with the Host Controller. The Host Controller receives the Web Client commands and processes the information according to a preprogrammed algorithm.

Figure 9.12 shows the ActiveX Web Client control software GUI. The software is installed as a self extracting executable file. The Web Client control software now resides on the remote client web browser, and connects via the TCP/IP protocols to the Host controller in the MR^{2}G laboratory. The CIM control program connects to the Host controller through a preprogrammed default IP address. The IP address is a fixed address, which simplifies connectivity.

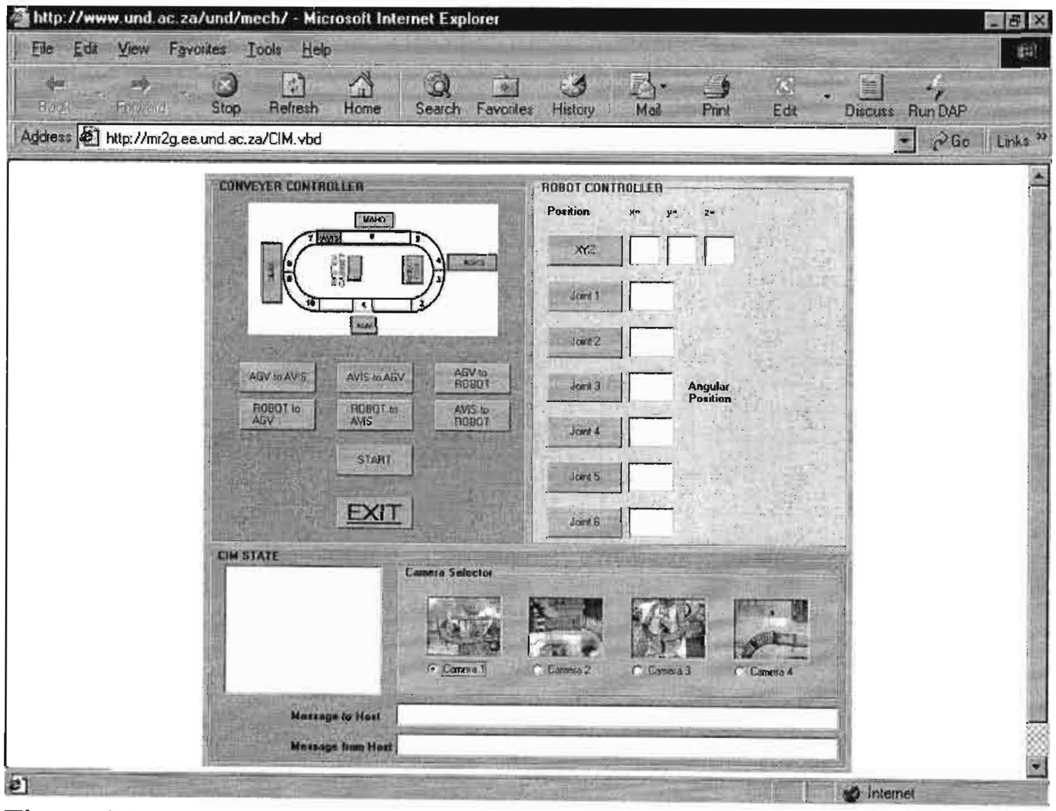


Figure 9.12. The ActiveX CIM control GUI.

Referring to Figure 9.12, the remote Client has the option to control the conveyer or robot on the same control window. The supervisory control structure of the conveyer system allows the remote Client to select a predetermined part process command. As the routine is executed, the appropriate camera view is automatically selected. At the bottom of the screen two message boxes show the messages relayed to and from the Host controller.

The CIM Client robot control is limited to positioning the robot end-effector at an XYZ Cartesian coordinate. The robot can be controlled by entering the XYZ coordinates of the end-effector or by individually rotating each of the six robot joints manually. The remote Client controller can manually select the various cameras in the laboratory.

9.3 OPERATIONAL REVIEW OF THE PRIMARY CIM CONTROL SYSTEM

9.3.1 The Conveyer Controller

The CIM conveyer system is a computer based technology, which incorporates mechatronic actuators, feedback devices and network hardware. The conveyer system is controlled remotely using the supervisory control strategy. The primary conveyer controller contains the machine logic to efficiently execute the supervisory control commands from the remote Web Client. The control software was written in VB6.

The CIM Host Controller (Server) receives the supervisory control signals from the remote Web Client and processes and relays the appropriate control signals to the conveyer Client controller. The Host Controller transmits the control information across the entire CIM network, but only the appropriate CIM components receive and use the information. The conveyer Client controller setup allows the conveyer Client to only communicate with the Host Controller. The Host Controller receives the conveyer Client feedback commands and processes the information according to a preprogrammed algorithm.

Figure 9.13 shows the [*conveyer.frm*] form. The conveyer system can be controlled manually or over the Internet. The conveyer form is located in the [*conveyer.vbd*] project. The conveyer project also contains the [*frmTCPClient.frm*] form. The frm *TCPClient* form is not visible to the operator and continues to monitor the network for the appropriate control signals from the CIM Host Controller.

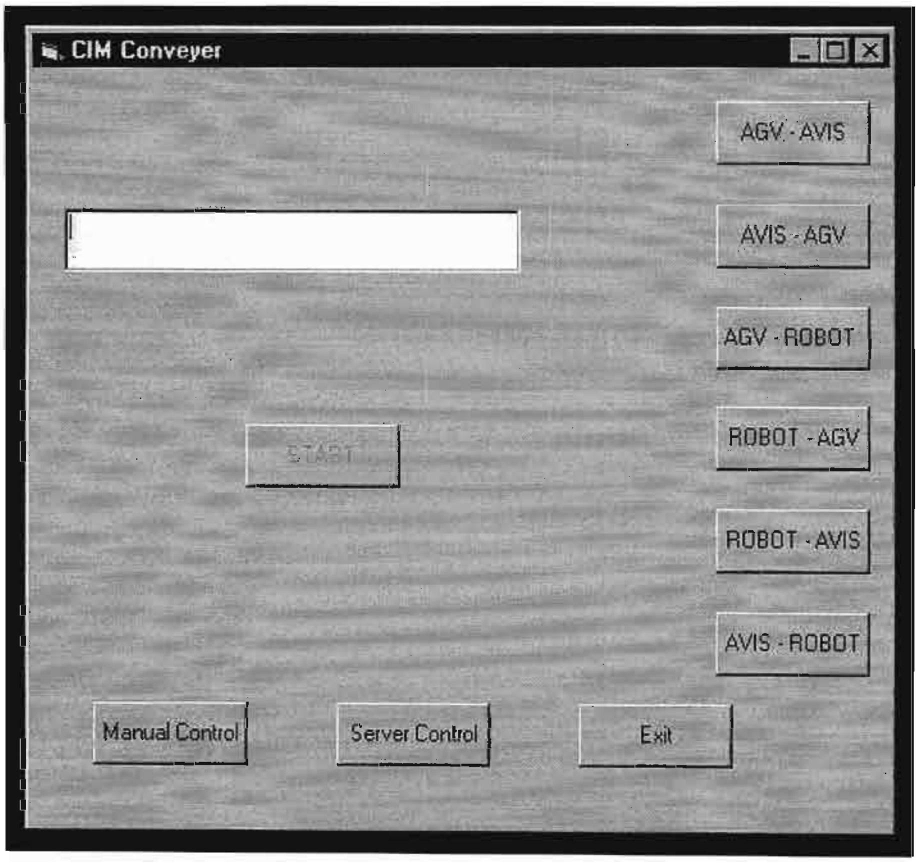


Figure 9.13. The front panel of the conveyer controller.

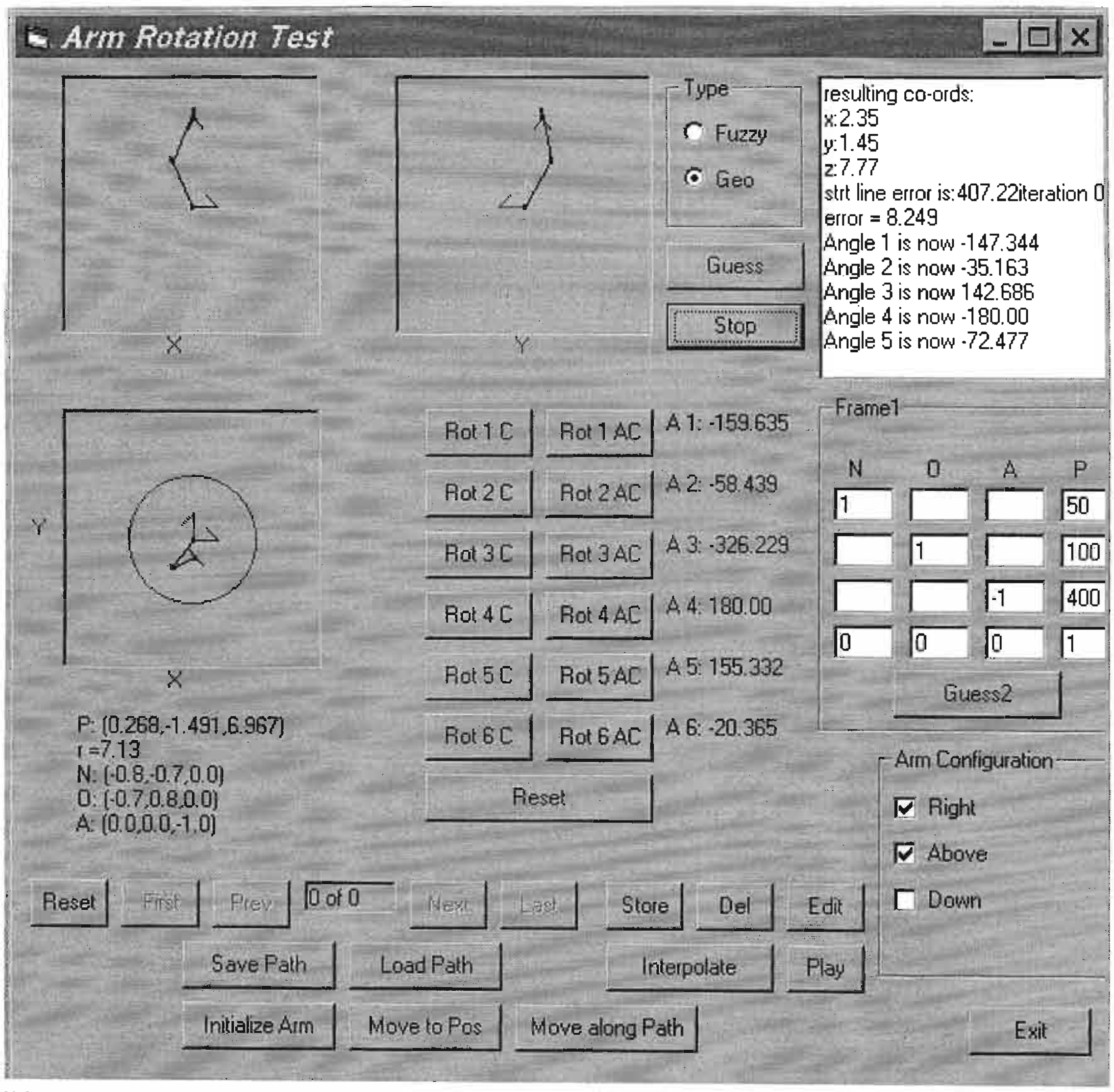
(Ref. Figure 9.13) The CIM conveyer front panel provides the operator with preprogrammed material handling tasks to choose from. Once the CIM conveyer is under Server Control the manual control button is disabled allowing no manual control of any of the processes. The control buttons on the conveyer front panel refer to the material handling task to be executed. For example the [AGV-AVIS] control routine, moves the pallet from the AGV docking station to the AVIS (Automated Visual Inspection System) part inspection station. When the conveyer project is executed, the conveyer Client is automatically connected to the Host Controller. Table 9.1 contains the control commands between the Host controller and the conveyer Client. The conveyer Client continuously updates the Host Controller on network status and pallet position. The remote Web Client can visually see if the conveyer system is operational, as the conveyer system activates an amber warning light in the laboratory as the conveyer starts.

9.3.2 Operational Review of the PUMA Robot

The PUMA industrial robot is a computer-based technology, since the dedicated robot controller was discarded and replaced with a PC-based controller. The PC-based PUMA robot is initialised at the primary controller and controlled remotely, using the supervisory control strategy. The robot control strategy is similar to the control strategy employed by the conveyer system. The robot controller software was written in VB6, although the kinematic matrix solver software was written in Visual C and compiled as a dynamic link library (DLL).

Figure 9.13 shows the front panel of the PC-based PUMA robot controller. The software for the PUMA controller was contained in the *RobotClient.vbp* VB6 project. The front panel was called *frmInvKin.frm*. The project file also contained two other GUI's, the TCP/IP communications form [*frmClient.frm*] and the Advantech PLC-832 servo card initialisation form [*frmPCL832.frm*]. Communication with the servo control cards was done by using the *funcs.bas* and *PCL832.bas* drivers.

Referring to Figure 9.14, Figure 9.15 shows the robot arm orientations with respect to the world coordinate frame. The XY frame refers to the PLAN view. The XZ frame refers to the FRONT view, while the YZ frame represents the SIDE view of the robot arm.



Type: Fuzzy [✓ Geo]

```
resulting co-ords:
x:2.35
y:1.45
z:7.77
strt line error is:407.22iteration 0
error = 8.249
Angle 1 is now -147.344
Angle 2 is now -35.163
Angle 3 is now 142.686
Angle 4 is now -180.00
Angle 5 is now -72.477
```

A 1: -159.635
A 2: -58.439
A 3: -326.229
A 4: 180.00
A 5: 155.332
A 6: -20.365

P: (0.268,-1.491,6.967)
r =7.13
N: (-0.8,-0.7,0.0)
O: (-0.7,0.8,0.0)
A: (0.0,0.0,-1.0)

0 of 0

Frame1

N	O	A	P
1			50
	1		100
		-1	400
0	0	0	1

Arm Configuration: [✓ Right] [✓ Above] Down

Figure 9.14. The front panel of the PC-based PUMA robot controller.

Figure 9.16 shows the manual robot arm control panel, each command button rotates the associated joint by ten degrees. The column on the right displays the current arm positions.

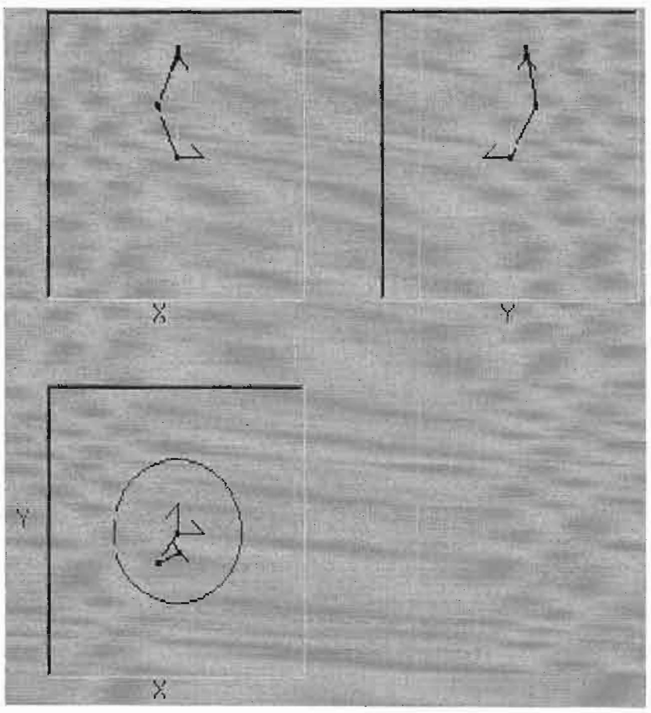
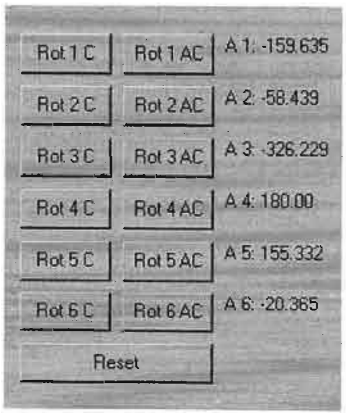


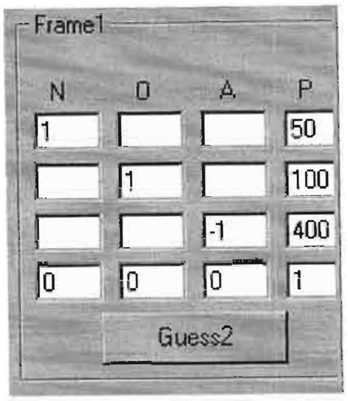
Figure 9.15. The three views of the robot arm.



A 1: -159.635
A 2: -58.439
A 3: -326.229
A 4: 180.00
A 5: 155.332
A 6: -20.365

Figure 9.16. Manual control window.

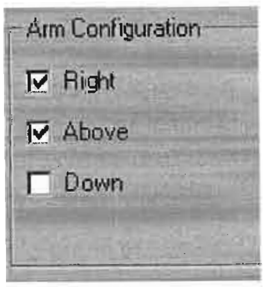
Figure 9.17 shows the end-effector coordinate orientation. The operator can set the final Cartesian coordinate of the end-effector as in column P. the matrix NOA refers to the orientation of the end-effector with respect to the coordinate frame.



N	O	A	P
1			50
	1		100
		-1	400
0	0	0	1

Figure 9.17. The end-effector position.

Figure 9.18 shows the arm configuration of the PUMA robot under control (ref. Chapter 7). The ability of the operator to set the arm configuration shows the modularity of the control software and how the software can be used on all Series 500 PUMA robots, irrespective of the arm configuration. The default setting for the PUMA 560 robot is shown in figure 9.17.



- [✓] Right
- [✓] Above
- [ ] Down

Figure 9.18. The arm configuration of the PUMA robot.

Figure 9.19 shows the operator command functions available for the PUMA robot. At robot startup the robot arm must be initialised. The operator uses the mouse cursor and clicks the [Initialise Arm] button. This displays the PCL-832 initialisation form (ref. Figure 9.19). Once the robot arm is initialised, the operator has the option to load a preprogrammed routine, or to create a new routine. When the operator enters two coordinates for the end-effector to move between, any set amount of interpolated points in-between can be calculated. The more interpolated points the operator specifies, the smoother the robot motion becomes while increasing the processing requirements to solve the kinematic solution.

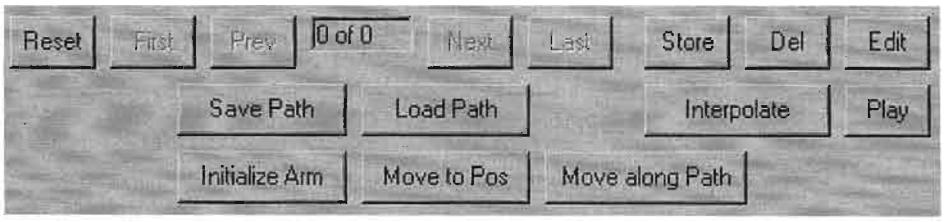
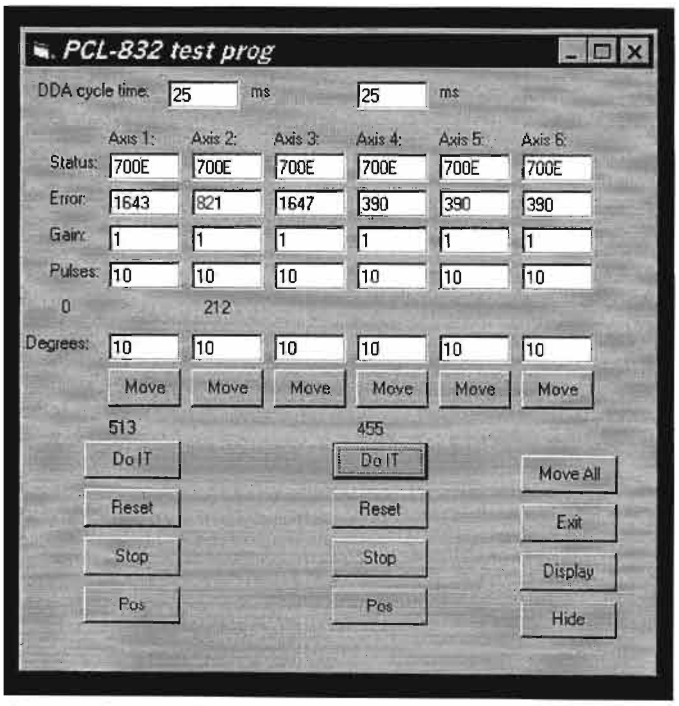


Figure 9.19. The operator command buttons.

Figure 9.20 shows the [*frmPCL832.frm*] initialisation form.



DDA cycle time: 25 ms 25 ms

	Axis 1:	Axis 2:	Axis 3:	Axis 4:	Axis 5:	Axis 6:
Status:	700E	700E	700E	700E	700E	700E
Error:	1643	821	1647	390	390	390
Gain:	1	1	1	1	1	1
Pulses:	10	10	10	10	10	10
Degrees:	10	10	10	10	10	10

0 212

513 455

Figure 9.20. The [*frmPCL832.frm*] initialisation form of the PUMA robot.

9.4 APPLICATIONS OF THE MECHATRONIC INTERNET CONTROL SYSTEM

The section presents two other applications of the operational procedures adopted by the modular mechatronic Internet control system. The aim of this section is to highlight the practical applications of the research presented.

9.4.1 Example 1 : Telecare and Telehealth

This section presents an example that discusses the application of the modular mechatronic Internet control system in the Telehealth and Telecare sectors. This example is based on research conducted by Prof. D. A. Bradley, Professor of Mechatronic Systems in the School of Science and Engineering at the University of Abertay Dundee.

Remote or tele-operated systems present a particular problem for the designer of human-machine interface in that they must not only provide information on the operator of the system itself, but also of its environment [Bradley et.al.]. This is particularly true for the telecare and telehealth environments. It is difficult enough for the human-machine interface when the operator is skilled, but in the case of online monitoring of the aged and frail the system becomes more complex. Not only must the monitoring and human-machine interaction be failsafe and intelligent, but also noninvasive.

The modular mechatronic Internet control system has applications in the design and implementation of frail-care monitoring devices. Once the monitoring devices are in place the required sensor information must be distributed on a widely accepted network system. The network must be capable of delivering the required information to the appropriate health care professionals and emergency services. Figure 9.21 shows the telecare system design strategy using the modular mechatronic design methodology.

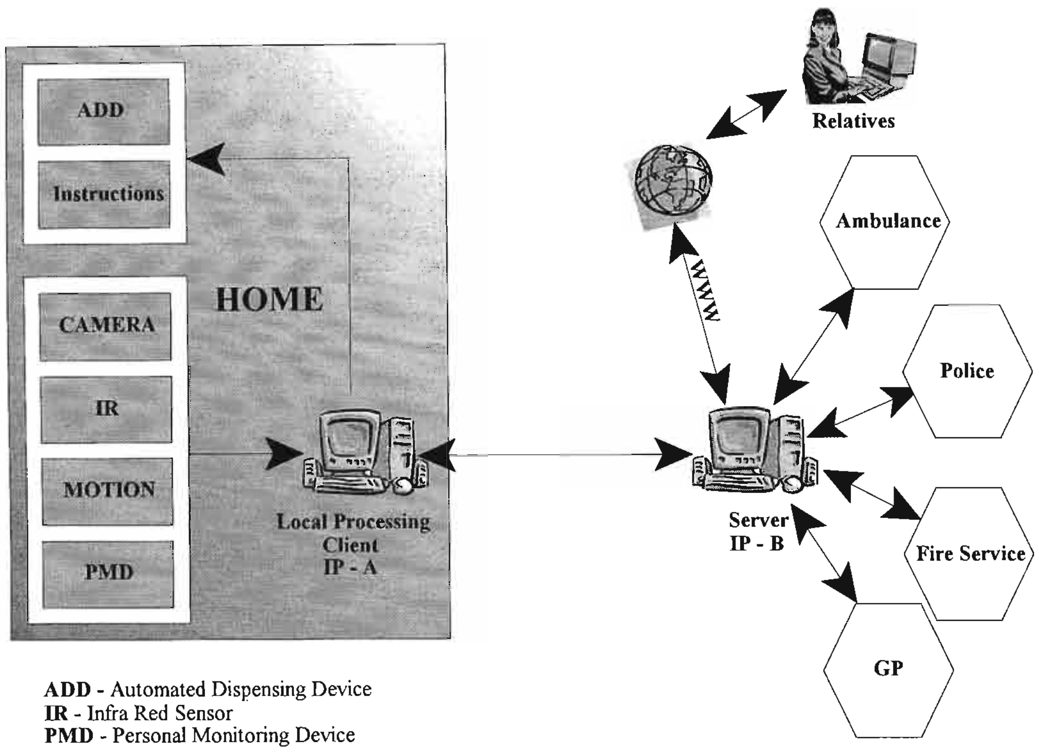


Figure 9.21. The telecare system design strategy.

The telecare system also allows relatives and friends to visually communicate with the house bound patients, using the mechatronic visual feedback module. The module can be modified to relay simple speech.

The telecare system shown in figure 9.21 has the following system functions.

- Lifestyle monitoring - Automatic emergency assistance detection.
- Emergency monitoring - Detecting possible unsafe situations.
- Security - Monitor and activate security system.
- Medical monitoring - Monitor medical needs and drug dispensing.
- GP consultations - Consult with virtual GP.
- Improved communication - Update database and communication with relatives.

The modular Internet monitoring system can be expanded to include more sensors and actuators. These actuators and sensors could control more house functions as the patient might not be mobile or capable of performing elementary household tasks.

9.4.2 Example 2 : The Networked House

Network technologies have started to invade the ordinary home- to carry telephone conversations, television, signals from surveillance cameras, commands for controlling appliances, and multimedia flow from the Internet (ref. Figure 9.22).

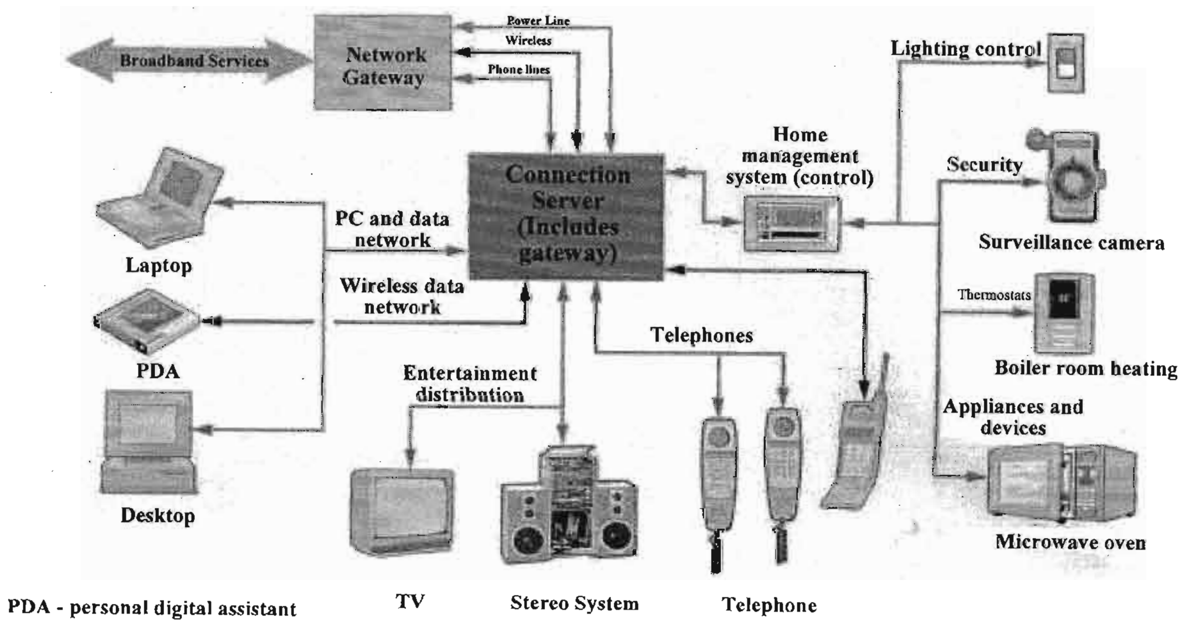


Figure 9.22. Networks in the home will distribute data and entertainment signals that are generated locally, as well as brought in from the Internet through a broadband connection. [Dutta-Roy]

With home networking it is also possible for electric utilities to remotely control and monitor the flow of electricity into individual homes. At the moment home networking falls into two main categories:

- Computer interconnection - accessing the Internet, and connecting multiple PCs with peripherals for communication and entertainment.
- Controlling items like lights, appliances, climate control systems and surveillance cameras.

Currently the best candidates for home-based networks are homes with two or more computers, using inexpensive Category 5 twisted pair network cables or voice-grade telephone wiring. This scheme accounts for about 17 million homes in the United States alone [Dutta-Roy]. To succeed home networks must be based on standard products operable with any form of media. For mass market appeal the, the networks have to be inexpensive, easy to install, and he software easy to configure and operate.

The modular mechatronic Internet control system makes a significant contribution towards a simplified networked environment. The backbone of the network once again relies on the TCP/IP protocols, for communication and connectivity. The PC and data networks are connected to the in-house Server, controlling the various peripherals in the home.

CHAPTER 10

10 CONCLUSION

The modern global manufacturing environment is characterised by product diversification and intense competition. The competitiveness of the global consumer market has resulted in a trend towards increased decentralisation of manufacturing processes with the increased customisation of products. The need for greater product customisation has resulted in tele-manufacturing systems being developed. Standard production-orientated manufacturing systems are not suited to the degree of product diversification necessitated by the trend towards increased customisation. Customisation requires the implementation of the latest computer- and Internet-based manufacturing technologies in a flexible manufacturing environment.

The field of Internet-based manufacturing and multimedia manufacturing environments have been an area of intense research during the last ten years. The result has been the enterprise integration of decentralised manufacturing operations, using multimedia capable computer networks. The modular design approach significantly simplified the implementation of the modular mechatronic CIM Internet-based control system. The effective Internet control system was based on a large-scale, integrated communications system across a variety of machine platforms. The modular mechatronic design methodology was integrated into the standardised Internet developmental control environment.

The thesis described six specific design objectives for modular mechatronic CIM control for Internet manufacturing and how these objectives were achieved.

1. Chapter 3 describes the design and development of the necessary power electronics required to operate the motion systems of the conveyer material handling system and the PC-based PUMA industrial robot. The *power modules* use lowcost electronic components, which are inexpensive and commercially available. The power electronic modules are flexible in their applications as they meet the power requirements of the CIM cell components. The LM12 driver module has a power rating of 150W and

operates within a broad voltage range, where speed control is crucial to system stability. The relay modules' power ratings are dependant upon the relay selected to power the required actuator. The relays are interchangeable and up-gradable as the systems power requirement's change. This provides a high degree of modularity to the actuator drive modules.

2. Chapter 4 describes the design and development of the *mechatronic feedback devices*, and the signal conditioning electronics required to monitor and control the motion systems of the CIM cell components. Mechatronic systems require sensors and measuring devices to acquire system information for data processing and information of the dynamic system changes. The feedback modules designed and investigated have applications in all areas of CIM control and monitoring.

3. Chapter 5 describes the Internet and network technologies used to communicate with and control the individual CIM components. *The network module* used the star topology. This network topology was already in use as part of the University network. The star network topology allows for the simple addition of CIM Clients, with no extra requirements from the network resources.

   The TCP *control protocol module* was used as the network communication protocol. The TCP protocol is a connection-based protocol. The data transmitted via this protocol is secure from becoming corrupt and always reaches its destination, even with changes in the network and bandwidth limitations. With the TCP protocol data buffering does not occur as the data is sent as one complete packet, as apposed to the UDP protocol which uses individual sends according to maximum data size. The limitation of the TCP protocol is that each connection between a Client controller and the Server require a WinSock connection with a varying port number. Each new Client connecting to the Server requires a modification and the addition of a WinSock communications proxy. If the Server side of the application is designed correctly, the additional WinSock proxies can be added as modular software components.

The *Server module* was used to communicate and control the individual CIM Client controllers. The Server used the File Transfer Protocol (FTP) and the Hyper Text Transfer Protocol (HTTP). These protocols contained the Web pages and control programmes required for the CIM control over the Internet.

The *visual feedback module* was designed to dynamically update the web page at four frames per second. This update time is ideal for high network bandwidth conditions. However, during periods of increased network congestion and activity the page updates more slowly. The Web Client is still able to monitor the CIM process, as the visual feedback module maintains system integrity.

4. Chapter 6 outlines the design of the PC-based controller for the conveyer material handling system. The controller was programmed to perform preset material handling routines while interacting with the Host controller. The control algorithm was designed to receive a single phrase command from the Host controller, allowing the primary machine intelligence to perform the task. This control strategy eliminated the additional system resources required.

5. A desktop computer was used to control the six degree of freedom PUMA 560 series robot. This PC-based controller was designed using lowcost PC plugin cards and low cost driver power modules and electronics. The robot used the kinematic solutions described in Chapter 7 for accurate end-effector positioning. Since the new robot controller was a computer-based technology, networking the robot with the other CIM components was achieved using the network technologies described in Chapter 5. The control signals between the robot Client and the Host controllers were only related to the new positional values of the robot end-effector. The robot receives the new positional command and is responsible for the machine intelligence for final end-effector positioning.

6. The PUMA robot and conveyer system were networked and controlled, using the

supervisory control strategy. The CIM components were networked via a network HUB to the Host controller and responded to direct TCP/IP control commands from the Web Client supervisory controller.

The Internet-based control system has been developed to control and enhance computer integrated manufacturing systems. The CIM control strategy detailed in this thesis has been developed as a PC-based technology using the *modular mechatronic* design methodology, and therefore represents an original and meaningful contribution to the fields of Internet manufacturing technologies and modular mechatronic computer integrated manufacturing systems.

10.1 INTERNET MANUFACTURING TECHNOLOGIES

Web-based manufacturing and Web teleoperations have applications not only in traditional manufacturing environments but also in entirely new areas made possible by the lack of a requirement for special equipment at the operator's end. These areas also include tele-medicine and training. The challenge with information technology software tools and interface design is to provide enough information to make interpretation easy while minimising transmitted data and maintaining a simple modular layout.

This was achieved by restricting operator control to an open-loop control structure. Minimising the time delay in communication and preventing the system instability problems associated with traditional closed loop control situations. There was a significant improvement of operator performance, when control was shared between the CIM cell controllers and the human operator. The new control strategy relied on the exchange of discreet system commands, rather than control algorithms. The supervisory control strategy allowed unskilled operators to control CIM components safely without having any knowledge of the primary control process.

The CIM conveyer system and PUMA robot were implemented as computer-based technologies. The advanced control algorithms required to sequence and coordinate the interaction of the CIM components to perform the standard routines were developed and coded in Microsoft Visual Basic 6 (VB6). The Web client graphical user interface (GUI) was developed using a combination of VBScript, HTML and JAVA. The development of the teleoperation control framework was detailed in Chapter 5. The functional capabilities of the developed Internet control structure introduced a high level of flexibility for product development and manufacturing.

10.2 MODULAR MECHATRONICS

The *modular mechatronic* design methodology optimised the development of the Internet-based CIM environment. The *modular mechatronic* system building blocks were successfully implemented in the control of two CIM part processes, and the rapid system development tool provided a practical design solution for the mechatronic system development. The standardisation of the design and control methodology simplified the manufacturing system while minimising the system complexity.

The proposed modular design approach, using the functional decomposition of the project into functional modules, was successfully developed. The functional modules were integrated across two varying system platforms, hence minimising the development time. The various project definitions were decomposed into independent sub-tasks according to the individual mechatronic functionality of each module. The success of the modular mechatronic design methodology was dependant on the computer-based integration of the various functional mechatronic modules. The *modular mechatronic* design methodology allowed the CIM control responsibilities to be organised in multiple levels, integrating Internet-based supervisory control and general system management.

The integration of the *modular mechatronic* system in a software domain was achieved by the development of the control algorithms that coordinated the interaction of the supervisory Web controller with the individual CIM components. Figure 10.1 shows the flow chart of the hierarchical CIM control structure integration of the modular sub-tasks.

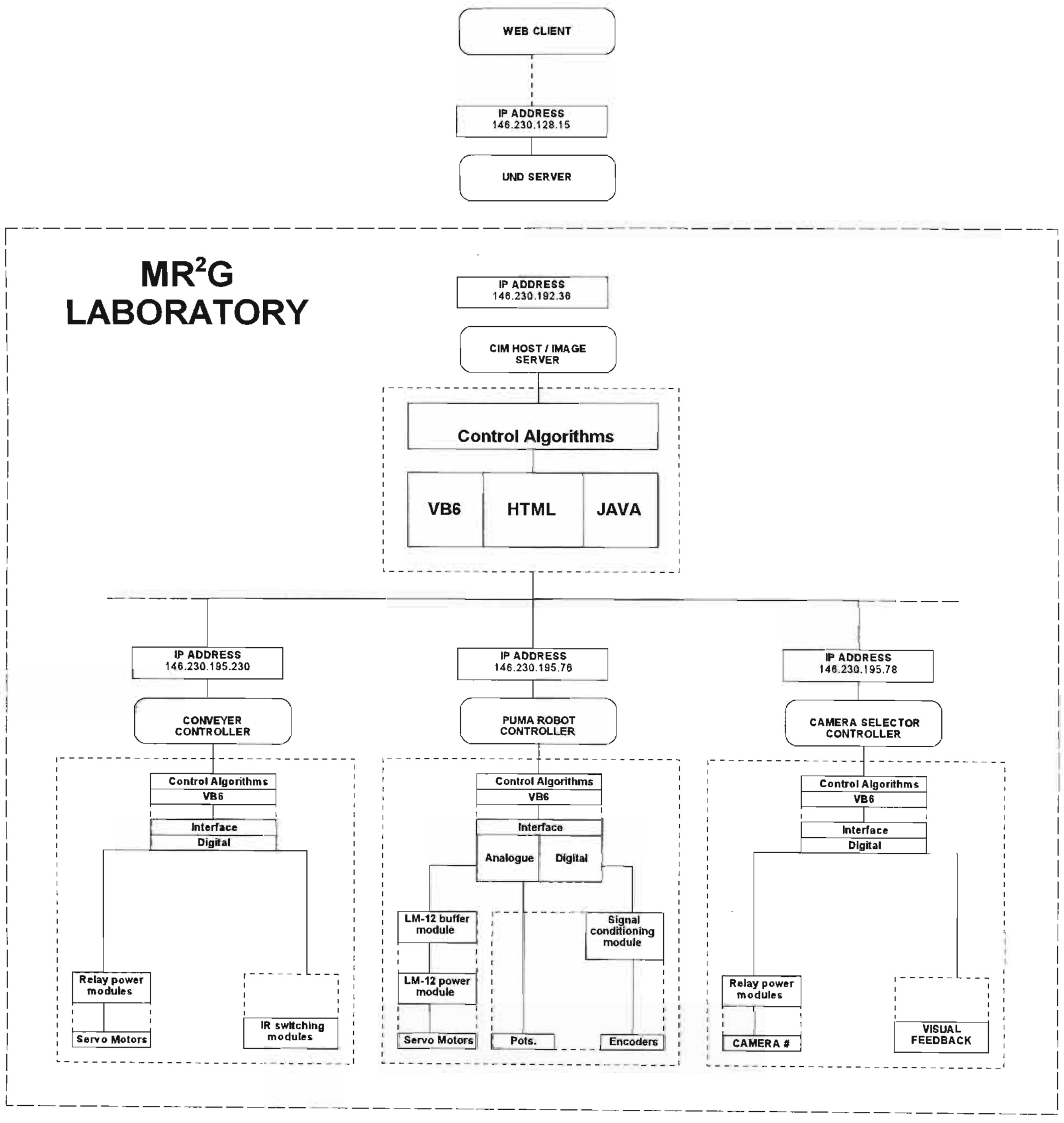


Figure 10.1. A flow chart showing the hierarchical CIM control structure integration of the modular mechatronic sub-tasks.

The *modular mechatronic* control for Internet manufacturing produced an efficient and effective solution to Computer Integrated Manufacturing processes. This approach allowed for a remote user to monitor and control a manufacturing process in real time over the Internet.

10.3 SUGGESTIONS FOR FURTHER WORK

This thesis embodies the first steps towards the practical implementation of the low-cost modular mechatronic CIM control system, using Internet technologies. Table 10.1 shows how all the computer-based CIM technologies can be controlled using the modular sub-tasks described in this thesis.

MODULAR CIM CONTROL FOR INTERNET MANUFACTURING						
Module / Component	**ROBOT**	**Conveyer**	**AS/RS**	**CMM**	**AGV**	**AVIA**
Relay Power Drive	✓	✓				✓
LM12 Power Drive	✓		✓	✓	✓	
Potentiometers	✓					
IR Circuits		✓	✓		✓	✓
Encoders	✓		✓	✓		
Buffer Circuit	✓		✓	✓	✓	
Encoder Signal	✓		✓	✓		
PC-30GA	✓				✓	✓
PCL-836		✓				
PCL-832 (servo card)	✓		✓	✓		
Network	✓	✓	✓	✓	✓	✓
Control Protocol	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP	TCP/IP
Server	✓	✓	✓	✓	✓	✓
Visual Feedback	✓	✓	✓	✓	✓	✓

Table 10.1. The Modular Mechatronic CIM Sub-Tasks.

The main steps that follow on from the work in this thesis are thus:

a) Networking all the CIM Client controllers to complete the CIM cell (ref. Table 10.1).

b) The Ethernet network used for the communication between the CIM Server and the CIM Clients can be modified to use wireless network adapters. The wireless Internet-based multimedia system can be based on the standard IEEE 802.11b Ethernet products. These include transceivers implementing a carrier radio frequency of 2.4 GHz. The maximum physical layer throughput of the system is 11 Mbits/sec.

c) The development of a high level computer system that should be able to create a data base or defect log and report defect rates back to the upper level manufacturing execution system.

d) The implementation of advanced camera tracking systems to follow machined components around the CIM.

e) The development of a generic view-planning algorithm that must be able to select the optimum set of view points required for manufacturing and material handling processes.

f) The use of wireless cameras in conjunction with wireless Ethernet network adapters to enhance the flexibility of the Computer Integrated Manufacturing environment.

GLOSSARY

A

AC: Area Controller.

Actuator: A device that transforms a signal into a physical effect such as motion or force (or a combination of the two). The same concept applies for other physical effects such as heating, fluid motion, etc.

Acquisition Time: This term relates to sampling A/Ds which utilize a track/hold amplifier on the input to acquire and hold the analog input signal. Acquisition time is the time required by the T/H amplifier to settle to its final value after it is placed in the track module.

Amplifier: A device which draws power from a source other than the input signal and which produces as an output an enlarged reproduction of the essential features of its input.

Analog Ground: In high-speed acquisition applications, system ground is generally physically separated into analog and digital grounds in an attempt to suppress digital switching noise and minimize its effect on noise-sensitive analog signal processing circuitry. Input signal conditioners, amplifiers, references, and A/D converters are usually connected to analog ground.

Analog-to-Digital Converter (A/D or ADC): A device or circuit that outputs a binary number corresponding to an analog signal level at the input.

ANSI: American National Standards Institute.

ASCII: American Standard Code for Information Interchange. A seven or eight bit code used to represent alphanumeric characters. It is the standard code used for communications between data processing systems and associated equipment.

ASP: Active Server Page.

Autonomous vehicle: A vehicle that can substantially control its own motion, usually (but not always) deciding where it is to travel. Mostly these are classified as robots, but often not, as in 'guided missile' for example. Typical examples: unmanned aerial vehicles (UAV), unmanned underwater vehicle (UUV). A remotely operated vehicle (ROV) is usually not autonomous because its motion is remotely controlled by a human operator. UAV's and UUV's are not considered autonomous if they need to be controlled at all times by a human operator. However, some may be autonomous for part of the time, and controlled by a human operator at other times.

B

Backbone: A high-speed line or series of connections that forms a major pathway within a network. The term is relative as a backbone in a small network will likely be much smaller than many non-backbone lines in a large network.

Baud: In common usage the baud rate of a modem is how many bits it can send or receive per second. Technically, baud is the number of times per second that the carrier signal shifts value - for example a 1200 bit-per-second modem actually runs at 300 baud, but it moves 4 bits per baud (4 x 300= 1200 bits per second).

Baud Rate: The speed at which data is transmitted. Measured in symbols per second. This is not the same as bits-per-second since each symbol can carry several bits of information.

Binary: Refers to base 2 numbering system, in which the only allowable digits are 0 and 1. Pertaining to a condition that has only two possible values or states.

Bit Rate: The rate of transfer of information necessary to ensure satisfactory reproduction of the information

Bit: Acronym for binary digit. The smallest unit of computer information, it is either a binary 0 or 1.

Bitmap: An image is displayed on the screen as a collection of tiny squares called pixels, which together form a pattern. Each pixel in the image corresponds with one or more bits; the number of bits per pixel determines how many shades of gray or colours can be displayed

Bluetooth: A standardized, short-range radio link between information appliances. Named after a 10th century Danish king credited with uniting Denmark and Norway. It promises easy connectivity between computers, peripherals, PDAs and other devices.

BMP: A Windows format for a bitmapped graphics file.

BPS: Bits per second.

Byte: The representation of a character in binary. Eight bits in length.

C

Cache: A temporary storage area for frequently-accessed or recently-accessed data. Having certain data stored in a cache speeds up the operation of the computer. There are two kinds of cache: internal (or memory cache) and external (or disk cache). Internal cache is built into the processor, and external cache is on the motherboard. When an item is called for, the computer first checks the internal cache, then the external cache, and finally the slower main storage.

CAD: Computer Aided Design.

CADD: Computer Aided Design and Drafting.

CAE: Computer Aided Engineering.

CAP: Computer Aided Planning.

CAPP: Computer Aided Process Planning.

CAQ: Computer Aided Quality Control.

CC: Cell Controller.

Cell: An ATM packet that is 53 bytes in length with a 5 byte header and 48 byte payload.

CIM: Computer Integrated Manufacturing.

Client-Server Network: A network that uses a central computer (server) to store data that is accessed from other computers on the network (clients).

Clock: The device that generates periodic signals for synchronization.

Codec: An abbreviation for Coder-Decoder. An analog-to-digital (A/D) and digital-to-analog (D/A) converter for translating the signals from the outside world to digital, and back again.

Computer vision: Use of camera to transmit one or more images to a computer which performs some analysis on the image to obtain information, or to present the processed image to a human operator. (see also "image sensor")

Control system: As most mechatronic systems involve control systems of one kind or another, only use this keyword where the control details are discussed.

Conveyor: A device that transfers people, material or parts from one place to another.

Counts: The number of time intervals counted by the dual-slope A/D converter and displayed as the reading of the panel metre, before addition of the decimal point.

CPU: Central processing unit. The part of the computer that contains the circuits that control and perform the execution of computer instructions.

Cycle Time: The time usually expressed in seconds for a controller to complete one on/off cycle.

D

DAS: Data Acquisition System.

DC: Direct current; an electric current flowing in one direction only and substantially constant in value.

Design: Use this keyword where there is some useful discussion or description of the design process. All man-made objects are 'designed' so this keyword could easily be mis-used.

Differential Input: A signal-input circuit where SIG LO and SIG HI are electrically floating with respect to ANALOG GND (METRE GND, which is normally tied to DIG GND). This allows the measurement of the voltage difference between two signals tied to the same ground and provides superior common-mode noise rejection.

Digital Filtering: The process of smoothing, or removing noise from a signal via mathematical functions that are performed on the digital data stream.

Digital Output: An output signal which represents the size of an input in the form of a series of discrete quantities.

Digital Signal Processor (DSP): This technology, when used in conjunction with mixed-signal devices and embedded software, is referred to as a DSP Solution, and it collects, processes, compresses, transmits and displays analog and digital data.

Digital-to-Analog Converter (D/A or DAC) A device or circuit to convert a digital value to an analog signal

Dithering: The technique of adding controlled amounts of noise to a signal to improve overall system loop control, or to smear quantizing error in an A/D convertor application.

DLL: Dynamic Linked Library

DMA: Acronym direct memory access. A high speed data storage mode of the IBM PC.

DNC: Direct Numerical Control.

DNS: Domain Name System. A mechanism used in the Internet for translating names of host computers into addresses. The DNS also allows host computers not directly on the Internet to have registered names in the same style.

Domain Name: The unique name that identifies an Internet site. Domain Names always have 2 or more parts, separated by dots. The part on the left is the most specific, and the part on the right is the most general. A given machine may have more than one Domain Name but a given Domain Name points to only one machine. For example, the domain names: matisse.netmail.matisse.networkshop.matisse.netcan all refer to the same machine, but each domain name can refer to no more than one machine. Usually, all of the machines on a given Network will have the same thing as the right-hand portion of their Domain Names (matisse.net in the examples above). It is also possible for a Domain Name to exist but not be connected to an actual machine. This is often done so that a group or business can have an Internet e-mail address without having to establish a real Internet site. In these cases, some real Internet machine must handle the mail on behalf of the listed Domain Name.

DRAM: Dynamic Random Access Memory.

DSP: Digital signal processing or digital signal processor.

E

Electric machine: Motor, transformer, solenoid, relay or other form of electro-mechanical device, sensor or actuator.

Email - (Electronic Mail): Messages, usually text, sent from one person to another via computer. E-mail can also be sent automatically to a large number of addresses.

Ethernet: Xerox standard networking protocol used in local area networks, often connecting dissimilar devices.

F

FAS: Flexible Manufacturing Assembly.

File: A set of related records or data treated as a unit.

Fire Wall: A combination of hardware and software that separates a Network into two or more parts for security purposes.

Firmware: Software that is embedded in a hardware device that allows reading and executing the software, but does not allow modification, e.g., writing or deleting data by an end user.

FMC: Flexible Manufacturing Cell.

FMS: Flexible Manufacturing Systems.

Frequency Response: The range of frequencies over which the transducer voltage output will follow the sinusoidally varying mechanical input within specified limits.

FTP - (File Transfer Protocol): A very common method of moving files between two Internet sites. FTP is a way to login to another Internet site for the purposes of retrieving and/or sending files. There are many Internet sites that have established publicly accessible repositories of material that can be obtained using FTP, by logging in using the account name "anonymous", thus these sites are called "anonymous FTP servers". FTP was invented and in wide use long before the advent of the World Wide Web and originally was always used from a text-only interface.

Fuzzy: A control method invented by Zadeh in 1965 in which the control actions can be described by simple rules, for example: If error is "low and positive" set speed to "slow and negative". Continuous membership functions define the relationship between a "crisp" continuous variable and membership of certain "fuzzy" input or output classes.

G

Gain: The amount of amplification used in an electrical circuit. Gain is usually measured in decibels, but it can also be expressed as the ratio of output power to input power.

Gateway: The technical meaning is a hardware or software set-up that translates between two dissimilar protocols, for example America Online has a gateway that translates between its internal, proprietary e-mail format and Internet e-mail format. Another, sloppier meaning of gateway is to describe any mechanism for providing access to another system, e.g. AOL might be called a gateway to the Internet.

GIF - (Graphic Interchange Format): A common format for image files, especially suitable for images containing large areas of the same colour. GIF format files of simple images are often smaller than the same file would be if stored in JPEG format, but GIF format does not store photographic images as well as JPEG. GIF files are limited to 256 colours, but allow for animated images.

Gigabyte: 1000 or 1024 Megabytes, depending on who is measuring.

Graphics: Use of visual communication, usually on operator displays, typically with computer generated images.

Ground: 1. The electrical neutral line having the same potential as the surrounding earth. 2. The negative side of DC power supply. 3. Reference point for an electrical system.

GUI: Graphical user interface.

H

Handshake: An interface procedure that is based on status/data signals that assure orderly data transfer as opposed to asynchronous exchange.

Header: The portion of a packet, preceding the actual data, containing source and destination addresses and error-checking fields.

HTML - (HyperText Markup Language): The coding language used to create Hypertext documents for use on the World Wide Web. HTML looks a lot like old-fashioned typesetting code, where you surround a block of text with codes that indicate how it should appear. The "hyper" in Hypertext comes from the fact that in HTML you can specify that a block of text, or an image, is linked to another file on the Internet. HTML files are meant to be viewed using a "Web Browser". HTML is loosely based on a more comprehensive system for markup called SGML.

HTTP - (HyperText Transfer Protocol): The protocol for moving hypertext files across the Internet. Requires a HTTP client program on one end, and an HTTP server program on the other end. HTTP is the most important protocol used in the World Wide Web (WWW).

Hub: A central node in a star network to which all other nodes are connected by means of point-to-point communications links.

I

IANA: Internet Assigned Number Authority

ICANN : Internet Corporation for Assigned Names and Numbers

Icon: A graphic functional symbol display. A graphic representation of a function or functions to be performed by the computer

Image Processing: Enhancing an image or extracting information or features from an image.

Image Sensor: A device used to obtain images, for example a camera. May also refer to the VLSI chip used to convert an image to electronic form. Also used for special applications, for example an endoscope where obtaining a useful image is not simple.

IMAP - (Internet Message Access Protocol): IMAP is gradually replacing POP as the main protocol used by email clients in communicating with email servers. Using IMAP an email client program can not only retrieve email but can also manipulate message stored on the server, without having to actually retrieve the messages. So messages can be deleted, have their status changed, multiple mail boxes can be managed, etc. IMAP is defined in RFC 2060

Internet Protocol (IP): The network layer protocol for the Internet. It is the datagram protocol defined by RFC 791.

Internet: The Internet is a network of networks, linking computers to computers by speaking the same language called TCP/IP protocol. Each computer runs software to provide or serve information and/or to access and view information. The Internet includes a variety of electronic services such as electronic mail (e-mail), Telnet (remote login), FTP (File Transfer Protocol for downloading or uploading of files), Gopher (an early, text-only method for accessing Internet documents), and the World Wide Web. The Internet was originally developed for the United States military, and then became used for government, academic, and commercial research and communications.

IP Number - (Internet Protocol Number): Sometimes called a dotted quad. A unique number consisting of 4 parts separated by dots, e.g. 165.113.245.2Every machine that is on the Internet has a unique IP number - if a machine does not have an IP number, it is not really on the Internet. Many machines (especially servers) also have one or more Domain Names that are easier for people to remember.

ISA: Industry Standard Architecture (PC-AT Bus) or Instrument Society of America.

ISDN: Integrated Services Digital Network: A single communications vehicle that supports all forms of signal traffic-low and medium-speed data, audio, and video--across a standardized interface and on a single hardware platform. A communications network intended to carry digitized voice and data multiplexed into the public telephone network

ISO: International Standards Organization.

ISP - (Internet Service Provider): An institution that provides access to the Internet in some form, usually for money.

J

Java: Java is a network-friendly programming language invented by Sun Microsystems. Java is often used to build large, complex systems that involve several different computers interacting across networks, for example transaction processing systems. Java is also becoming popular for creating programs that run in small electronic devices, such as mobile telephones. A very common use of Java is to create programs that can be safely downloaded to your computer through the Internet and immediately run without fear of viruses or other harm to your computer or files. Using small Java programs (called "Applets"), Web pages can include functions such as animations, calculators, and other fancy tricks.

JavaScript: JavaScript is a programming language that is mostly used in web pages, usually to add features that make the web page more interactive. When JavaScript is included in an HTML file it relies upon the browser to interpret the JavaScript. When JavaScript is combined with Cascading Style Sheets(CSS), and later versions of HTML (4.0 and later) the result is often called DHTML.

JPEG (Joint Photographic Experts Group): Industry standard for compressing images. This format provides lossy compression (you lose sharpness from the original) by dividing the image into tiny pixel blocks, which are halved over and over until an adequate compression ratio is achieved.

K

Kilobyte: A thousand bytes. Actually, usually 1024 (210) bytes.

Kinematics: Study of motion of objects without regard to the causes of this motion.

L

Lag: A time delay that occurs between the output of a signal and the response of the instrument to which the signal is sent.

LAN: Local Area Network. A network that takes advantage of the proximity of computers to offer relatively efficient, higher-speed communications than long-haul or wide-area networks.

LAP: Link Access Protocol.

LED: Light Emitting Diode

M

Mechatronics: The introduction of electronic controls into mechanical components.

Megabyte: A million bytes. Technically, 1024 kilobytes. (or 1,048,576 bytes)

MIME - (Multipurpose Internet Mail Extension): Originally a standard for defining the types of files attached to standard Internet mail messages. The MIME standard has come to be used in many situations where one computer program needs to communicate with another program about what kind of file is being sent. For example, HTML files have a MIME-type of text/html, JPEG files are image/jpeg, etc.

Modem: Modulator/Demodulator. A device that transforms digital signals into audio tones for transmission over telephone lines, and does the reverse for reception.

Modular Mechatronics: Is a design approach that requires the decomposition of a project into separate modules, identifiable by their individual mechatronic functionality.

MOSFET: Metal Oxide Semiconductor Field Effect Transistor. A class of voltage driven devices that does not require the large drive currents of bipolar devices. Used in products that require large current levels to be switched electronically.

Motherboard: The pc board of a computer that contains the bus lines and edge connectors to accommodate other boards in the system. In a microcomputer, the motherboard contains the microprocessor and connectors for expansion boards.

MPEG (Motion Picture Experts Group): An industry standard for compressing video and audio. MPEG I, used in video CDs, provides a standard image of 352x240 dots per inch, 30 frames per second, 15-bit colour and CD-quality sound. MPEG II is an emerging standard for full broadcast-quality video.

MRP: Material Resource Planning.

N

Netscape: A WWW Browser and the name of a company. The Netscape (tm) browser was originally based on the Mosaic program developed at the National Centre for Supercomputing Applications (NCSA).

Network: A group of computers that are connected to each other by communications lines to share information and resources.

Nibble: One half of a byte.

NIC - (Network Information Centre): Generally, any office that handles information for a network. The most famous of these on the Internet was the InterNIC, which was where most new domain names were registered until that process was decentralized to a number of private companies.

NNTP – (Network News Transport Protocol): The protocol used by client and server software to carry USENET postings back and forth over a TCP/IP network. If you are using any of the more common software such as Netscape, Nuntius, Internet Explorer, etc. to participate in newsgroups then you are benefiting from an NNTP connection.

O

Op Amp: The Operational Amplifier is a general purpose integrated circuit used as a basic building block for the implementation of linear functions. Op amps form the "front end" or sensory apparatus of many electronics systems, capturing weak signals emanating from the real world and amplifying and filtering them for processing. A typical use of an op amp is to take a smaller signal and increase it by a specific amount to become a larger signal.

P

Packet Switching: The method used to move data around on the Internet. In packet switching all the data coming out of a machine is broken up into chunks, each chunk has the address of where it came from and where it is going. This enables chunks of data from many different sources to co-mingle on the same lines, and be sorted and directed along different routes by special machines along the way. This way many people can use the same lines at the same time. You might think of several caravans of trucks all using the same road system. to carry materials.

PC Card: A computer device packaged in a small card approximately the size of a credit card and conforming to the PCMCIA standard.

PCMCIA: Portable Computer Memory Card International Association

Precision: Precision is reproducibility. Saying "These measurements are precise" is the same as saying, "The same measurement was repeated several times, and the measurements were all very close to one another". Don’’t confuse precision with accuracy.

Protocol: A formal definition that describes how data is to be exchanged.

R

Resolution: The smallest signal increment that can be detected by a measurement system, expressed in bits, proportions, or as percentage of full scale reading; resolution in a video system is the amount of detail in a graphic image.

Router: A special-purpose computer (or software package) that handles the connection between 2 or more Packet-Switched networks. Routers spend all their time looking at the source and destination addresses of the packets passing through them and deciding which route to send them on.

S

S-video: Separate luminance and chrominance

Sampling Rate: The rate at which an analog signal is sampled for conversion to and from the digital domain. The sampling rate is measured as the number of samples per unit of time.

SCADA: Supervisory control and data-acquisition systems

Signal Conditioning: To process the form or mode of a signal so as to make it intelligible to, or compatible with, a given device, including such manipulation as pulse shaping, pulse clipping, compensating, digitizing, and linearizing.

Signal Ground: The common return or reference point for analog signals.

Signal: An electrical transmittance (either input or output) that conveys information.

T

T-1: A leased-line connection capable of carrying data at 1,544,000 bits-per-second. At maximum theoretical capacity, a T-1 line could move a megabyte in less than 10 seconds. That is still not fast enough for full-screen, full-motion video, for which you need at least 10,000,000 bits-per-second. T-1 lines are commonly used to connect large LANs to the internet.

T-3: A leased-line connection capable of carrying data at 44,736,000 bits-per-second. This is more than enough to do full-screen, full-motion video.

TCP/IP: Transmission Control Protocol/Internet Protocol. This is a common shorthand which refers to the suite of application and transport protocols which run over IP. These include FTP, Telnet, SMTP, and UDP (a transport layer protocol).

TELNET: The Internet standard protocol for remote terminal connection service. Telnet allows a user at one site to interact with a remote timesharing system at another site as if the user ′s terminal was connected directly to the remote computer.

TTL: Transistor-to-transistor logic. A form of solid state logic which uses only transistors to form the logic gates.

Twisted Pair: Two insulated wires, usually made from copper, that are twisted in a regular, six turns per inch spiral pattern used to connect most telephones. Also used as a medium by several local area networks.

U

UDP - (User Datagram Protocol): One of the protocols for data transfer that is part of the TCP/IP suite of protocols. UDP is a "stateless" protocol in that UDP makes no provision for acknowledgement of packets received.

User Interface: Related to *ergonomics* and *human computer interaction*. Discussion of the displays and controls that enable a human to interact with a machine or process.

UUENCODE - (Unix to Unix Encoding): A method for converting files from Binary to ASCII (text) so that they can be sent across the Internet via e-mail.

W

Web: World Wide Web - use keyword where web technology forms a specific part of a process. Be careful to distinguish processes that rely on a specific web technology (such as a browser, use of HTML etc.) from those that simply use the internet for communication, but do not use web-specific technologies. Sockets, for example, are used by web applications but are an internet technology: they are not specific to web applications.

Wide Area Network (WAN): A group of computer networks connected together over long distances. The Internet is a WAN.

Window: In computer graphics, a defined area in a system not bounded by any limits; unlimited "space" in graphics.

Word: Number of bits treated as a single unit by the CPU. In an 8-bit machine, the word length is 8 bits; in a sixteen bit machine, it is 16 bits.

WWW: World Wide Web.

REFERENCES

[Allegri]	Allegri T.H., *Materials handling : Principles and Practice.* Van Nostrand Reinhold Company Limited, London, UK, 1984
[Auslander et.al.]	Auslander D.M. and Kempf C.J., *Mechatronics, Mechanical System Interfacing*, Prentice Hall, New Jersey, USA, 1996
[Backes]	Backes P., *Pathfinder Sojourner Rover Simulation Web Page:* *http://mars.graham.com/wits/*
[Backes et.al.]	Backes P., Tao K.S., Tharp G.K., *Mars Pathfinder Mission Internet - Based Operation Using WITS*, IEEE Int. Conf. Robotics and Automation, pp 284 - 291, May 1998
[Barnes]	Barnes J. Wesley., *Statistical Analysis for Engineers and Scientists: A Computer-based Approach*, McGraw-Hill, Highstown, New Jersey, USA, 1994
[Bercea]	Bercea N., *The Attributes of a New Generation Tool for the Calculation of Screw Assemblies*, 9th International DAAM Symposium, 1(1), Vienna, 1998, pp 039-040
[Bolton]	Bolton W., *Mechatronics : Electrical Control Systems in Mechanical Engineering*, Addison Wesley Longman Limited, Essex, England, 1995
[Bradley et.al.]	Bradley D., Seward D., Dawson S., Burge S., *Mechatronics and the Design of Intelligent Machines and Systems*, Stanley Thornes (Publishers) Ltd, United Kingdom, 2000

[Chamiak]	Chamiak E., *Introduction to Artificial Intelligence*, Wiley, New York, 1986
[Chen]	Chen C., *Fuzzy Logic and Neural Network Handbook*, IEEE Press, McGraw-Hill, USA, 1996
[Counsell et.al.]	Counsell J., Porter I., Dawson D., Duffy M,. *Schemebuilder: Computer Aided Knowledge Based Design of Mechatronic Systems*, Assembly and Automation, Vol. 19, No. 2, pp 129 - 138, 1999
[Craig]	Craig K., *is Anything Really New in Mechatronics Education?*, IEEE Robotics and Automation, Vol. 8, No. 2, pp 12 - 19, June 2001
[Dalton et.al.]	Dalton B., Taylor K., *Distributed Robotics over the Internet*, IEEE Robotics and Automation, Vol. 7, No. 2, pp 22 - 27, June 2000
[Demas]	Demas J.N. and Demas S.E., *Interfacing and Scientific Computing on Personal Computers*, Allyn and Bacon, Massachusetts, USA, 1990
[Deitel et.al.]	Deitel H.M, Deitel P.J. and Nieto T.R., *Visual Basic 6 How to Program*, Prentice Hall Inc., New Jersey, USA, 1999
[Dutta-Roy]	Dutta-Roy, A., *Networks for Homes*, IEEE Spectrum, Vol. 36, No. 12, pp 26 - 33, December 1999
[Erden et.al.]	Erden Z., Erkmen A. and Erden A., *Modelling Job-resource Interactions in Mechatronics Design Process by Petri Nets*, Proceedings of the International Conference on Engineering Design, ICED '97, Tampere, 1997

[Ferretti et.al.]	Ferretti G., Filippi S., Maffezzoni C., Magnani G., Rocco P., *Modular Dynamic Virtual-Reality Modelling of Robotic Systems*, IEEE Robotics and Automation, Vol, 6, No. 4, pp 13 - 23, December 1999
[Fielding et.al.]	Fielding E.R., Illos E.D., *P.C. Based Robotic Control System*, University of Witwatersrand, South Africa, 1995.
[FU et.al.]	FU K.S., Gonzalez R.C., CSG. Lee, *Robotics, Control, Sensing, Vision and Intelligence,* McGraw-Hill, London,1994
[Groover]	Groover M.P, *Automation, Production Systems, and Computer Integrated Manufacturing*, Prentice Hall International, 1987
[Gunasekeran]	Gunasekeran S., *Agile Manufacturing: A Framework for Research and Development*, International Journal on Production Economics, Vol. 62, No. 1, pp 87-105, 1999
[Harashima]	Harashima F., *Recent Advances of Mechatronics*, Proceedings of the IEEE International Symposium on Industrial Electronics, ISIE '96, Vol. 1. pp 1-5, 1996
[Haykin]	Haykin S., *Neural Networks : A Comprehensive Foundation*, Macmillan, USA, 1994
[Histand et.al.]	Histand M.B., Alciatore D.g., *Introduction to Mechatronics and Measurement Systems*, McGraw-Hill, USA, 1999
[Honekamp et.al.]	Honekamp U., Stolpe R., Neumann R. and Luckel J., *Structuring Approach for Complex Mechatronic Systems*, Proceedings of the 30th ISATA Conference, Florence, Italy, 1997

[Hu et.al.]	Hu H., Yu L., Tsui P.W., Zhou Q., *Internet-based Robotic System for Teleoperation*, Assembly and Automation, Vol. 21, No. 2, pp 143 - 151, 2001
[Huang]	Huang Chun-Che, *Overview of Modular Product Development*, Proceedings of the National Science Council, Republic of China, Part A: Physical Science and Engineering, Vol. 24, No. 3, pp 149-165, 2000
[Isermann]	Isermann R., *Modelling and Design Methodology for Mechatronic Systems*, IEEE/ASME Transactions on Mechatronics, Vol. 1, No. 1, pp 10-15, 1996
[Johnson]	Johnson, G.W., *LabView Graphical Programming, Practical Applications in Instrumentation and Control*, R R Donnelley and Sons Company, USA, 1994
[Kalpakjian et.al.]	Kalpakjian S., Schmid S.R., *Manufacturing Engineering and Technology*, 4th Edition, Prentice-Hall, 2001
[Kaplan]	Kaplan G., *Industrial Electronics*, IEEE Spectrum, Vol.31, No. 1, pp 74-76, 1994
[KUO]	Kuo, B., C., *Automatic Control Systems*, Fourth Edition, Prentice-Hall, USA, 1982
[McGarry]	McGarry S.L, *Acceptable Quality Levels Are Not Acceptable Any-more*, in Proceedings of Autofact, Anaheim, CA, pp 15-1 - 15-13, 1984

[McGee]	McGee M., *Web pages: a programmer's guide*, Microsoft Press, Redmond, Washington, 1997
[McKinney]	McKinney B., *Hardcore Visual Basic : 2nd Edition*, Microsoft Press, Wahington, USA, 1997
[NODA]	NODA A., *Manufacturing Industries in the Internet Era*, Proceedings of 14th International Conference on Computer-Aided Production Engineering, 1(1), Tokyo, 1998, 31-36
[Pemberton]	Pemberton A.W.,*Plant Layout and Materials Handling*, Mcmillian Press, London, 1984
[Pennebaker]	Pennebaker W.B. and Mitchell J.l., *JPEG Still Image Data Compression Standard*, Chapman & Hall, London, UK, 1993
[Parrish]	Parrish D., *Flexible Manufacturing*, Butterworth-Heinemann, Oxford, 1993
[Pomezny]	Pomezny L., *Visualisation of Technological Processes on Internet*, 9th International DAAM Symposium, 1(2), Vienna, 1998, 401-402
[Potgieter 1]	Potgieter J. and Bright G. *Robotic Internet System for CIM Processes*, SA Mechanical Engineer, South Africa, May 2000, pp 23 - 25
[Potgieter 2]	Potgieter J. and Bright G. *PC-Based Mechatronic Robotic Plug and Play System for Part Assembly Operations*, IEEE International Symposium on Industrial Electronics, Pretoria, South Africa, July, 1998, pp. 426 - 429

[Potgieter 3] Potgieter J. and Bright G. *PC-Based Modular Robotic System for Part Assembly Operations*, IASTED International Conference on Robotics and Manufacturing, Banff, Canada, July 1998, pp. 86 - 88

[Potgieter 4] Potgieter J. and Bright G. *Operating System for part Assembly Operations using PC-Based Plug and Play Mechatronic Technology*, 9th DAAAM International Symposium on Intelligent Manufacturing, Automation and Networking, Technical University Cluj-Napoca, Romania, October, 1998, pp. 73 - 76

[Potgieter 5] Potgieter J. and Bright G. *Flexible PC-Based Modular Mechatronic Operating Systems for Computer Integrated Manufacturing*, The 15th ISPE/IEE International Conference on CAD/CAM, Robotics and Factories of the Future, Aguas de Lindoia, SP, Brazil, August, 1999, pp. RF2-1 - RF2-4

[Potgieter 6] Potgieter J. and Bright G. *Mechatronic Modular Robotic Internet Based Control System for Computer Integrated Manufacturing Processes*, 7th IASTED International Robotics and Applications 2000, Honolulu, Hawaii, USA, August 2000

[Potgieter 7] Potgieter J. and Bright G. *Mechatronic Internet Based Control System for Computer Integrated Manufacturing Processes*, International Conference on Competitive Manufacturing, COMA'01, Stellenbosch, South Africa, February 2001

[Potgieter 8] Potgieter J. and Bright G. *Integrated Mechatronic Control Approach for Global Manufacturing Enterprises*, IASTED International Conference on Robotics and Manufacturing (RM2001)

[Potgieter 9]	Potgieter J. and Bright G. *Mechatronic Control Approach for Global Internet Manufacturing*, The 17th ISPE/IEE International Conference on CAD/CAM, Robotics and Factories of the Future, Durban, South Africa, July 2001
[Rahman et.al.]	Rahman SM., Sarker N., *Internet for Higher Productivity in Manufacturing*, Proceedings of the 4th International Conference on Computer Integrated Manufacturing, 1(1), Tokyo, 1998, 269-278
[Rehg]	Rehg J. A., *Introduction to Robotics in CIM systems*, Prentice Hall, New Jersey, 1997
[Reselman et.al.]	Reselman B. and Peasley R., *Practical Visual Basic 6*, QUE, MacMillan Computer Publishing, Indiana, USA, 1999
[Schwarz et.al.]	Schwarz S.E. and Oldham W.G., *ELECTRICAL ENGINEERING : An Introduction*, 2nd Edition, Oxford University Press, New York, 1993
[SGS-Thompson]	SGS-Thompson, *Power Linear Actuators, Databook*, 1st Edition, 1983
[Siegwart]	Siegwart, R., *Grasping the Interdisciplinarity of Mechatronics*, IEEE Robotics and Automation, Vol. 8, No. 2, pp 27-34, June 2001
[Stadler]	Stadler S., *Analytical Robotics and Mechatronics*, McGraw - Hill series in electrical and computer engineering, New York, USA, 1995
[Taylor]	Taylor J., *Principles of Signals and Systems*, McGraw-Hill, New York, 1994

[Taylor et.al.]	Taylor K., Dalton B., *Internet Robots: A New Robotics Niche*, IEEE Robotics and Automation, Vol. 7, No. 1, pp 27-34, March 2000
[Van Gool et.al.]	Van Gool L., Wounbacq P. and Oosterlinck A., *Intelligent Robotic Vision Systems, Intelligent Robotic Vision Systems* (S.G. Tzafestas, Ed.), Dekker, New York, pp 457-507, 1995
[Zhang et.al.]	Zhang J, Gu J., Li P, and Duan Z,, *Object-oriented modelling of control system for agile manufacturing cells.* International Journal on Production Economics, Vol. 62, No. 1, pp 145-153, 1999

Appendix A

Code Listing for the Modular Mechatronic CIM Internet Control System

This appendix list presents a listing of the source code of the Modular Mechatronic CIM Internet control software that was developed.

Components of the control software

The software for the CIM Internet control system was developed using Visual Basic 6 and JAVA. The individual software components are listed below with their forms and modules.

1.	**Web Image**	***- Video.java***
2.	**Host Controller**	***- TCPServerMain.vbp*** *-frmTCPServerMain* *-frmTCPServer1* *-frmTCPServer2* *-frmTCPServer3* *-frmTCPServer4*
3.	**Camera Selector Client**	***- CAMClient.vbp*** *-frmCAMClient* *-frmCAMSelect* *- Driver.bas* *- Global.bas*
4.	**Conveyer Client**	***- ConveyerClient.vbp*** *-frmConveyer* *-frmTCPClient* *- Driver.bas* *-Global.bas*
5.	**Robot Client**	***- ROBOTClient.vbp*** *-frmRobotClient* *-frmPCL832* *-frmInvkin* *- PCL832.bas* *- Def2.bas* *- Funcs.bas*
6.	**Web Client**	***- WEBClient.vbp*** *-frmWEBClient* *-frmWEBTCPClient*
7.	**Image Capture**	***-flahptb.vbp*** *-frmConfig* *-frmFlashPT* *-frmGrab* *-frmLoadImage*

- *frmPrint*
- *frmSaveImage*
- *frmVideo*
- *fp3d.bas*
- *fpg.bas*

1 Code listing for the JAVA applet - Video.java

```
/**This video stream applet loads a new image every 250ms
 */
import java.applet.*;
import java.awt.*;
import java.net.*;
import java.io.*;
import java.awt.event.*;
public class Video extends Applet implements Runnable{
private Image img,buffer;
private Thread video = null;
private URL url;
private MediaTracker tracker;
public synchronized void init(){
tracker = new MediaTracker(this);
try{url = new URL(getCodeBase(),"webcam32.jpg");}
catch(MalformedURLException e){}
img = getImage(url);
tracker.addImage(img, 0);
buffer = createImage(320,240);
resize(320,240);
if(video!=null)return;
video = new Thread(this,"VideoStream");
video.start();
}
public synchronized void destroy(){
if(video!=null)video=null;
}
public void run(){
while(!Thread.interrupted()){
// Load new image.
try{tracker.waitForID(0);}
catch(InterruptedException e){}
// Draw new Image.
repaint();
// Remove old image & add new one.
tracker.removeImage(img,0);
img.flush();
tracker.addImage(img,0);
// Sleep
try{Thread.sleep(250);}
catch(InterruptedException e){}
}
}
public synchronized void update(Graphics g){
Graphics bufGraph = buffer.getGraphics();
bufGraph.drawImage(img,0,0,this);
paint(bufGraph);
g.drawImage(buffer,0,0,this);
}
public void paint(Graphics g){
g.drawImage(buffer,0,0,this);
}
}
```

2 Code listing for the Host controller - TCPServerMain.vbp

2.1 frmTCPServerMain

```
Option Explicit
        Dim MessageIN As String
        Dim MessageOUT As String

Private Sub cmdSend_Click()
        MessageOUT = txtOut.Text
        frmTCPServer1.txtSend.Text = MessageOUT
        frmTCPServer2.txtSend.Text = MessageOUT
        frmTCPServer3.txtSend.Text = MessageOUT
        frmTCPServer4.txtSend.Text = MessageOUT
End Sub

Private Sub Form_Load()
        Load frmTCPServer1
        Load frmTCPServer2
        Load frmTCPServer3
        Load frmTCPServer4
        frmTCPServer1.Hide
        frmTCPServer2.Hide
        frmTCPServer3.Hide
        frmTCPServer4.Hide
End Sub
```

2.2 frmTCPServer1

```
Option Explicit

Private Sub Form_Load()
        ' set up local port and wait for connection
        tcpServer.LocalPort = 5000
        Call tcpServer.Listen
End Sub


Private Sub Form_Terminate()
        Call tcpServer.Close
End Sub

Private Sub tcpServer_ConnectionRequest( _
        ByVal requestID As Long)
        ' Ensure that tcpServer is closed
        ' before accepting a new connection
        If tcpServer.State <> sckClosed Then
                Call tcpServer.Close
        End If

        Call tcpServer.Accept(requestID) ' accept connection
        txtOutput.Text = _
        "Connection from IP address: " & _
        tcpServer.RemoteHostIP & vbCrLf & _
        "Port #: " & tcpServer.RemotePort & vbCrLf & vbCrLf
End Sub

Private Sub tcpServer_DataArrival(ByVal bytesTotal As Long)
        Dim message As String
        Call tcpServer.GetData(message) ' get data from client
        txtOutput.Text = _
        txtOutput.Text & message & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        frmServerMain.lblIn.Caption = message
End Sub

Private Sub tcpServer_Close()
        Call tcpServer.Close  ' client closed, server should too
        txtOutput.Text = txtOutput.Text & _
        "Client closed connection." & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        Call tcpServer.Listen  ' listen for next connection
End Sub

Private Sub tcpServer_Error(ByVal Number As Integer, _
        Description As String, ByVal Scode As Long, _
        ByVal Source As String, ByVal HelpFile As String, _
        ByVal HelpContext As Long, CancelDisplay As Boolean)
        Dim result As Integer
        result = MsgBox(Source & ": " & Description, _
        vbOKOnly, "TCP/IP Error")
End
End Sub

Private Sub txtSend_Change()
        ' send data to the client
        Call tcpServer.SendData(txtSend.Text)
End Sub
```

2.3 frmTCPServer2

```
Option Explicit

Private Sub Form_Load()
        ' set up local port and wait for connection
        tcpServer.LocalPort = 5001
        Call tcpServer.Listen
End Sub


Private Sub Form_Terminate()
        Call tcpServer.Close
End Sub

Private Sub tcpServer_ConnectionRequest( _
        ByVal requestID As Long)
        ' Ensure that tcpServer is closed
        ' before accepting a new connection
        If tcpServer.State <> sckClosed Then
                Call tcpServer.Close
        End If

        Call tcpServer.Accept(requestID)  ' accept connection
        txtOutput.Text = _
        "Connection from IP address: " & _
        tcpServer.RemoteHostIP & vbCrLf & _
        "Port #: " & tcpServer.RemotePort & vbCrLf & vbCrLf
End Sub


Private Sub tcpServer_DataArrival(ByVal bytesTotal As Long)
        Dim message As String
        Call tcpServer.GetData(message)  ' get data from client
        txtOutput.Text = _
        txtOutput.Text & message & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        frmServerMain.lblIn.Caption = message
End Sub

Private Sub tcpServer_Close()
        Call tcpServer.Close  ' client closed, server should too
        txtOutput.Text = txtOutput.Text & _
        "Client closed connection." & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        Call tcpServer.Listen  ' listen for next connection
End Sub

Private Sub tcpServer_Error(ByVal Number As Integer, _
        Description As String, ByVal Scode As Long, _
        ByVal Source As String, ByVal HelpFile As String, _
        ByVal HelpContext As Long, CancelDisplay As Boolean)
        Dim result As Integer
        result = MsgBox(Source & ": " & Description, _
        vbOKOnly, "TCP/IP Error")
End
End Sub

Private Sub txtSend_Change()
        ' send data to the client
        Call tcpServer.SendData(txtSend.Text)
End Sub
```

2.4 frmTCPServer3

```
Option Explicit

Private Sub Form_Load()
        ' set up local port and wait for connection
        tcpServer.LocalPort = 5002
        Call tcpServer.Listen
End Sub


Private Sub Form_Terminate()
        Call tcpServer.Close
End Sub

Private Sub tcpServer_ConnectionRequest( _
        ByVal requestID As Long)
        ' Ensure that tcpServer is closed
        ' before accepting a new connection
        If tcpServer.State <> sckClosed Then
                Call tcpServer.Close
        End If

        Call tcpServer.Accept(requestID)  ' accept connection
        txtOutput.Text = _
        "Connection from IP address: " & _
        tcpServer.RemoteHostIP & vbCrLf & _
        "Port #: " & tcpServer.RemotePort & vbCrLf & vbCrLf
End Sub

Private Sub tcpServer_DataArrival(ByVal bytesTotal As Long)
        Dim message As String
        Call tcpServer.GetData(message)  ' get data from client
        txtOutput.Text = _
        txtOutput.Text & message & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        frmServerMain.lblIn.Caption = message
End Sub

Private Sub tcpServer_Close()
        Call tcpServer.Close  ' client closed, server should too
        txtOutput.Text = txtOutput.Text & _
        "Client closed connection." & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        Call tcpServer.Listen  ' listen for next connection
End Sub

Private Sub tcpServer_Error(ByVal Number As Integer, _
        Description As String, ByVal Scode As Long, _
        ByVal Source As String, ByVal HelpFile As String, _
        ByVal HelpContext As Long, CancelDisplay As Boolean)
        Dim result As Integer
        result = MsgBox(Source & ": " & Description, _
        vbOKOnly, "TCP/IP Error")
End
End Sub

Private Sub txtSend_Change()
        ' send data to the client
        Call tcpServer.SendData(txtSend.Text)
End Sub
```

2.5 frmTCPServer4

```
Option Explicit

Private Sub Form_Load()
        ' set up local port and wait for connection
        tcpServer.LocalPort = 5003
        Call tcpServer.Listen
End Sub

Private Sub Form_Terminate()
        Call tcpServer.Close
End Sub

Private Sub tcpServer_ConnectionRequest( _
        ByVal requestID As Long)
        ' Ensure that tcpServer is closed
        ' before accepting a new connection
        If tcpServer.State <> sckClosed Then
                Call tcpServer.Close
        End If

        Call tcpServer.Accept(requestID)  ' accept connection
        txtOutput.Text = _
        "Connection from IP address: " & _
        tcpServer.RemoteHostIP & vbCrLf & _
        "Port #: " & tcpServer.RemotePort & vbCrLf & vbCrLf
End Sub

Private Sub tcpServer_DataArrival(ByVal bytesTotal As Long)
        Dim message As String
        Call tcpServer.GetData(message)  ' get data from client
        txtOutput.Text = _
        txtOutput.Text & message & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        frmServerMain.lblIn.Caption = message
```

```
End Sub

Private Sub tcpServer_Close()
        Call tcpServer.Close  ' client closed, server should too
        txtOutput.Text = txtOutput.Text & _
        "Client closed connection." & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        Call tcpServer.Listen  ' listen for next connection
End Sub

Private Sub tcpServer_Error(ByVal Number As Integer, _
        Description As String, ByVal Scode As Long, _
        ByVal Source As String, ByVal HelpFile As String, _
        ByVal HelpContext As Long, CancelDisplay As Boolean)
        Dim result As Integer
        result = MsgBox(Source & ": " & Description, _
        vbOKOnly, "TCP/IP Error")
End
End Sub

Private Sub txtSend_Change()
        ' send data to the client
        Call tcpServer.SendData(txtSend.Text)
End Sub
```

3 Code listing for the Camera Selector Client - CAMClient.vbp

3.1 frmCAMClient

```
Option Explicit

Private Sub Form_Load()
  cmdSend.Enabled = False
  ' set up local port and wait for connection
  tcpClient.RemoteHost = "146.230.192.36"
  tcpClient.RemotePort = 5000  ' server port
  Call tcpClient.Connect  ' connect to RemoteHost address
End Sub

Private Sub Form_Terminate()
  Call tcpClient.Close
End Sub

Private Sub Form_Resize()
  On Error Resume Next
  Call cmdSend.Move(ScaleWidth - cmdSend.Width, 0)
  Call txtSend.Move(0, 0, ScaleWidth - cmdSend.Width)
  Call txtOutput.Move(0, txtSend.Height, ScaleWidth, _
    ScaleHeight - txtSend.Height)
End Sub

Private Sub tcpClient_Connect()
  ' when connection occurs, display a message
  cmdSend.Enabled = True
  txtOutput.Text = "Connected to Web Server at IP Address: " & _
    tcpClient.RemoteHostIP & vbCrLf & "Port #: " & _
    tcpClient.RemotePort & vbCrLf & vbCrLf
End Sub

Private Sub tcpClient_DataArrival(ByVal bytesTotal As Long)
  Dim message As String
  Call tcpClient.GetData(message)  ' get data from server
  txtOutput.Text = txtOutput.Text & message & vbCrLf & vbCrLf
  txtOutput.SelStart = Len(txtOutput.Text)
  frmCamSelect.txtMessage.Text = message
End Sub

Private Sub tcpClient_Close()
  cmdSend.Enabled = False
  Call tcpClient.Close  ' server closed, client should too
  txtOutput.Text = _
    txtOutput.Text & "Server closed connection." & vbCrLf
  txtOutput.SelStart = Len(txtOutput.Text)
End Sub

Private Sub tcpClient_Error(ByVal Number As Integer, _
    Description As String, ByVal Scode As Long, _
    ByVal Source As String, ByVal HelpFile As String, _
    ByVal HelpContext As Long, CancelDisplay As Boolean)
  Dim result As Integer
  result = MsgBox(Source & ": " & Description, _
    vbOKOnly, "TCP/IP Error")
  End
End Sub

Private Sub cmdSend_Click()
  ' send data to server
  Call tcpClient.SendData("CLIENT >>> " & txtSend.Text)
```

```
   txtOutput.Text = txtOutput.Text & _
     "CLIENT >>> " & txtSend.Text & vbCrLf & vbCrLf
   txtOutput.SelStart = Len(txtOutput.Text)
   txtSend.Text = ""
End Sub
```

3.2 frmCAMSelect

```
Option Explicit

Private Sub cmdCam1_Click()
      Text1.Text = "HALLO"
      lpDioWritePort.Port = 0          'port number
      lpDioWritePort.Mask = 255
      lpDioWritePort.state = 1         'line numbers bin to decimal
      ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
      If (ErrCde <> 0) Then
         Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
         Exit Sub
      End If
End Sub

Private Sub cmdCam2_Click()
 lpDioWritePort.Port = 0          'port number
      lpDioWritePort.Mask = 255
      lpDioWritePort.state = 2         'line numbers bin to decimal
      ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
      If (ErrCde <> 0) Then
         Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
         Exit Sub
      End If
End Sub

Private Sub cmdCam3_Click()
 lpDioWritePort.Port = 0          'port number
      lpDioWritePort.Mask = 255
      lpDioWritePort.state = 4         'line numbers bin to decimal
      ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
      If (ErrCde <> 0) Then
         Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
         Exit Sub
      End If
End Sub

Private Sub cmdCam4_Click()
 lpDioWritePort.Port = 0          'port number
      lpDioWritePort.Mask = 255
      lpDioWritePort.state = 8         'line numbers bin to decimal
      ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
      If (ErrCde <> 0) Then
         Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
         Exit Sub
      End If
End Sub

Private Sub cmdExit_Click()
   lpDioWritePort.Port = 0          'port number
   lpDioWritePort.Mask = 255
   lpDioWritePort.state = 0         'line numbers bin to decimal
      ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
      If (ErrCde <> 0) Then
      Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
      End If
   lpDioWritePort.Port = 1          'port number
   lpDioWritePort.Mask = 255
   lpDioWritePort.state = 0          'line numbers bin to decimal
     ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
     If (ErrCde <> 0) Then
     Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
     End If
   ErrCde = DRV_DeviceClose(DeviceHandle)
   If (ErrCde <> 0) Then
     DRV_GetErrorMessage ErrCde, szErrMsg
     Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
   End If
End
End Sub

Private Sub Form_Load()
 Dim Cameramessage As String
 Load frmTCPClient
 frmTCPClient.Hide

 'Open Advantech Card
  ErrCde = DRV_DeviceOpen(0, DeviceHandle)
  If (ErrCde <> 0) Then
    .DRV_GetErrorMessage ErrCde, szErrMsg
    Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
    Exit Sub
  Else
    bRun = True
  End If
End Sub

Private Sub Text1_Change()

End Sub

Private Sub txtMessage_Change()
Dim Cameramessage As String
Let Cameramessage = txtMessage.Text
Select Case Cameramessage

  Case "SERVER >>> ClientCAM-CAMERA1":
    lpDioWritePort.Port = 0          'port number
    lpDioWritePort.Mask = 255
    lpDioWritePort.state = 1         'line numbers bin to decimal
    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
    If (ErrCde <> 0) Then
      Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
      Exit Sub
    End If
  Case "SERVER >>> ClientCAM-CAMERA2":
    lpDioWritePort.Port = 0          'port number
    lpDioWritePort.Mask = 255
    lpDioWritePort.state = 2         'line numbers bin to decimal
    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
    If (ErrCde <> 0) Then
      Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
      Exit Sub
    End If
  Case "SERVER >>> ClientCAM-CAMERA3":
    lpDioWritePort.Port = 0          'port number
    lpDioWritePort.Mask = 255
    lpDioWritePort.state = 4         'line numbers bin to decimal
    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
    If (ErrCde <> 0) Then
      Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
      Exit Sub
   End If
  Case "SERVER >>> ClientCAM-CAMERA4":
   lpDioWritePort.Port = 0          'port number
    lpDioWritePort.Mask = 255
    lpDioWritePort.state = 8         'line numbers bin to decimal
    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
    If (ErrCde <> 0) Then
       Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
       Exit Sub
    End If
  Case "SERVER >>> ClientCAM-EXIT":
   lpDioWritePort.Port = 0           'port number
  lpDioWritePort.Mask = 255
  lpDioWritePort.state = 0          'line numbers bin to decimal
    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
    If (ErrCde <> 0) Then
    Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
    End If
  lpDioWritePort.Port = 1           'port number
  lpDioWritePort.Mask = 255
  lpDioWritePort.state = 0          'line numbers bin to decimal
    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
    If (ErrCde <> 0) Then
    Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
    End If
  ErrCde = DRV_DeviceClose(DeviceHandle)
  If (ErrCde <> 0) Then
    DRV_GetErrorMessage ErrCde, szErrMsg
    Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
  End If
End
End Select
End Sub
```

3.3 Driver.bas

```
'*****************************************************************************
' Module Name: DRIVER.BAS
' Purpose: the declaration of functions, data structures, status codes,
'       constants, and messages
'*****************************************************************************

'*****************************************************************************
'   Constant Definition
'*****************************************************************************
Global Const MaxDev = 255 ' max. # of devices
Global Const MaxDevNameLen = 49        ' original is 64; max lenght of device name
Global Const MaxGroup = 6
Global Const MaxPort = 3
Global Const MaxszErrMsgLen = 80
Global Const MAX_DEVICE_NAME_LEN = 64
Global Const MAX_DRIVER_NAME_LEN = 16
Global Const MAX_DAUGHTER_NUM = 16
Global Const MAX_DIO_PORT = 48
Global Const MAX_AO_RANGE = 16

Global Const REMOTE = 1
Global Const REMOTE1 = REMOTE + 1               ' For PCL-818L JP7 = 5V
Global Const REMOTE2 = REMOTE1 + 1               ' For PCL-818L JP7 =10V
Global Const NONPROG = 0
Global Const PROG = REMOTE
Global Const INTERNAL = 0
Global Const EXTERNAL = 1
Global Const SINGLEENDD = 0
Global Const DIFFERENTIAL = 1
Global Const BIPOLAR = 0
Global Const UNIPOLAR = 1
Global Const PORTA = 0
Global Const PORTB = 1
Global Const PORTC = 2
Global Const INPORT = 0
Global Const OUTPORT = 1

'*****************************************************************************
'   Define board vendor ID
'*****************************************************************************
Global Const AAC = &H0                  'Advantech
Global Const KGS = &H4000
'*****************************************************************************
'   Define DAS I/O Board ID.
'*****************************************************************************
Global Const NONE = &H0                 ' not available

'Advantech board ID

Global Const BD_PCL836 = AAC Or &H54            ' PCL-836

'*****************************************************************************
'   Define subsection identifier
'*****************************************************************************
Global Const DAS_AISECTION = &H1                ' A/D subsection
Global Const DAS_AOSECTION = &H2                ' D/A sbusection
Global Const DAS_DISECTION = &H3                ' Digital input subsection
Global Const DAS_DOSECTION = &H4                ' Digital output sbusection
Global Const DAS_TEMPSECTION = &H5              ' thermocouple section
Global Const DAS_ECSECTION = &H6                ' Event count subsection
Global Const DAS_FMSECTION = &H7                ' frequency measurement section
Global Const DAS_POSECTION = &H8                ' pulse output section
Global Const DAS_ALSECTION = &H9                ' alarm section
Global Const MT_AISECTION = &HA                 ' monitoring A/D subsection
Global Const MT_DISECTION = &HB                 ' monitoring D/I subsection

'*****************************************************************************
'   Define Transfer Mode
'*****************************************************************************
Global Const POLLED_MODE = &H0                  ' software transfer
Global Const DMA_MODE = &H1                     ' DMA transfer
Global Const INTERRUPT_MODE = &H2               ' Interrupt transfer

'*****************************************************************************
'   Define Acquisition Mode
'*****************************************************************************
Global Const FREE_RUN = 0
Global Const PRE_TRIG = 1
Global Const POST_TRIG = 2
Global Const POSITION_TRIG = 3

'*****************************************************************************
'   Define Comparator's Condition
'*****************************************************************************
Global Const NOCONDITION = 0
Global Const LESS = 1
Global Const BETWEEN = 2
Global Const GREATER = 3
Global Const OUTSIDE = 4

'*****************************************************************************
'   Define Status Code
```

```
'**************************************************************************
Global Const SUCCESS = 0
Global Const DrvErrorCode = 1
Global Const KeErrorCode = 100
Global Const DnetErrorCode = 200
Global Const MemoryAllocateFailed = (DrvErrorCode + 0)
Global Const ConfigDataLost = (DrvErrorCode + 1)
Global Const InvalidDeviceHandle = (DrvErrorCode + 2)
Global Const AIConversionFailed = (DrvErrorCode + 3)
Global Const AIScaleFailed = (DrvErrorCode + 4)
Global Const SectionNotSupported = (DrvErrorCode + 5)
Global Const InvalidChannel = (DrvErrorCode + 6)
Global Const InvalidGain = (DrvErrorCode + 7)
Global Const DataNotReady = (DrvErrorCode + 8)
Global Const InvalidInputParam = (DrvErrorCode + 9)
Global Const NoExpansionBoardConfig = (DrvErrorCode + 10)
Global Const InvalidAnalogOutValue = (DrvErrorCode + 11)
Global Const ConfigIoPortFailed = (DrvErrorCode + 12)
Global Const CommOpenFailed = (DrvErrorCode + 13)
Global Const CommTransmitFailed = (DrvErrorCode + 14)
Global Const CommReadFailed = (DrvErrorCode + 15)
Global Const CommReceiveFailed = (DrvErrorCode + 16)
Global Const CommConfigFailed = (DrvErrorCode + 17)
Global Const CommChecksumError = (DrvErrorCode + 18)
Global Const InitError = (DrvErrorCode + 19)
Global Const DMABufAllocFailed = (DrvErrorCode + 20)
Global Const IllegalSpeed = (DrvErrorCode + 21)
Global Const ChanConflict = (DrvErrorCode + 22)
Global Const BoardIDNotSupported = (DrvErrorCode + 23)
Global Const FreqMeasurementFailed = (DrvErrorCode + 24)
Global Const CreateFileFailed = (DrvErrorCode + 25)
Global Const FunctionNotSupported = (DrvErrorCode + 26)
Global Const LoadLibraryFailed = (DrvErrorCode + 27)
Global Const GetProcAddressFailed = (DrvErrorCode + 28)
Global Const InvalidDriverHandle = (DrvErrorCode + 29)
Global Const InvalidModuleType = (DrvErrorCode + 30)
Global Const InvalidInputRange = (DrvErrorCode + 31)
Global Const InvalidWindowsHandle = (DrvErrorCode + 32)
Global Const InvalidCountNumber = (DrvErrorCode + 33)
Global Const InvalidInterruptCount = (DrvErrorCode + 34)
Global Const InvalidEventCount = (DrvErrorCode + 35)
Global Const OpenEventFailed = (DrvErrorCode + 36)
Global Const InterruptProcessFailed = (DrvErrorCode + 37)
Global Const InvalidDOSetting = (DrvErrorCode + 38)
Global Const InvalidEventType = (DrvErrorCode + 39)
Global Const EventTimeOut = (DrvErrorCode + 40)
Global Const InvalidDmaChannel = (DrvErrorCode + 41)
Global Const KeInvalidHandleValue = (KeErrorCode + 0)
Global Const KeFileNotFound = (KeErrorCode + 1)
Global Const KeInvalidHandle = (KeErrorCode + 2)
Global Const KeTooManyCmds = (KeErrorCode + 3)
Global Const KeInvalidParameter = (KeErrorCode + 4)
Global Const KeNoAccess = (KeErrorCode + 5)
Global Const KeUnsuccessful = (KeErrorCode + 6)
Global Const KeConInterruptFailure = (KeErrorCode + 7)
Global Const KeCreateNoteFailure = (KeErrorCode + 8)
Global Const KeInsufficientResources = (KeErrorCode + 9)
Global Const KeHalGetAdapterFailure = (KeErrorCode + 10)
Global Const KeOpenEventFailure = (KeErrorCode + 11)
Global Const KeAllocCommBufFailure = (KeErrorCode + 12)
Global Const KeAllocMdlFailure = (KeErrorCode + 13)
Global Const KeBufferSizeTooSmall = (KeErrorCode + 14)
Global Const DNInitFailed = (DnetErrorCode + 1)
Global Const DNSendMsgFailed = (DnetErrorCode + 2)
Global Const DNRunOutOfMsgID = (DnetErrorCode + 3)
Global Const DNInvalidInputParam = (DnetErrorCode + 4)
Global Const DNErrorResponse = (DnetErrorCode + 5)
Global Const DNNoResponse = (DnetErrorCode + 6)
Global Const DNBusyOnNetwork = (DnetErrorCode + 7)
Global Const DNUnknownResponse = (DnetErrorCode + 8)
Global Const DNNotEnoughBuffer = (DnetErrorCode + 9)
Global Const DNFragResponseError = (DnetErrorCode + 10)
Global Const DNTooMuchDataAck = (DnetErrorCode + 11)
Global Const DNFragRequestError = (DnetErrorCode + 12)
Global Const DNEnableEventError = (DnetErrorCode + 13)
Global Const DNCreateOrOpenEventError = (DnetErrorCode + 14)
Global Const DNIORequestError = (DnetErrorCode + 15)
Global Const DNGetEventNameError = (DnetErrorCode + 16)
Global Const DNTimeOutError = (DnetErrorCode + 17)
Global Const DNOpenFailed = (DnetErrorCode + 18)
Global Const DNCloseFailed = (DnetErrorCode + 19)
Global Const DNResetFailed = (DnetErrorCode + 20)
' define user window message
Global Const WM_USER = &H400
Global Const WM_ATODNOTIFY = (WM_USER + 200)
Global Const WM_DTOANOTIFY = (WM_USER + 201)
Global Const WM_DIGINNOTIFY = (WM_USER + 202)
Global Const WM_DIGOUTNOTIFY = (WM_USER + 203)
Global Const WM_MTNOTIFY = (WM_USER + 204)
Global Const WM_CANTRANSMITCOMPLETE = (WM_USER + 205)
Global Const WM_CANMESSAGE = (WM_USER + 206)
Global Const WM_CANERROR = (WM_USER + 207)
' define the wParam in user window message
Global Const AD_NONE = 0              ' AD Section
Global Const AD_TERMINATE = 1
Global Const AD_INT = 2
Global Const AD_BUFFERCHANGE = 3
Global Const AD_OVERRUN = 4
Global Const AD_WATCHDOGACT = 5
Global Const AD_TIMEOUT = 6
Global Const DA_TERMINATE = 0         ' DA Section
Global Const DA_DMATC = 1
Global Const DA_INT = 2
Global Const DA_BUFFERCHANGE = 3
Global Const DA_OVERRUN = 4
Global Const DI_TERMINATE = 0         ' DI Section
Global Const DI_DMATC = 1
Global Const DI_INT = 2
Global Const DI_BUFFERCHANGE = 3
Global Const DI_OVERRUN = 4
Global Const DI_WATCHDOGACT = 5
Global Const DO_TERMINATE = 0         ' DO Section
Global Const DO_DMATC = 1
Global Const DO_INT = 2
Global Const DO_BUFFERCHANGE = 3
Global Const DO_OVERRUN = 4
Global Const MT_ATOD = 0              ' MT Section
Global Const MT_DIGIN = 1
Global Const CAN_TRANSFER = 0         ' CAN Section
Global Const CAN_RECEIVE = 1
Global Const CAN_ERROR = 2
'*****************************************************************************
'   define service type for COMEscape()
'*****************************************************************************
Global Const EscapeFlushInput = 1
Global Const EscapeFlushOutput = 2
Global Const EscapeSetBreak = 3
Global Const EscapeClearBreak = 4

'*****************************************************************************
```

```
'  define gate mode
'*****************************************************************************
Global Const GATE_DISABLED = 0          ' no gating
Global Const GATE_HIGHLEVEL = 1         ' active high level
Global Const GATE_LOWLEVEL = 2          ' active low level
Global Const GATE_HIGHEDGE = 3          ' active high edge
Global Const GATE_LOWEDGE = 4           ' active low edge
'*****************************************************************************
'  define event type for interrupt and DMA transfer
'*****************************************************************************
Global Const ADS_EVT_INTERRUPT = &H1      ' interrupt
Global Const ADS_EVT_BUFCHANGE = &H2      ' buffer change
Global Const ADS_EVT_TERMINATED = &H4     ' termination
Global Const ADS_EVT_OVERRUN = &H8        ' overrun
Global Const ADS_EVT_WATCHDOG = &H10      ' watchdog actived
Global Const ADS_EVT_CHGSTATE = &H20      ' change state event
Global Const ADS_EVT_ALARM = &H40         ' alarm event
Global Const ADS_EVT_PORT0 = &H80         ' port 0 event
Global Const ADS_EVT_PORT1 = &H100        ' port 1 event

'*****************************************************************************
'  define event name by device number
'*****************************************************************************
Global Const ADS_EVT_INTERRUPT_NAME = "ADS_EVT_INTERRUPT"
Global Const ADS_EVT_BUFCHANGE_NAME = "ADS_EVT_BUFCHANGE"
Global Const ADS_EVT_TERMINATED_NAME = "ADS_EVT_TERMINATED"
Global Const ADS_EVT_OVERRUN_NAME = "ADS_EVT_OVERRUN"
Global Const ADS_EVT_WATCHDOG_NAME = "ADS_EVT_WATCHDOG"
Global Const ADS_EVT_CHGSTATE_NAME = "ADS_EVT_CHGSTATE"
Global Const ADS_EVT_ALARM_NAME = "ADS_EVT_ALARM"

' *****************************************************************************
'  define FIFO size
' *****************************************************************************
Global Const FIFO_SIZE = 512            ' 1K FIFO size (512* 2byte/each data)

'*****************************************************************************
'  Function ID Definition
'*****************************************************************************
Global Const FID_DeviceOpen = 0
Global Const FID_DeviceClose = 1
Global Const FID_DeviceGetFeatures = 2
Global Const FID_AIConfig = 3
Global Const FID_AIGetConfig = 4
Global Const FID_AIBinaryIn = 5
Global Const FID_AIScale = 6
Global Const FID_AIVoltageIn = 7
Global Const FID_AIVoltageInExp = 8
Global Const FID_MAIConfig = 9
Global Const FID_MAIBinaryIn = 10
Global Const FID_MAIVoltageIn = 11
Global Const FID_MAIVoltageInExp = 12
Global Const FID_TCMuxRead = 13
Global Const FID_AOConfig = 14
Global Const FID_AOBinaryOut = 15
Global Const FID_AOVoltageOut = 16
Global Const FID_AOScale = 17
Global Const FID_DioSetPortMode = 18
Global Const FID_DioGetConfig = 19
Global Const FID_DioReadPortByte = 20
Global Const FID_DioWritePortByte = 21
Global Const FID_DioReadBit = 22
Global Const FID_DioWriteBit = 23
Global Const FID_DioGetCurrentDOByte = 24
Global Const FID_DioGetCurrentDOBit = 25
Global Const FID_WritePortByte = 26
Global Const FID_WritePortWord = 27
Global Const FID_ReadPortByte = 28
Global Const FID_ReadPortWord = 29
Global Const FID_CounterEventStart = 30
Global Const FID_CounterEventRead = 31
Global Const FID_CounterFreqStart = 32
Global Const FID_CounterFreqRead = 33
Global Const FID_CounterPulseStart = 34
Global Const FID_CounterReset = 35
Global Const FID_QCounterConfig = 36
Global Const FID_QCounterConfigSys = 37
Global Const FID_QCounterStart = 38
Global Const FID_QCounterRead = 39
Global Const FID_AlarmConfig = 40
Global Const FID_AlarmEnable = 41
Global Const FID_AlarmCheck = 42
Global Const FID_AlarmReset = 43
Global Const FID_COMOpen = 44
Global Const FID_COMConfig = 45
Global Const FID_COMClose = 46
Global Const FID_COMRead = 47
Global Const FID_COMWrite232 = 48
Global Const FID_COMWrite485 = 49
Global Const FID_COMWrite85 = 50
Global Const FID_COMInit = 51
Global Const FID_COMLock = 52
Global Const FID_COMUnlock = 53
Global Const FID_WDTEnable = 54
Global Const FID_WDTRefresh = 55
Global Const FID_WDTReset = 56
Global Const FID_FAIIntStart = 57
Global Const FID_FAIIntScanStart = 58
Global Const FID_FAIDmaStart = 59
Global Const FID_FAIDmaScanStart = 60
Global Const FID_FAIDualDmaStart = 61
Global Const FID_FAIDualDmaScanStart = 62
Global Const FID_FAICheck = 63
Global Const FID_FAITransfer = 64
Global Const FID_FAIStop = 65
Global Const FID_FAIWatchdogConfig = 66
Global Const FID_FAIIntWatchdogStart = 67
Global Const FID_FAIDmaWatchdogStart = 68
Global Const FID_FAIWatchdogCheck = 69
Global Const FID_FAOIntStart = 70
Global Const FID_FAODmaStart = 71
Global Const FID_FAOScale = 72
Global Const FID_FAOLoad = 73
Global Const FID_FAOCheck = 74
Global Const FID_FAOStop = 75
Global Const FID_ClearOverrun = 76
Global Const FID_EnableEvent = 77
Global Const FID_CheckEvent = 78
Global Const FID_AllocateDMABuffer = 79
Global Const FID_FreeDMABuffer = 80
Global Const FID_EnableCANEvent = 81
Global Const FID_GetCANEventData = 82

'*************************************************************************
'   define gain listing
'*************************************************************************
Type GainList
   usGainCde    As Integer
   fMaxGainVal  As Single
   fMinGainVal  As Single
```

```
  szGainStr(0 To 15)    As Byte
End Type

'****************************************************************************
'  Define hardware board(device) features.
'
'  Note: definition for dwPermutaion member
'
'        Bit 0: Software AI
'        Bit 1: DMA AI
'        Bit 2: Interrupt AI
'        Bit 3: Condition AI
'        Bit 4: Software AO
'        Bit 5: DMA AO
'        Bit 6: Interrupt AO
'        Bit 7: Condition AO
'        Bit 8: Software DI
'        Bit 9: DMA DI
'        Bit 10: Interrupt DI
'        Bit 11: Condition DI
'        Bit 12: Software DO
'        Bit 13: DMA DO
'        Bit 14: Interrupt DO
'        Bit 15: Condition DO
'        Bit 16: High Gain
'        Bit 17: Auto Channel Scan
'        Bit 18: Pacer Trigger
'        Bit 19: External Trigger
'        Bit 20: Down Counter
'        Bit 21: Dual DMA
'        Bit 22: Monitoring
'        Bit 23: QCounter
'
'****************************************************************************
Type DEVFEATURES
   szDriverVer(0 To 7) As Byte   ' device driver version
   szDriverName(0 To (MAX_DRIVER_NAME_LEN - 1)) As Byte ' device driver name
   dwBoardID      As Long        ' board ID
   usMaxAIDiffChl  As Integer     ' Max. number of differential channel
   usMaxAISiglChl  As Integer     ' Max. number of single-end channel
   usMaxAOChl      As Integer     ' Max. number of D/A channel
   usMaxDOChl      As Integer     ' Max. number of digital out channel
   usMaxDIChl     As Integer      ' Max. number of digital input channel
   usDIOPort      As Integer     ' specifies if programmable or not
   usMaxTimerChl  As Integer      ' Max. number of Counter/Timer channel
   usMaxAlarmChl  As Integer      ' Max number of  alram channel
   usNumADBit     As Integer      ' number of bits for A/D converter
   usNumADByte    As Integer      ' A/D channel width in bytes.
   usNumDABit     As Integer      ' number of bits for D/A converter.
   usNumDAByte    As Integer      ' D/A channel width in bytes.
   usNumGain      As Integer      ' Max. number of gain code
   glGainList(15)  As GainList    ' Gain listing
   dwPermutation(3) As Long       ' Permutation
End Type

'****************************************************************************
'  AOSET Definition
'****************************************************************************
Type AOSET
   usAOSource As Integer      ' 0-internal, 1-external
   fAOMaxVol  As Single       ' maximum output voltage
   fAOMinVol  As Single       ' minimum output voltage
End Type

'****************************************************************************
```

```
'   DaughterSet Definition
'**************************************************************************
Type DAUGHTERSET
   dwBoardID As Long            ' expansion board ID
   usNum    As Integer          ' available expansion channels
   fGain    As Single           ' gain for expansion channel
   usCards  As Integer          ' number of expansion cards
End Type

'**************************************************************************
'   Analog Input Configuration Definition
'**************************************************************************
Type DEVCONFIG_AI
   dwBoardID    As Long        ' board ID code
   usChanConfig  As Integer     ' 0-single ended, 1-differential
   usGainCtrMode As Integer     ' 1-by jumper, 0-programmable
   usPolarity   As Integer     ' 0-bipolar, 1-unipolar
   usDasGain    As Integer     ' not used if GainCtrMode = 1
   usNumExpChan  As Integer     ' DAS channels attached expansion board
   usCjcChannel  As Integer     ' cold junction channel
   Daughter(MAX_DAUGHTER_NUM - 1) As DAUGHTERSET  ' expansion board settings
End Type

'**************************************************************************
'   DEVCONFIG_COM Definition
'**************************************************************************
Type DEVCONFIG_COM
   usCommPort    As Integer           ' serial port
   dwBaudRate    As Long              ' baud rate
   usParity      As Integer           ' parity check
   usDataBits    As Integer           ' data bits
   usStopBits    As Integer           ' stop bits
   usTxMode      As Integer           ' transmission mode
   usPortAddress As Integer           ' communication port address
End Type

'**************************************************************************
'   TRIGLEVEL Definition
'**************************************************************************
Type TRIGLEVEL
  fLow  As Single
  fHigh As Single
End Type


Type PT_DEVLIST
   dwDeviceNum  As Long
   szDeviceName(0 To 49) As Byte
   nNumOfSubdevices As Integer
End Type

Type PT_DeviceGetFeatures
   buffer As Long      ' LPDEVFEATURES
   size   As Integer
End Type

Type PT_AIConfig
   DasChan As Integer
   DasGain As Integer
End Type

Type PT_AIGetConfig
   buffer As Long      ' LPDEVCONFIG_AI
   size   As Integer
End Type
```

```
Type PT_AIBinaryIn
   chan     As Integer
   TrigMode As Integer
   reading  As Long     ' USHORT far * reading
End Type

Type PT_AIScale
   reading  As Integer
   MaxVolt  As Single
   MaxCount As Integer
   offset   As Integer
   voltage  As Long     ' FLOAT far *voltage
End Type

Type PT_AIVoltageIn
   chan     As Integer
   gain     As Integer
   TrigMode As Integer
   voltage  As Long     ' FLOAT far *voltage
End Type

Type PT_AIVoltageInExp
   DasChan As Integer
   DasGain As Integer
   ExpChan As Integer
   voltage As Long      ' FLOAT far *voltage
End Type

Type PT_MAIConfig
   NumChan   As Integer
   StartChan As Integer
   GainArray As Long    ' USHORT far *GainArray
End Type

Type PT_MAIBinaryIn
   NumChan      As Integer
   StartChan    As Integer
   TrigMode     As Integer
   ReadingArray As Long  'USHORT far *Reading
End Type

Type PT_MAIVoltageIn
   NumChan      As Integer
   StartChan    As Integer
   GainArray    As Long 'USHORT far *GainArray
   TrigMode     As Integer
   VoltageArray As Long  'FLOAT far *VoltageArray
End Type

Type PT_MAIVoltageInExp
   NumChan      As Integer
   DasChanArray As Long  ' USHORT far *DasChanArray
   DasGainArray As Long  ' USHORT far *DasGainArray
   ExpChanArray As Long  ' USHORT far *ExpChanArray
   VoltageArray As Long  ' FLOAT  far *VoltageArray
End Type

Type PT_TCMuxRead
   DasChan   As Integer
   DasGain   As Integer
   ExpChan   As Integer
   TCType    As Integer
   TempScale As Integer
   temp      As Long    ' FLOAT far *temp
End Type
```

```
Type PT_AOConfig
  chan    As Integer
  RefSrc  As Integer
  MaxValue As Single
  MinValue As Single
End Type

Type PT_AOBinaryOut
  chan    As Integer
  BinData As Integer
End Type

Type PT_AOVoltageOut
  chan        As Integer
  OutputValue As Single
End Type

Type PT_AOScale
  chan        As Integer
  OutputValue As Single
  BinData     As Long   ' USHORT far *BinData
End Type

Type PT_DioSetPortMode
  Port As Integer
  dir  As Integer
End Type

Type PT_DioGetConfig
  PortArray  As Long    ' SHORT far *PortArray
  NumOfPorts As Integer
End Type

Type PT_DioReadPortByte
  Port As Integer
  value As Long         ' USHORT far *value
End Type

Type PT_DioWritePortByte
  Port  As Integer
  Mask  As Integer
  state As Integer
End Type

Type PT_DioReadBit
  Port  As Integer
  bit   As Integer
  state As Long         ' USHORT far *state
End Type

Type PT_DioWriteBit
  Port  As Integer
  bit   As Integer
  state As Integer
End Type

Type PT_DioGetCurrentDOByte
  Port  As Integer
  value As Long         ' USHORT far *value
End Type

Type PT_DioGetCurrentDOBit
  Port As Integer
  bit  As Integer
  state As Long         ' USHORT far *state
```

```
End Type

Type PT_WritePortByte
  Port    As Integer
  ByteData As Integer
End Type

Type PT_WritePortWord
  Port    As Integer
  WordData As Integer
End Type

Type PT_ReadPortByte
  Port    As Integer
  ByteData As Long      ' USHORT far *ByteData
End Type

Type PT_ReadPortWord
  Port    As Integer
  WordData As Long      ' USHORT far *WordData
End Type

Type PT_CounterEventStart
  counter  As Integer
  GateMode As Integer
End Type

Type PT_CounterEventRead
  counter  As Integer
  overflow As Long      ' USHORT far *overflow
  Count    As Long      ' ULONG  far *count
End Type

Type PT_CounterFreqStart
  counter    As Integer
  GatePeriod As Integer
  GateMode   As Integer
End Type

Type PT_CounterFreqRead
  counter As Integer
  freq    As Long      'FLOAT far *freq
End Type

Type PT_CounterPulseStart
  counter  As Integer
  period   As Single
  UpCycle  As Single
  GateMode As Integer
End Type

Type PT_QCounterConfig
  counter       As Integer
  LatchSrc      As Integer
  LatchOverflow As Integer
  ResetOnLatch  As Integer
  ResetValue    As Integer
End Type

Type PT_QCounterConfigSys
  SysClock    As Integer
  TimeBase    As Integer
  TimeDivider As Integer
  CascadeMode As Integer
End Type
```

```
Type PT_QCounterStart
   counter   As Integer
   InputMode As Integer
End Type

Type PT_QCounterRead
   counter  As Integer
   overflow As Long     ' USHORT far *overflow
   LoCount  As Long     ' ULONG  far *LoCount
   HiCount  As Long     ' ULONG  far *HiCount
End Type

Type PT_AlarmConfig
   chan    As Integer
   LoLimit As Single
   HiLimit As Single
End Type

Type PT_AlarmEnable
   chan      As Integer
   LatchMode As Integer
   Enabled   As Integer
End Type

Type PT_AlarmCheck
   chan    As Integer
   LoState As Long      ' USHORT far *LoState
   HiState As Long      ' USHORT far *HiState
End Type

Type PT_WDTEnable
   message     As Integer
   Destination As Long  ' HWND Destination
End Type

Type PT_FAIIntStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   gain        As Integer
   buffer      As Long
   Count       As Long
   cyclic      As Integer
   IntrCount   As Integer
End Type

Type PT_FAIIntScanStart
   TrigSrc     As Integer
   SampleRate  As Long
   NumChans    As Integer
   StartChan   As Integer
   GainList    As Long
   buffer      As Long
   Count       As Long
   cyclic      As Integer
   IntrCount   As Integer
End Type

Type PT_FAIDmaStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   gain        As Integer
   buffer      As Long
   Count       As Long
```

```
End Type

Type PT_FAIDmaScanStart
   TrigSrc     As Integer
   SampleRate  As Long
   NumChans    As Integer
   StartChan   As Integer
   GainList    As Long
   buffer      As Long
   Count       As Long
End Type

Type PT_FAIDualDmaStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   gain        As Integer
   BufferA     As Long
   BufferB     As Long
   Count       As Long
   cyclic      As Integer
End Type

Type PT_FAIDualDmaScanStart
   TrigSrc     As Integer
   SampleRate  As Long
   NumChans    As Integer
   StartChan   As Integer
   GainList    As Long
   BufferA     As Long
   BufferB     As Long
   Count       As Long
   cyclic      As Integer
End Type

Type PT_FAITransfer
   ActiveBuf   As Integer
   DataBuffer  As Long
   DataType    As Integer
   start       As Long
   Count       As Long
   Overrun     As Long
End Type

Type PT_FAICheck
   ActiveBuf   As Long
   stopped     As Long
   retrieved   As Long
   Overrun     As Long
   HalfReady   As Long
End Type

Type PT_FAIWatchdogConfig
   TrigMode    As Integer
   NumChans    As Integer
   StartChan   As Integer
   GainList    As Long
   CondList    As Long
   LevelList   As Long
End Type

Type PT_FAIIntWatchdogStart
   TrigSrc     As Integer
   SampleRate  As Long
   buffer      As Long
   Count      As Long
   cyclic     As Integer
   IntrCount  As Integer
End Type

Type PT_FAIDmaWatchdogStart
   TrigSrc     As Integer
   SampleRate  As Long
   BufferA     As Long
   BufferB     As Long
   Count       As Long
End Type

Type PT_FAIWatchdogCheck
   DataType    As Integer
   ActiveBuf   As Long
   triggered   As Long
   TrigChan    As Long
   TrigIndex   As Long
   TrigData    As Long
End Type

Type PT_FAOIntStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   buffer      As Long
   Count       As Long
   cyclic      As Integer
End Type

Type PT_FAODmaStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   buffer      As Long
   Count       As Long
End Type

Type PT_FAOScale
   chan        As Integer
   Count       As Long
   VoltArray   As Long
   BinArray    As Long
End Type

Type PT_FAOLoad
   ActiveBuf   As Integer
   DataBuffer  As Long
   start       As Integer
   Count       As Long
End Type

Type PT_FAOCheck
   ActiveBuf    As Long
   stopped      As Long
   CurrentCount As Long
   Overrun      As Long
   HalfReady    As Long
End Type

Type PT_EnableEvent
   EventType   As Integer
   Enabled     As Integer
   Count       As Integer
```

```
End Type

Type PT_CheckEvent
   EventType   As Long
   Milliseconds As Long
End Type

Type PT_AllocateDMABuffer
   CyclicMode    As Integer
   RequestBufSize As Long
   ActualBufSize  As Long
   buffer        As Long
End Type

'*************************************************************************
'  Function Declaration for ADSAPI32
'*************************************************************************
Declare Function DRV_DeviceGetNumOfList Lib "adsapi32.dll" (NumOfDevices As Integer) As Long
Declare Function DRV_DeviceGetList Lib "adsapi32.dll" (ByVal devicelist As Long, ByVal MaxEntries As Integer, nOutEntries As
Integer) As Long
Declare Function DRV_DeviceGetSubList Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal SubDevList As Long, ByVal
MaxEntries As Integer, nOutEntries As Integer) As Long
Declare Function DRV_DeviceOpen Lib "adsapi32.dll" (ByVal DeviceNum As Long, DriverHandle As Long) As Long
Declare Function DRV_DeviceClose Lib "adsapi32.dll" (DriverHandle As Long) As Long
Declare Function DRV_DeviceGetFeatures Lib "adsapi32.dll" (ByVal DriverHandle As Long, lpDevFeatures As
PT_DeviceGetFeatures) As Long
Declare Function DRV_BoardTypeMapBoardName Lib "adsapi32.dll" (ByVal BoardID As Long, ByVal ExpName As String) As Long
Declare Sub DRV_GetErrorMessage Lib "adsapi32.dll" (ByVal lError As Long, ByVal lpszszErrMsg As String)
Declare Function DRV_AIConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIConfig As PT_AIConfig) As Long
Declare Function DRV_AIGetConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIGetConfig As PT_AIGetConfig) As Long
Declare Function DRV_AIBinaryIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIBinaryIn As PT_AIBinaryIn) As Long
Declare Function DRV_AIScale Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIScale As PT_AIScale) As Long
Declare Function DRV_AIVoltageIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIVoltageIn As PT_AIVoltageIn) As Long
Declare Function DRV_AIVoltageInExp Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIVoltageInExp As PT_AIVoltageInExp)
As Long
Declare Function DRV_MAIConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIConfig As PT_MAIConfig) As Long
Declare Function DRV_MAIBinaryIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIBinaryIn As PT_MAIBinaryIn) As Long
Declare Function DRV_MAIVoltageIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIVoltageIn As PT_MAIVoltageIn) As
Long
Declare Function DRV_MAIVoltageInExp Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIVoltageInExp As
PT_MAIVoltageInExp) As Long
Declare Function DRV_TCMuxRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, TCMuxRead As PT_TCMuxRead) As Long
Declare Function DRV_AOConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOConfig As PT_AOConfig) As Long
Declare Function DRV_AOBinaryOut Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOBinaryOut As PT_AOBinaryOut) As Long
Declare Function DRV_AOVoltageOut Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOVoltageOut As PT_AOVoltageOut) As
Long
Declare Function DRV_AOScale Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOScale As PT_AOScale) As Long
Declare Function DRV_DioSetPortMode Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioSetPortMode As PT_DioSetPortMode)
As Long
Declare Function DRV_DioGetConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioGetConfig As PT_DioGetConfig) As
Long
Declare Function DRV_DioReadPortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioReadPortByte As
PT_DioReadPortByte) As Long
Declare Function DRV_DioWritePortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioWritePortByte As
PT_DioWritePortByte) As Long
Declare Function DRV_DioReadBit Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioReadBit As PT_DioReadBit) As Long
Declare Function DRV_DioWriteBit Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioWriteBit As PT_DioWriteBit) As Long
Declare Function DRV_DioGetCurrentDOByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioGetCurrentDOByte As
PT_DioGetCurrentDOByte) As Long
Declare Function DRV_DioGetCurrentDOBit Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioGetCurrentDOBit As
PT_DioGetCurrentDOBit) As Long
Declare Function DRV_WritePortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, WritePortByte As PT_WritePortByte) As
Long
Declare Function DRV_WritePortWord Lib "adsapi32.dll" (ByVal DriverHandle As Long, WritePortWord As PT_WritePortWord) As
Long
```

```
Declare Function DRV_ReadPortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, ReadPortByte As PT_ReadPortByte) As Long
Declare Function DRV_ReadPortWord Lib "adsapi32.dll" (ByVal DriverHandle As Long, ReadPortWord As PT_ReadPortWord) As
Long
Declare Function DRV_CounterEventStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterEventStart As
PT_CounterEventStart) As Long
Declare Function DRV_CounterEventRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterEventRead As
PT_CounterEventRead) As Long
Declare Function DRV_CounterFreqStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterFreqStart As
PT_CounterFreqStart) As Long
Declare Function DRV_CounterFreqRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterFreqRead As
PT_CounterFreqRead) As Long
Declare Function DRV_CounterPulseStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterPulseStart As
PT_CounterPulseStart) As Long
Declare Function DRV_CounterReset Lib "adsapi32.dll" (ByVal DriverHandle As Long, ByVal counter As Integer) As Long
Declare Function DRV_QCounterConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterConfig As PT_QCounterConfig)
As Long
Declare Function DRV_QCounterConfigSys Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterConfigSys As
PT_QCounterConfigSys) As Long
Declare Function DRV_QCounterStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterStart As PT_QCounterStart) As
Long
Declare Function DRV_QCounterRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterRead As PT_QCounterRead) As
Long
Declare Function DRV_AlarmConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AlarmConfig As PT_AlarmConfig) As Long
Declare Function DRV_AlarmEnable Lib "adsapi32.dll" (ByVal DriverHandle As Long, AlarmEnable As PT_AlarmEnable) As Long
Declare Function DRV_AlarmCheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, AlarmCheck As PT_AlarmCheck) As Long
Declare Function DRV_AlarmReset Lib "adsapi32.dll" (ByVal DriverHandle As Long, ByVal chan As Integer) As Long
Declare Function DRV_WDTEnable Lib "adsapi32.dll" (ByVal DriverHandle As Long, WDTEnable As PT_WDTEnable) As Long
Declare Function DRV_WDTRefresh Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_WDTReset Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_GetAddress Lib "adsapi32.dll" (lpVoid As Any) As Long

' Direct I/O Functions List
Declare Function DRV_outp Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByVal ByteData As Long) As
Long
Declare Function DRV_outpw Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByVal ByteData As Long) As
Long
Declare Function DRV_inp Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByteData As Long) As Long
Declare Function DRV_inpw Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByteData As Long) As Long

' High speed function declaration
Declare Function DRV_FAIWatchdogConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIWatchdogConfig As
PT_FAIWatchdogConfig) As Long
Declare Function DRV_FAIIntStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIIntStart As PT_FAIIntStart) As Long
Declare Function DRV_FAIIntScanStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIIntScanStart As PT_FAIIntScanStart)
As Long
Declare Function DRV_FAIDmaStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDmaStart As PT_FAIDmaStart) As Long
Declare Function DRV_FAIDmaScanStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDmaScanStart As
PT_FAIDmaScanStart) As Long
Declare Function DRV_FAIDualDmaStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDualDmaStart As
PT_FAIDualDmaStart) As Long
Declare Function DRV_FAIDualDmaScanStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDualDmaScanStart As
PT_FAIDualDmaScanStart) As Long
Declare Function DRV_FAIIntWatchdogStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIIntWatchdogStart As
PT_FAIIntWatchdogStart) As Long
Declare Function DRV_FAIDmaWatchdogStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDmaWatchdogStart As
PT_FAIDmaWatchdogStart) As Long
Declare Function DRV_FAICheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAICheck As PT_FAICheck) As Long
Declare Function DRV_FAIWatchdogCheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIWatchdogCheck As
PT_FAIWatchdogCheck) As Long
Declare Function DRV_FAITransfer Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAITransfer As PT_FAITransfer) As Long
Declare Function DRV_FAIStop Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_FAOIntStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOIntStart As PT_FAOIntStart) As Long
Declare Function DRV_FAODmaStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAODmaStart As PT_FAODmaStart) As
Long
Declare Function DRV_FAOScale Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOScale As PT_FAOScale) As Long
Declare Function DRV_FAOLoad Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOLoad As PT_FAOLoad) As Long
Declare Function DRV_FAOCheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOCheck As PT_FAOCheck) As Long
Declare Function DRV_FAOStop Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_ClearOverrun Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_EnableEvent Lib "adsapi32.dll" (ByVal DriverHandle As Long, EnableEvent As PT_EnableEvent) As Long
Declare Function DRV_CheckEvent Lib "adsapi32.dll" (ByVal DriverHandle As Long, CheckEvent As PT_CheckEvent) As Long
Declare Function DRV_AllocateDMABuffer Lib "adsapi32.dll" (ByVal DriverHandle As Long, AllocateDMABuffer As
PT_AllocateDMABuffer) As Long
Declare Function DRV_FreeDMABuffer Lib "adsapi32.dll" (ByVal DriverHandle As Long, ByVal buffer As Long) As Long

' CAN bus function declaration
Declare Function CANPortOpen Lib "ads841.dll" (ByVal DevNum As Integer, wPort As Integer, wHostID As Integer) As Long
Declare Function CANPortClose Lib "ads841.dll" (ByVal wPort As Integer) As Long
Declare Function CANInit Lib "ads841.dll" (ByVal Port As Integer, ByVal BTR0 As Integer, ByVal BTR1 As Integer, ByVal usMask
As Byte) As Long
Declare Function CANReset Lib "ads841.dll" (ByVal Port As Integer) As Long
Declare Function CANInpb Lib "ads841.dll" (ByVal Port As Integer, ByVal offset As Integer, Data As Byte) As Long
Declare Function CANOutpb Lib "ads841.dll" (ByVal Port As Integer, ByVal offset As Integer, ByVal value As Byte) As Long
Declare Function CANSetBaud Lib "ads841.dll" (ByVal Port As Integer, ByVal BTR0 As Integer, ByVal BTR1 As Integer) As Long
Declare Function CANSetAcp Lib "ads841.dll" (ByVal Port As Integer, ByVal Acp As Integer, ByVal Mask As Integer) As Long
Declare Function CANSetOutCtrl Lib "ads841.dll" (ByVal Port As Integer, ByVal OutCtrl As Integer) As Long
Declare Function CANSetNormal Lib "ads841.dll" (ByVal Port As Integer) As Long
Declare Function CANHwReset Lib "ads841.dll" (ByVal Port As Integer) As Long
Declare Function CANSendMsg Lib "ads841.dll" (ByVal Port As Integer, ByVal TxBuf As String, ByVal Wait As Long) As Long
Declare Function CANQueryMsg Lib "ads841.dll" (ByVal Port As Integer, Ready As Long, ByVal RcvBuf As String) As Long
Declare Function CANWaitForMsg Lib "ads841.dll" (ByVal Port As Integer, ByVal RcvBuf As String, ByVal uTimeValue As Long)
As Long
Declare Function CANQueryID Lib "ads841.dll" (ByVal Port As Integer, Ready As Long, IDBuf As Byte) As Long
Declare Function CANWaitForID Lib "ads841.dll" (ByVal Port As Integer, IDBuf As Byte, ByVal uTimeValue As Long) As Long
Declare Function CANEnableMessaging Lib "ads841.dll" (ByVal Port As Integer, ByVal Type1 As Integer, ByVal Enabled As Long,
ByVal AppWnd As Long, RcvBuf As String) As Long
Declare Function CANGetEventName Lib "ads841.dll" (ByVal Port As Integer, RcvBuf As Byte) As Long
```

3.4 Global.bas

```
Global Const MaxEntries = 255
Global DeviceHandle As Long
Global ptDevGetFeatures As PT_DeviceGetFeatures
Global lpDevFeatures As DEVFEATURES
Global devicelist(0 To MaxEntries) As PT_DEVLIST
Global SubDevicelist(0 To MaxEntries) As PT_DEVLIST
Global ErrCde As Long
Global szErrMsg As String * 80
Global bRun As Boolean
Global lpDioPortMode As PT_DioSetPortMode
Global lpDioWritePort As PT_DioWritePortByte
Global lpDioReadPort As PT_DioReadPortByte
Const ModeDir = 0   ' for input mode
```

4 Code listing for the Conveyer Client - ConveyerClient.vbp

4.1 frmConveyer

```
Option Explicit
Dim DiValue As Long
Dim counter_AGAV As Integer
Dim counter_AVAG As Integer
Dim counter_AGR As Integer
Dim counter_RAG As Integer
Dim counter_RAVIS As Integer
Dim counter_AVISR As Integer

Private Sub cmd_Exit_Click()
        Unload frmTCPClient
        timer_AGAV.Enabled = False
        timer_AVAG.Enabled = False
        timer_AGR.Enabled = False
        timer_RAGV.Enabled = False
        timer_RAVIS.Enabled = False
        timer_AVISR.Enabled = False

        lpDioWritePort.Port = 0          'port number
        lpDioWritePort.Mask = 255
        lpDioWritePort.state = 0         'line numbers bin to decimal
        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
        End If
        lpDioWritePort.Port = 1          'port number
        lpDioWritePort.Mask = 255
        lpDioWritePort.state = 0         'line numbers bin to decimal
        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
        End If
        ErrCde = DRV_DeviceClose(DeviceHandle)
        If (ErrCde <> 0) Then
                DRV_GetErrorMessage ErrCde, szErrMsg
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
        End If
End
End Sub

Private Sub cmd_Manual_Click()
        cmd_Start.Enabled = True
End Sub

Private Sub cmd_Server_Click()
        Load frmTCPClient
        frmTCPClient.Hide
        'Open Advantech Card

        ErrCde = DRV_DeviceOpen(0, DeviceHandle)
        If (ErrCde <> 0) Then
                DRV_GetErrorMessage ErrCde, szErrMsg
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        Else
                bRun = True
        End If

        cmd_Start.Enabled = False
        cmd_AVIS_R.Enabled = False
        cmd_R_AVIS.Enabled = False
        cmd_R_AGV.Enabled = False
        cmd_AGV_R.Enabled = False
        cmd_AVIS_AGV.Enabled = False
        cmd_AGV_AVIS.Enabled = False
        'cmd_Manual.Enabled = False
End Sub

Private Sub cmd_Start_Click()
        cmd_Start.Enabled = False
        cmd_AVIS_R.Enabled = True
        cmd_R_AVIS.Enabled = True
        cmd_R_AGV.Enabled = True
        cmd_AGV_R.Enabled = True
        cmd_AVIS_AGV.Enabled = True
        cmd_AGV_AVIS.Enabled = True

        lpDioWritePort.Port = 1             'port number
        lpDioWritePort.Mask = 255
        lpDioWritePort.state = 16         'line numbers bin to decimal
        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If
End Sub

Private Sub Form_Load()
        Dim Conveyermessage As String
        cmd_Start.Enabled = False
        cmd_AVIS_R.Enabled = False
        cmd_R_AVIS.Enabled = False
        cmd_R_AGV.Enabled = False
        cmd_AGV_R.Enabled = False
        cmd_AVIS_AGV.Enabled = False
        cmd_AGV_AVIS.Enabled = False
        Let counter_AGAV = 0
        Let counter_AVAG = 0
        Let counter_AGR = 0
        Let counter_RAG = 0
        Let counter_RAVIS = 0
        Let counter_AVISR = 0

        'Open Advantech Card
        ErrCde = DRV_DeviceOpen(0, DeviceHandle)
        If (ErrCde <> 0) Then
                DRV_GetErrorMessage ErrCde, szErrMsg
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        Else
                bRun = True
        End If

        timer_AGAV.Enabled = False
        timer_AVAG.Enabled = False
        timer_AGR.Enabled = False
        timer_RAGV.Enabled = False
        timer_RAVIS.Enabled = False
        timer_AVISR.Enabled = False
End Sub

Private Sub cmd_AGV_AVIS_Click()
        timer_AGAV.Enabled = True
End Sub
```

```
Private Sub Text1_Change()
        Dim Conveyermessage As String
        Let Conveyermessage = Text1.Text
        Select Case Conveyermessage

                Case "SERVER >>> ClientCON-START"
                        cmd_Start.Enabled = False
                        cmd_AVIS_R.Enabled = True
                        cmd_R_AVIS.Enabled = True
                        cmd_R_AGV.Enabled = True
                        cmd_AGV_R.Enabled = True
                        cmd_AVIS_AGV.Enabled = True
                        cmd_AGV_AVIS.Enabled = True
                        lpDioWritePort.Port = 1              'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 16        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If

                Case "SERVER >>> ClientCON-AGVAVIS":
                        lpDioWritePort.Port = 1              'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 16        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                timer_AGAV.Enabled = True

                Case "SERVER >>> ClientCON-AVISAGV":
                        lpDioWritePort.Port = 1              'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 16        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                timer_AVAG.Enabled = True

                Case "SERVER >>> ClientCON-AGVROBOT":
                        lpDioWritePort.Port = 1              'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 16        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                timer_AGR.Enabled = True

                Case "SERVER >>> ClientCON-ROBOTAGV"
                        lpDioWritePort.Port = 1              'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 16        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                timer_RAGV.Enabled = True
```

```
            Case "SERVER >>> ClientCON-ROBOTAVIS"
                    lpDioWritePort.Port = 1              'port number
                    lpDioWritePort.Mask = 255
                    lpDioWritePort.state = 16         'line numbers bin to decimal
                    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                    If (ErrCde <> 0) Then
                            Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                            Exit Sub
                    End If
            timer_RAVIS.Enabled = True

            Case "SERVER >>> ClientCON-AVISROBOT"
                    lpDioWritePort.Mask = 255
                    lpDioWritePort.state = 16         'line numbers bin to decimal
                    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                    If (ErrCde <> 0) Then
                            Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                            Exit Sub
                    End If

            timer_AVISR.Enabled = True

            Case "SERVER >>> ClientCON-EXIT":
                    timer_AGAV.Enabled = False
                    timer_AVAG.Enabled = False
                    timer_AGR.Enabled = False
                    timer_RAGV.Enabled = False
                    timer_RAVIS.Enabled = False
                    timer_AVISR.Enabled = False

                    lpDioWritePort.Port = 0           'port number
                    lpDioWritePort.Mask = 255
                    lpDioWritePort.state = 0          'line numbers bin to decimal
                    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                    If (ErrCde <> 0) Then
                            Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                    End If
                    lpDioWritePort.Port = 1           'port number
                    lpDioWritePort.Mask = 255
                    lpDioWritePort.state = 0          'line numbers bin to decimal
                    ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                    If (ErrCde <> 0) Then
                            Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                    End If
                    ErrCde = DRV_DeviceClose(DeviceHandle)
                    If (ErrCde <> 0) Then
                            DRV_GetErrorMessage ErrCde, szErrMsg
                            Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                    End If
            End
        End Select
End Sub


Private Sub timer_AGAV_Timer()
        lpDioReadPort.Port = 0
        lpDioReadPort.value = DRV_GetAddress(DiValue)
        ErrCde = DRV_DioReadPortByte(DeviceHandle, lpDioReadPort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If

        AGVAVIS (DiValue)
```

```
End Sub

Private Sub AGVAVIS(BitValue As Long)
        If counter_AGAV = 0 Then
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 48       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 16 Then
                Let counter_AGAV = 1
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 60       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 8 Then
                Let counter_AGAV = 2
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 12       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 0 Then
                If counter_AGAV = 2 Then
                        lpDioWritePort.Port = 0         'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 12       'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If

        If DiValue = 4 Then
                Let counter_AGAV = 3
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 15       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 2 Then
                Let counter_AGAV = 4
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 3        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 0 Then
                If counter_AGAV = 4 Then
                        lpDioWritePort.Port = 0         'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 3        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If

        If DiValue = 1 Then
                Let counter_AGAV = 5
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 0        'line numbers bin to decimal

  '................MATE WITH AVIS.........................

                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If
End Sub

Private Sub cmd_AVIS_AGV_Click()
        timer_AVAG.Enabled = True
End Sub

Private Sub timer_AVAG_Timer()
        lpDioReadPort.Port = 0
        lpDioReadPort.value = DRV_GetAddress(DiValue)
        ErrCde = DRV_DioReadPortByte(DeviceHandle, lpDioReadPort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If

        AVISAGV (DiValue)

End Sub

Private Sub AVISAGV(BitValue As Long)
        If counter_AVAG = 0 Then
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 0        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
        End If
End If

If DiValue = 1 Then
        Let counter_AVAG = 1
        lpDioWritePort.Port = 0          'port number
        lpDioWritePort.Mask = 255
        lpDioWritePort.state = 1         'line numbers bin to decimal
        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If
End If

If DiValue = 0 Then
        If counter_AVAG = 1 Then
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 1         'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If
End If

If DiValue = 2 Then
        Let counter_AVAG = 2
        lpDioWritePort.Port = 0          'port number
        lpDioWritePort.Mask = 255
        lpDioWritePort.state = 5         'line numbers bin to decimal
        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If
End If

If DiValue = 4 Then
        Let counter_AVAG = 3
        lpDioWritePort.Port = 0          'port number
        lpDioWritePort.Mask = 255
        lpDioWritePort.state = 4         'line numbers bin to decimal
        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If
End If

If DiValue = 0 Then
        If counter_AVAG = 3 Then
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 4         'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If
End If
```

```
        If DiValue = 8 Then
                Let counter_AVAG = 4
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 20        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 16 Then
                Let counter_AVAG = 5
                lpDioWritePort.Port = 0          'port number
                 lpDioWritePort.Mask = 255
                 lpDioWritePort.state = 16       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 0 Then
                If counter_AVAG = 5 Then
                        lpDioWritePort.Port = 0          'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 16        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If
End Sub

Private Sub cmd_AGV_R_Click()
        timer_AGR.Enabled = True
End Sub

Private Sub timer_AGR_Timer()

        lpDioReadPort.Port = 0
        lpDioReadPort.value = DRV_GetAddress(DiValue)
        ErrCde = DRV_DioReadPortByte(DeviceHandle, lpDioReadPort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If

        AGVR (DiValue)
End Sub

Private Sub AGVR(BitValue As Long)
        If counter_AGR = 0 Then
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 16        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 32 Then
                Let counter_AGR = 1
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 80        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 64 Then
                Let counter_AGR = 2
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 64        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If
End Sub

Private Sub cmd_R_AGV_Click()
        timer_RAGV.Enabled = True
End Sub

Private Sub timer_RAGV_Timer()
        lpDioReadPort.Port = 0
        lpDioReadPort.value = DRV_GetAddress(DiValue)
        ErrCde = DRV_DioReadPortByte(DeviceHandle, lpDioReadPort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If

        RAGV (DiValue)
End Sub

Private Sub RAGV(BitValue As Long)

        If counter_RAG = 0 Then
                If counter_RAG = 0 Then
                        lpDioWritePort.Port = 0          'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 192       'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If

        If DiValue = 64 Then
                Let counter_RAG = 1
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 240       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
            End If
        End If

        If DiValue = 32 Then
            Let counter_RAG = 2
            lpDioWritePort.Port = 0          'port number
            lpDioWritePort.Mask = 255
            lpDioWritePort.state = 48        'line numbers bin to decimal
             ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
            If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
            End If
        End If

        If counter_RAG = 2 Then
            lpDioWritePort.Port = 0          'port number
            lpDioWritePort.Mask = 255
            lpDioWritePort.state = 48        'line numbers bin to decimal
            ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
            If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
            End If
        End If
    End Sub

    Private Sub cmd_R_AVIS_Click()
        timer_RAVIS.Enabled = True
    End Sub

    Private Sub timer_RAVIS_Timer()
        lpDioReadPort.Port = 1
        lpDioReadPort.value = DRV_GetAddress(DiValue)
        ErrCde = DRV_DioReadPortByte(DeviceHandle, lpDioReadPort)
        If (ErrCde <> 0) Then
            Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
            Exit Sub
        End If

        RAVIS (DiValue)
    End Sub

    Private Sub RAVIS(BitValue As Long)
        If counter_RAVIS = 0 Then
            If DiValue = 0 Then
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 64        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                    Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                    Exit Sub
                End If
            End If
        End If

        If DiValue = 1 Then
            Let counter_RAVIS = 1
            lpDioWritePort.Port = 0          'port number
             lpDioWritePort.Mask = 255
            lpDioWritePort.state = 64        'line numbers bin to decimal
            ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
```

```
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
                lpDioWritePort.Port = 1          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 17        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 2 Then
                Let counter_RAVIS = 2
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 0         'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
                lpDioWritePort.Port = 1          'port number
                 lpDioWritePort.Mask = 255
                lpDioWritePort.state = 17        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 0 Then
                If counter_RAVIS = 2 Then
                        lpDioWritePort.Port = 1          'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 17        'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If

        If DiValue = 4 Then
                Let counter_RAVIS = 3
                lpDioWritePort.Port = 1          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 21        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 8 Then
                Let counter_RAVIS = 4
                lpDioWritePort.Port = 1          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 20        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 0 Then
                If counter_RAVIS = 4 Then
                        lpDioWritePort.Port = 1         'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 20       'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If

        If DiValue = 16 Then
                lpDioWritePort.Port = 1         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 16       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If
End Sub

Private Sub cmd_AVIS_R_Click()
        timer_AVISR.Enabled = True
End Sub

Private Sub timer_AVISR_Timer()
        lpDioReadPort.Port = 0
        lpDioReadPort.value = DRV_GetAddress(DiValue)
        ErrCde = DRV_DioReadPortByte(DeviceHandle, lpDioReadPort)
        If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
        End If

        AVISR (DiValue)


End Sub

Private Sub AVISR(BitValue As Long)
        If counter_AVISR = 0 Then
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 0        'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 1 Then
                Let counter_AVISR = 1
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 1          'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 0 Then
                If counter_AVISR = 1 Then
                        lpDioWritePort.Port = 0          'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 1          'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If

        If DiValue = 2 Then
                Let counter_AVISR = 2
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 5          'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 4 Then
                Let counter_AVISR = 3
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 4          'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                        Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                        Exit Sub
                End If
        End If

        If DiValue = 0 Then
                If counter_AVISR = 3 Then
                        lpDioWritePort.Port = 0          'port number
                        lpDioWritePort.Mask = 255
                        lpDioWritePort.state = 4          'line numbers bin to decimal
                        ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                        If (ErrCde <> 0) Then
                                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                                Exit Sub
                        End If
                End If
        End If

        If DiValue = 8 Then
                Let counter_AVISR = 4
                lpDioWritePort.Port = 0          'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 20          'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
            End If
    End If

    If DiValue = 16 Then
            Let counter_AVISR = 5
            lpDioWritePort.Port = 0         'port number
            lpDioWritePort.Mask = 255
            lpDioWritePort.state = 16       'line numbers bin to decimal
            ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
            If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
            End If
    End If

    If DiValue = 0 Then
            If counter_AVISR = 5 Then
                lpDioWritePort.Port = 0         'port number
                lpDioWritePort.Mask = 255
                lpDioWritePort.state = 16       'line numbers bin to decimal
                ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
                If (ErrCde <> 0) Then
                    Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                    Exit Sub
                End If
            End If
    End If


    If DiValue = 32 Then
            Let counter_AVISR = 6
            lpDioWritePort.Port = 0         'port number
            lpDioWritePort.Mask = 255
            lpDioWritePort.state = 80       'line numbers bin to decimal
            ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
            If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
            End If
    End If

    If DiValue = 64 Then
            Let counter_AVISR = 7
            lpDioWritePort.Port = 0         'port number
            lpDioWritePort.Mask = 255
            lpDioWritePort.state = 64       'line numbers bin to decimal
            ErrCde = DRV_DioWritePortByte(DeviceHandle, lpDioWritePort)
            If (ErrCde <> 0) Then
                Call MsgBox(szErrMsg, vbOKOnly, "Error!!")
                Exit Sub
            End If
    End If
End Sub
```

4.2 frmTCPClient

```
Option Explicit

Private Sub Form_Load()
    cmdSend.Enabled = False
    ' set up local port and wait for connection
```

```
        tcpClient.RemoteHost = "146.230.192.36"
        tcpClient.RemotePort = 5000  ' server port
        Call tcpClient.Connect  ' connect to RemoteHost address
End Sub

Private Sub Form_Terminate()
        Call tcpClient.Close
End Sub

Private Sub Form_Resize()
        On Error Resume Next
        Call cmdSend.Move(ScaleWidth - cmdSend.Width, 0)
        Call txtSend.Move(0, 0, ScaleWidth - cmdSend.Width)
        Call txtOutput.Move(0, txtSend.Height, ScaleWidth, _
        ScaleHeight - txtSend.Height)
End Sub

Private Sub tcpClient_Connect()
        ' when connection occurs, display a message
        cmdSend.Enabled = True
        txtOutput.Text = "Connected to Web Server at IP Address: " & _
        tcpClient.RemoteHostIP & vbCrLf & "Port #: " & _
        tcpClient.RemotePort & vbCrLf & vbCrLf
End Sub

Private Sub tcpClient_DataArrival(ByVal bytesTotal As Long)
        Dim message As String
        Call tcpClient.GetData(message)  ' get data from server
        txtOutput.Text = txtOutput.Text & message & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        Conveyer.Text1.Text = message
End Sub

Private Sub tcpClient_Close()
        cmdSend.Enabled = False
        Call tcpClient.Close  ' server closed, client should too
        txtOutput.Text = _
                        txtOutput.Text & "Server closed connection." & vbCrLf
                        txtOutput.SelStart = Len(txtOutput.Text)
End Sub

Private Sub tcpClient_Error(ByVal Number As Integer, _
        Description As String, ByVal Scode As Long, _
        ByVal Source As String, ByVal HelpFile As String, _
        ByVal HelpContext As Long, CancelDisplay As Boolean)
        Dim result As Integer
        result = MsgBox(Source & ": " & Description, _
                vbOKOnly, "TCP/IP Error")
        End
End Sub

Private Sub cmdSend_Click()
        ' send data to server
        Call tcpClient.SendData("ClientCON >>> " & txtSend.Text)
        txtOutput.Text = txtOutput.Text & _
        "ClientCON >>> " & txtSend.Text & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        txtSend.Text = ""
End Sub
```

4.3 Driver.bas

```
'*****************************************************************************
' Module Name: DRIVER.BAS
' Purpose: the declaration of functions, data structures, status codes,
'        constants, and messages
'*****************************************************************************

'*****************************************************************************
'   Constant Definition
'*****************************************************************************
Global Const MaxDev = 255 ' max. # of devices
Global Const MaxDevNameLen = 49          ' original is 64; max lenght of device name
Global Const MaxGroup = 6
Global Const MaxPort = 3
Global Const MaxszErrMsgLen = 80
Global Const MAX_DEVICE_NAME_LEN = 64
Global Const MAX_DRIVER_NAME_LEN = 16
Global Const MAX_DAUGHTER_NUM = 16
Global Const MAX_DIO_PORT = 48
Global Const MAX_AO_RANGE = 16

Global Const REMOTE = 1
Global Const REMOTE1 = REMOTE + 1               ' For PCL-818L JP7 = 5V
Global Const REMOTE2 = REMOTE1 + 1              ' For PCL-818L JP7 =10V
Global Const NONPROG = 0
Global Const PROG = REMOTE
Global Const INTERNAL = 0
Global Const EXTERNAL = 1
Global Const SINGLEENDD = 0
Global Const DIFFERENTIAL = 1
Global Const BIPOLAR = 0
Global Const UNIPOLAR = 1
Global Const PORTA = 0
Global Const PORTB = 1
Global Const PORTC = 2
Global Const INPORT = 0
Global Const OUTPORT = 1

'*****************************************************************************
'   Define board vendor ID
'*****************************************************************************
Global Const AAC = &H0                   'Advantech
Global Const KGS = &H4000
'*****************************************************************************
'   Define DAS I/O Board ID.
'*****************************************************************************
Global Const NONE = &H0                  ' not available

'Advantech board ID

Global Const BD_PCL836 = AAC Or &H54            ' PCL-836

'*****************************************************************************
'   Define subsection identifier
'*****************************************************************************
Global Const DAS_AISECTION = &H1                ' A/D subsection
Global Const DAS_AOSECTION = &H2                ' D/A sbusection
Global Const DAS_DISECTION = &H3                ' Digital input subsection
Global Const DAS_DOSECTION = &H4                ' Digital output sbusection
Global Const DAS_TEMPSECTION = &H5              ' thermocouple section
Global Const DAS_ECSECTION = &H6                ' Event count subsection
Global Const DAS_FMSECTION = &H7                ' frequency measurement section
Global Const DAS_POSECTION = &H8                ' pulse output section
Global Const DAS_ALSECTION = &H9              ' alarm section
Global Const MT_AISECTION = &HA               ' monitoring A/D subsection
Global Const MT_DISECTION = &HB               ' monitoring D/I subsection

'****************************************************************************
'  Define Transfer Mode
'****************************************************************************
Global Const POLLED_MODE = &H0                ' software transfer
Global Const DMA_MODE = &H1                   ' DMA transfer
Global Const INTERRUPT_MODE = &H2             ' Interrupt transfer

'****************************************************************************
'  Define Acquisition Mode
'****************************************************************************
Global Const FREE_RUN = 0
Global Const PRE_TRIG = 1
Global Const POST_TRIG = 2
Global Const POSITION_TRIG = 3

'****************************************************************************
'  Define Comparator's Condition
'****************************************************************************
Global Const NOCONDITION = 0
Global Const LESS = 1
Global Const BETWEEN = 2
Global Const GREATER = 3
Global Const OUTSIDE = 4

'****************************************************************************
'  Define Status Code
'****************************************************************************
Global Const SUCCESS = 0
Global Const DrvErrorCode = 1
Global Const KeErrorCode = 100
Global Const DnetErrorCode = 200
Global Const MemoryAllocateFailed = (DrvErrorCode + 0)
Global Const ConfigDataLost = (DrvErrorCode + 1)
Global Const InvalidDeviceHandle = (DrvErrorCode + 2)
Global Const AIConversionFailed = (DrvErrorCode + 3)
Global Const AIScaleFailed = (DrvErrorCode + 4)
Global Const SectionNotSupported = (DrvErrorCode + 5)
Global Const InvalidChannel = (DrvErrorCode + 6)
Global Const InvalidGain = (DrvErrorCode + 7)
Global Const DataNotReady = (DrvErrorCode + 8)
Global Const InvalidInputParam = (DrvErrorCode + 9)
Global Const NoExpansionBoardConfig = (DrvErrorCode + 10)
Global Const InvalidAnalogOutValue = (DrvErrorCode + 11)
Global Const ConfigIoPortFailed = (DrvErrorCode + 12)
Global Const CommOpenFailed = (DrvErrorCode + 13)
Global Const CommTransmitFailed = (DrvErrorCode + 14)
Global Const CommReadFailed = (DrvErrorCode + 15)
Global Const CommReceiveFailed = (DrvErrorCode + 16)
Global Const CommConfigFailed = (DrvErrorCode + 17)
Global Const CommChecksumError = (DrvErrorCode + 18)
Global Const InitError = (DrvErrorCode + 19)
Global Const DMABufAllocFailed = (DrvErrorCode + 20)
Global Const IllegalSpeed = (DrvErrorCode + 21)
Global Const ChanConflict = (DrvErrorCode + 22)
Global Const BoardIDNotSupported = (DrvErrorCode + 23)
Global Const FreqMeasurementFailed = (DrvErrorCode + 24)
Global Const CreateFileFailed = (DrvErrorCode + 25)
Global Const FunctionNotSupported = (DrvErrorCode + 26)
Global Const LoadLibraryFailed = (DrvErrorCode + 27)
Global Const GetProcAddressFailed = (DrvErrorCode + 28)
Global Const InvalidDriverHandle = (DrvErrorCode + 29)
Global Const InvalidModuleType = (DrvErrorCode + 30)
Global Const InvalidInputRange = (DrvErrorCode + 31)
Global Const InvalidWindowsHandle = (DrvErrorCode + 32)
Global Const InvalidCountNumber = (DrvErrorCode + 33)
Global Const InvalidInterruptCount = (DrvErrorCode + 34)
Global Const InvalidEventCount = (DrvErrorCode + 35)
Global Const OpenEventFailed = (DrvErrorCode + 36)
Global Const InterruptProcessFailed = (DrvErrorCode + 37)
Global Const InvalidDOSetting = (DrvErrorCode + 38)
Global Const InvalidEventType = (DrvErrorCode + 39)
Global Const EventTimeOut = (DrvErrorCode + 40)
Global Const InvalidDmaChannel = (DrvErrorCode + 41)
Global Const KeInvalidHandleValue = (KeErrorCode + 0)
Global Const KeFileNotFound = (KeErrorCode + 1)
Global Const KeInvalidHandle = (KeErrorCode + 2)
Global Const KeTooManyCmds = (KeErrorCode + 3)
Global Const KeInvalidParameter = (KeErrorCode + 4)
Global Const KeNoAccess = (KeErrorCode + 5)
Global Const KeUnsuccessful = (KeErrorCode + 6)
Global Const KeConInterruptFailure = (KeErrorCode + 7)
Global Const KeCreateNoteFailure = (KeErrorCode + 8)
Global Const KeInsufficientResources = (KeErrorCode + 9)
Global Const KeHalGetAdapterFailure = (KeErrorCode + 10)
Global Const KeOpenEventFailure = (KeErrorCode + 11)
Global Const KeAllocCommBufFailure = (KeErrorCode + 12)
Global Const KeAllocMdlFailure = (KeErrorCode + 13)
Global Const KeBufferSizeTooSmall = (KeErrorCode + 14)
Global Const DNInitFailed = (DnetErrorCode + 1)
Global Const DNSendMsgFailed = (DnetErrorCode + 2)
Global Const DNRunOutOfMsgID = (DnetErrorCode + 3)
Global Const DNInvalidInputParam = (DnetErrorCode + 4)
Global Const DNErrorResponse = (DnetErrorCode + 5)
Global Const DNNoResponse = (DnetErrorCode + 6)
Global Const DNBusyOnNetwork = (DnetErrorCode + 7)
Global Const DNUnknownResponse = (DnetErrorCode + 8)
Global Const DNNotEnoughBuffer = (DnetErrorCode + 9)
Global Const DNFragResponseError = (DnetErrorCode + 10)
Global Const DNTooMuchDataAck = (DnetErrorCode + 11)
Global Const DNFragRequestError = (DnetErrorCode + 12)
Global Const DNEnableEventError = (DnetErrorCode + 13)
Global Const DNCreateOrOpenEventError = (DnetErrorCode + 14)
Global Const DNIORequestError = (DnetErrorCode + 15)
Global Const DNGetEventNameError = (DnetErrorCode + 16)
Global Const DNTimeOutError = (DnetErrorCode + 17)
Global Const DNOpenFailed = (DnetErrorCode + 18)
Global Const DNCloseFailed = (DnetErrorCode + 19)
Global Const DNResetFailed = (DnetErrorCode + 20)
' define user window message
Global Const WM_USER = &H400
Global Const WM_ATODNOTIFY = (WM_USER + 200)
Global Const WM_DTOANOTIFY = (WM_USER + 201)
Global Const WM_DIGINNOTIFY = (WM_USER + 202)
Global Const WM_DIGOUTNOTIFY = (WM_USER + 203)
Global Const WM_MTNOTIFY = (WM_USER + 204)
Global Const WM_CANTRANSMITCOMPLETE = (WM_USER + 205)
Global Const WM_CANMESSAGE = (WM_USER + 206)
Global Const WM_CANERROR = (WM_USER + 207)
' define the wParam in user window message
Global Const AD_NONE = 0              ' AD Section
Global Const AD_TERMINATE = 1
Global Const AD_INT = 2
Global Const AD_BUFFERCHANGE = 3
Global Const AD_OVERRUN = 4
Global Const AD_WATCHDOGACT = 5
Global Const AD_TIMEOUT = 6
Global Const DA_TERMINATE = 0          ' DA Section
Global Const DA_DMATC = 1
Global Const DA_INT = 2
Global Const DA_BUFFERCHANGE = 3
Global Const DA_OVERRUN = 4
Global Const DI_TERMINATE = 0          ' DI Section
Global Const DI_DMATC = 1
Global Const DI_INT = 2
Global Const DI_BUFFERCHANGE = 3
Global Const DI_OVERRUN = 4
Global Const DI_WATCHDOGACT = 5
Global Const DO_TERMINATE = 0          ' DO Section
Global Const DO_DMATC = 1
Global Const DO_INT = 2
Global Const DO_BUFFERCHANGE = 3
Global Const DO_OVERRUN = 4
Global Const MT_ATOD = 0               ' MT Section
Global Const MT_DIGIN = 1
Global Const CAN_TRANSFER = 0          ' CAN Section
Global Const CAN_RECEIVE = 1
Global Const CAN_ERROR = 2
'**************************************************************************
'   define service type for COMEscape()
'**************************************************************************
Global Const EscapeFlushInput = 1
Global Const EscapeFlushOutput = 2
Global Const EscapeSetBreak = 3
Global Const EscapeClearBreak = 4

'**************************************************************************
'   define  gate mode
'**************************************************************************
Global Const GATE_DISABLED = 0          ' no gating
Global Const GATE_HIGHLEVEL = 1         ' active high level
Global Const GATE_LOWLEVEL = 2          ' active low level
Global Const GATE_HIGHEDGE = 3          ' active high edge
Global Const GATE_LOWEDGE = 4           ' active low edge
'**************************************************************************
'  define event type for interrupt and DMA transfer
'**************************************************************************
Global Const ADS_EVT_INTERRUPT = &H1       ' interrupt
Global Const ADS_EVT_BUFCHANGE = &H2       ' buffer change
Global Const ADS_EVT_TERMINATED = &H4      ' termination
Global Const ADS_EVT_OVERRUN = &H8         ' overrun
Global Const ADS_EVT_WATCHDOG = &H10       ' watchdog actived
Global Const ADS_EVT_CHGSTATE = &H20       ' change state event
Global Const ADS_EVT_ALARM = &H40          ' alarm event
Global Const ADS_EVT_PORT0 = &H80          ' port 0 event
Global Const ADS_EVT_PORT1 = &H100         ' port 1 event

'**************************************************************************
'   define event name by device number
'**************************************************************************
Global Const ADS_EVT_INTERRUPT_NAME = "ADS_EVT_INTERRUPT"
Global Const ADS_EVT_BUFCHANGE_NAME = "ADS_EVT_BUFCHANGE"
Global Const ADS_EVT_TERMINATED_NAME = "ADS_EVT_TERMINATED"
Global Const ADS_EVT_OVERRUN_NAME = "ADS_EVT_OVERRUN"
Global Const ADS_EVT_WATCHDOG_NAME = "ADS_EVT_WATCHDOG"
Global Const ADS_EVT_CHGSTATE_NAME = "ADS_EVT_CHGSTATE"
Global Const ADS_EVT_ALARM_NAME = "ADS_EVT_ALARM"

' **************************************************************************
'   define FIFO size
' **************************************************************************
Global Const FIFO_SIZE = 512            ' 1K FIFO size (512* 2byte/each data)
```

```
'*****************************************************************************
' Function ID Definition
'*****************************************************************************
Global Const FID_DeviceOpen = 0
Global Const FID_DeviceClose = 1
Global Const FID_DeviceGetFeatures = 2
Global Const FID_AIConfig = 3
Global Const FID_AIGetConfig = 4
Global Const FID_AIBinaryIn = 5
Global Const FID_AIScale = 6
Global Const FID_AIVoltageIn = 7
Global Const FID_AIVoltageInExp = 8
Global Const FID_MAIConfig = 9
Global Const FID_MAIBinaryIn = 10
Global Const FID_MAIVoltageIn = 11
Global Const FID_MAIVoltageInExp = 12
Global Const FID_TCMuxRead = 13
Global Const FID_AOConfig = 14
Global Const FID_AOBinaryOut = 15
Global Const FID_AOVoltageOut = 16
Global Const FID_AOScale = 17
Global Const FID_DioSetPortMode = 18
Global Const FID_DioGetConfig = 19
Global Const FID_DioReadPortByte = 20
Global Const FID_DioWritePortByte = 21
Global Const FID_DioReadBit = 22
Global Const FID_DioWriteBit = 23
Global Const FID_DioGetCurrentDOByte = 24
Global Const FID_DioGetCurrentDOBit = 25
Global Const FID_WritePortByte = 26
Global Const FID_WritePortWord = 27
Global Const FID_ReadPortByte = 28
Global Const FID_ReadPortWord = 29
Global Const FID_CounterEventStart = 30
Global Const FID_CounterEventRead = 31
Global Const FID_CounterFreqStart = 32
Global Const FID_CounterFreqRead = 33
Global Const FID_CounterPulseStart = 34
Global Const FID_CounterReset = 35
Global Const FID_QCounterConfig = 36
Global Const FID_QCounterConfigSys = 37
Global Const FID_QCounterStart = 38
Global Const FID_QCounterRead = 39
Global Const FID_AlarmConfig = 40
Global Const FID_AlarmEnable = 41
Global Const FID_AlarmCheck = 42
Global Const FID_AlarmReset = 43
Global Const FID_COMOpen = 44
Global Const FID_COMConfig = 45
Global Const FID_COMClose = 46
Global Const FID_COMRead = 47
Global Const FID_COMWrite232 = 48
Global Const FID_COMWrite485 = 49
Global Const FID_COMWrite85 = 50
Global Const FID_COMInit = 51
Global Const FID_COMLock = 52
Global Const FID_COMUnlock = 53
Global Const FID_WDTEnable = 54
Global Const FID_WDTRefresh = 55
Global Const FID_WDTReset = 56
Global Const FID_FAIIntStart = 57
Global Const FID_FAIIntScanStart = 58
Global Const FID_FAIDmaStart = 59
Global Const FID_FAIDmaScanStart = 60
Global Const FID_FAIDualDmaStart = 61
Global Const FID_FAIDualDmaScanStart = 62
Global Const FID_FAICheck = 63
Global Const FID_FAITransfer = 64
Global Const FID_FAIStop = 65
Global Const FID_FAIWatchdogConfig = 66
Global Const FID_FAIIntWatchdogStart = 67
Global Const FID_FAIDmaWatchdogStart = 68
Global Const FID_FAIWatchdogCheck = 69
Global Const FID_FAOIntStart = 70
Global Const FID_FAODmaStart = 71
Global Const FID_FAOScale = 72
Global Const FID_FAOLoad = 73
Global Const FID_FAOCheck = 74
Global Const FID_FAOStop = 75
Global Const FID_ClearOverrun = 76
Global Const FID_EnableEvent = 77
Global Const FID_CheckEvent = 78
Global Const FID_AllocateDMABuffer = 79
Global Const FID_FreeDMABuffer = 80
Global Const FID_EnableCANEvent = 81
Global Const FID_GetCANEventData = 82

'************************************************************************
'  define gain listing
'************************************************************************
Type GainList
   usGainCde     As Integer
   fMaxGainVal   As Single
   fMinGainVal   As Single
   szGainStr(0 To 15)    As Byte
End Type

'************************************************************************
'  Define hardware board(device) features.
'
'  Note: definition for dwPermutaion member
'
'        Bit 0: Software AI
'        Bit 1: DMA AI
'        Bit 2: Interrupt AI
'        Bit 3: Condition AI
'        Bit 4: Software AO
'        Bit 5: DMA AO
'        Bit 6: Interrupt AO
'        Bit 7: Condition AO
'        Bit 8: Software DI
'        Bit 9: DMA DI
'        Bit 10: Interrupt DI
'        Bit 11: Condition DI
'        Bit 12: Software DO
'        Bit 13: DMA DO
'        Bit 14: Interrupt DO
'        Bit 15: Condition DO
'        Bit 16: High Gain
'        Bit 17: Auto Channel Scan
'        Bit 18: Pacer Trigger
'        Bit 19: External Trigger
'        Bit 20: Down Counter
'        Bit 21: Dual DMA
'        Bit 22: Monitoring
'        Bit 23: QCounter
'
'************************************************************************
Type DEVFEATURES
   szDriverVer(0 To 7) As Byte    ' device driver version
```

```
    szDriverName(0 To (MAX_DRIVER_NAME_LEN - 1)) As Byte ' device driver name
    dwBoardID      As Long       ' board ID
    usMaxAIDiffChl  As Integer     ' Max. number of differential channel
    usMaxAISiglChl  As Integer     ' Max. number of single-end channel
    usMaxAOChl     As Integer     ' Max. number of D/A channel
    usMaxDOChl     As Integer     ' Max. number of digital out channel
    usMaxDIChl     As Integer    ' Max. number of digital input channel
    usDIOPort      As Integer   ' specifies if programmable or not
    usMaxTimerChl  As Integer     ' Max. number of Counter/Timer channel
    usMaxAlarmChl  As Integer     ' Max number of  alram channel
    usNumADBit     As Integer    ' number of bits for A/D converter
    usNumADByte    As Integer     ' A/D channel width in bytes.
    usNumDABit     As Integer    ' number of bits for D/A converter.
    usNumDAByte    As Integer     ' D/A channel width in bytes.
    usNumGain      As Integer    ' Max. number of gain code
    glGainList(15)  As GainList    ' Gain listing
    dwPermutation(3) As Long       ' Permutation
End Type

'*************************************************************************
'   AOSET Definition
'*************************************************************************
Type AOSET
    usAOSource As Integer     ' 0-internal, 1-external
    fAOMaxVol  As Single      ' maximum output voltage
    fAOMinVol  As Single      ' minimum output voltage
End Type

'*************************************************************************
'   DaughterSet Definition
'*************************************************************************
Type DAUGHTERSET
    dwBoardID As Long             ' expansion board ID
    usNum     As Integer          ' available expansion channels
    fGain     As Single          ' gain for expansion channel
    usCards   As Integer          ' number of expansion cards
End Type

'*************************************************************************
'   Analog Input Configuration Definition
'*************************************************************************
Type DEVCONFIG_AI
    dwBoardID     As Long       ' board ID code
    usChanConfig  As Integer     ' 0-single ended, 1-differential
    usGainCtrMode As Integer      ' 1-by jumper, 0-programmable
    usPolarity    As Integer    ' 0-bipolar, 1-unipolar
    usDasGain     As Integer    ' not used if GainCtrMode = 1
    usNumExpChan  As Integer     ' DAS channels attached expansion board
    usCjcChannel  As Integer     ' cold junction channel
    Daughter(MAX_DAUGHTER_NUM - 1) As DAUGHTERSET  ' expansion board settings
End Type

'*************************************************************************
'   DEVCONFIG_COM Definition
'*************************************************************************
Type DEVCONFIG_COM
    usCommPort   As Integer            ' serial port
    dwBaudRate   As Long               ' baud rate
    usParity     As Integer          ' parity check
    usDataBits   As Integer            ' data bits
    usStopBits   As Integer           ' stop bits
    usTxMode     As Integer             ' transmission mode
    usPortAddress As Integer            ' communication port address
End Type
```

```
'*****************************************************************************
'  TRIGLEVEL Definition
'*****************************************************************************
Type TRIGLEVEL
 fLow  As Single
 fHigh As Single
End Type


Type PT_DEVLIST
   dwDeviceNum  As Long
   szDeviceName(0 To 49) As Byte
   nNumOfSubdevices As Integer
End Type

Type PT_DeviceGetFeatures
   buffer As Long       ' LPDEVFEATURES
   size   As Integer
End Type

Type PT_AIConfig
   DasChan As Integer
   DasGain As Integer
End Type

Type PT_AIGetConfig
   buffer As Long       ' LPDEVCONFIG_AI
   size   As Integer
End Type

Type PT_AIBinaryIn
   chan     As Integer
   TrigMode As Integer
   reading  As Long     ' USHORT far * reading
End Type

Type PT_AIScale
   reading  As Integer
   MaxVolt  As Single
   MaxCount As Integer
   offset   As Integer
   voltage  As Long     ' FLOAT far *voltage
End Type

Type PT_AIVoltageIn
   chan     As Integer
   gain     As Integer
   TrigMode As Integer
   voltage  As Long     ' FLOAT far *voltage
End Type

Type PT_AIVoltageInExp
   DasChan As Integer
   DasGain As Integer
   ExpChan As Integer
   voltage As Long      ' FLOAT far *voltage
End Type

Type PT_MAIConfig
   NumChan   As Integer
   StartChan As Integer
   GainArray As Long    ' USHORT far *GainArray
End Type

Type PT_MAIBinaryIn
```

```
  NumChan     As Integer
  StartChan   As Integer
  TrigMode    As Integer
  ReadingArray As Long  'USHORT far *Reading
End Type

Type PT_MAIVoltageIn
  NumChan     As Integer
  StartChan   As Integer
  GainArray   As Long  'USHORT far *GainArray
  TrigMode    As Integer
  VoltageArray As Long  'FLOAT far *VoltageArray
End Type

Type PT_MAIVoltageInExp
  NumChan     As Integer
  DasChanArray As Long  ' USHORT far *DasChanArray
  DasGainArray As Long  ' USHORT far *DasGainArray
  ExpChanArray As Long  ' USHORT far *ExpChanArray
  VoltageArray As Long  ' FLOAT  far *VoltageArray
End Type

Type PT_TCMuxRead
  DasChan   As Integer
  DasGain   As Integer
  ExpChan   As Integer
  TCType    As Integer
  TempScale As Integer
  temp      As Long     ' FLOAT far *temp
End Type

Type PT_AOConfig
  chan     As Integer
  RefSrc   As Integer
  MaxValue As Single
  MinValue As Single
End Type

Type PT_AOBinaryOut
  chan    As Integer
  BinData As Integer
End Type

Type PT_AOVoltageOut
  chan        As Integer
  OutputValue As Single
End Type

Type PT_AOScale
  chan        As Integer
  OutputValue As Single
  BinData     As Long   ' USHORT far *BinData
End Type

Type PT_DioSetPortMode
  Port As Integer
  dir  As Integer
End Type

Type PT_DioGetConfig
  PortArray  As Long    ' SHORT far *PortArray
  NumOfPorts As Integer
End Type

Type PT_DioReadPortByte
```

```
   Port As Integer
   value As Long        ' USHORT far *value
End Type

Type PT_DioWritePortByte
   Port  As Integer
   Mask  As Integer
   state As Integer
End Type

Type PT_DioReadBit
   Port  As Integer
   bit   As Integer
   state As Long       ' USHORT far *state
End Type

Type PT_DioWriteBit
   Port  As Integer
   bit   As Integer
   state As Integer
End Type

Type PT_DioGetCurrentDOByte
   Port  As Integer
   value As Long        ' USHORT far *value
End Type

Type PT_DioGetCurrentDOBit
   Port  As Integer
   bit   As Integer
   state As Long        ' USHORT far *state
End Type

Type PT_WritePortByte
   Port     As Integer
   ByteData As Integer
End Type

Type PT_WritePortWord
   Port     As Integer
   WordData As Integer
End Type

Type PT_ReadPortByte
   Port     As Integer
   ByteData As Long     ' USHORT far *ByteData
End Type

Type PT_ReadPortWord
   Port     As Integer
   WordData As Long     ' USHORT far *WordData
End Type

Type PT_CounterEventStart
   counter  As Integer
   GateMode As Integer
End Type

Type PT_CounterEventRead
   counter  As Integer
   overflow As Long     ' USHORT far *overflow
   Count    As Long     ' ULONG  far *count
End Type

Type PT_CounterFreqStart
```

```
   counter    As Integer
   GatePeriod As Integer
   GateMode   As Integer
End Type

Type PT_CounterFreqRead
   counter As Integer
   freq    As Long      'FLOAT far *freq
End Type

Type PT_CounterPulseStart
   counter  As Integer
   period   As Single
   UpCycle  As Single
   GateMode As Integer
End Type

Type PT_QCounterConfig
   counter       As Integer
   LatchSrc      As Integer
   LatchOverflow As Integer
   ResetOnLatch  As Integer
   ResetValue    As Integer
End Type

Type PT_QCounterConfigSys
   SysClock    As Integer
   TimeBase    As Integer
   TimeDivider As Integer
   CascadeMode As Integer
End Type

Type PT_QCounterStart
   counter   As Integer
   InputMode As Integer
End Type

Type PT_QCounterRead
   counter  As Integer
   overflow As Long    ' USHORT far *overflow
   LoCount  As Long    ' ULONG  far *LoCount
   HiCount  As Long    ' ULONG  far *HiCount
End Type

Type PT_AlarmConfig
   chan    As Integer
   LoLimit As Single
   HiLimit As Single
End Type

Type PT_AlarmEnable
   chan      As Integer
   LatchMode As Integer
   Enabled   As Integer
End Type

Type PT_AlarmCheck
   chan    As Integer
   LoState As Long     ' USHORT far *LoState
   HiState As Long     ' USHORT far *HiState
End Type

Type PT_WDTEnable
   message     As Integer
   Destination As Long  ' HWND Destination
```

```
End Type

Type PT_FAIIntStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   gain        As Integer
   buffer      As Long
   Count       As Long
   cyclic      As Integer
   IntrCount   As Integer
End Type

Type PT_FAIIntScanStart
   TrigSrc     As Integer
   SampleRate  As Long
   NumChans    As Integer
   StartChan   As Integer
   GainList    As Long
   buffer      As Long
   Count       As Long
   cyclic      As Integer
   IntrCount   As Integer
End Type

Type PT_FAIDmaStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   gain        As Integer
   buffer      As Long
   Count       As Long
End Type

Type PT_FAIDmaScanStart
   TrigSrc     As Integer
   SampleRate  As Long
   NumChans    As Integer
   StartChan   As Integer
   GainList    As Long
   buffer      As Long
   Count       As Long
End Type

Type PT_FAIDualDmaStart
   TrigSrc     As Integer
   SampleRate  As Long
   chan        As Integer
   gain        As Integer
   BufferA     As Long
   BufferB     As Long
   Count       As Long
   cyclic      As Integer
End Type

Type PT_FAIDualDmaScanStart
   TrigSrc     As Integer
   SampleRate  As Long
   NumChans    As Integer
   StartChan   As Integer
   GainList    As Long
   BufferA     As Long
   BufferB     As Long
   Count       As Long
   cyclic      As Integer
```

```
End Type

Type PT_FAITransfer
   ActiveBuf  As Integer
   DataBuffer As Long
   DataType   As Integer
   start      As Long
   Count      As Long
   Overrun    As Long
End Type

Type PT_FAICheck
   ActiveBuf  As Long
   stopped    As Long
   retrieved  As Long
   Overrun    As Long
   HalfReady  As Long
End Type

Type PT_FAIWatchdogConfig
   TrigMode   As Integer
   NumChans   As Integer
   StartChan  As Integer
   GainList   As Long
   CondList   As Long
   LevelList  As Long
End Type

Type PT_FAIIntWatchdogStart
   TrigSrc    As Integer
   SampleRate As Long
   buffer     As Long
   Count      As Long
   cyclic     As Integer
   IntrCount  As Integer
End Type

Type PT_FAIDmaWatchdogStart
   TrigSrc    As Integer
   SampleRate As Long
   BufferA    As Long
   BufferB    As Long
   Count      As Long
End Type

Type PT_FAIWatchdogCheck
   DataType   As Integer
   ActiveBuf  As Long
   triggered  As Long
   TrigChan   As Long
   TrigIndex  As Long
   TrigData   As Long
End Type

Type PT_FAOIntStart
   TrigSrc    As Integer
   SampleRate As Long
   chan       As Integer
   buffer     As Long
   Count      As Long
   cyclic     As Integer
End Type

Type PT_FAODmaStart
   TrigSrc    As Integer
```

```
  SampleRate  As Long
  chan        As Integer
  buffer      As Long
  Count       As Long
End Type

Type PT_FAOScale
  chan        As Integer
  Count       As Long
  VoltArray   As Long
  BinArray    As Long
End Type

Type PT_FAOLoad
  ActiveBuf   As Integer
  DataBuffer  As Long
  start       As Integer
  Count       As Long
End Type

Type PT_FAOCheck
  ActiveBuf    As Long
  stopped      As Long
  CurrentCount As Long
  Overrun      As Long
  HalfReady    As Long
End Type

Type PT_EnableEvent
  EventType    As Integer
  Enabled      As Integer
  Count        As Integer
End Type

Type PT_CheckEvent
  EventType    As Long
  Milliseconds As Long
End Type

Type PT_AllocateDMABuffer
  CyclicMode     As Integer
  RequestBufSize As Long
  ActualBufSize  As Long
  buffer         As Long
End Type

'*****************************************************************************
'  Function Declaration for ADSAPI32
'*****************************************************************************
Declare Function DRV_DeviceGetNumOfList Lib "adsapi32.dll" (NumOfDevices As Integer) As Long
Declare Function DRV_DeviceGetList Lib "adsapi32.dll" (ByVal devicelist As Long, ByVal MaxEntries As Integer, nOutEntries As
Integer) As Long
Declare Function DRV_DeviceGetSubList Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal SubDevList As Long, ByVal
MaxEntries As Integer, nOutEntries As Integer) As Long
Declare Function DRV_DeviceOpen Lib "adsapi32.dll" (ByVal DeviceNum As Long, DriverHandle As Long) As Long
Declare Function DRV_DeviceClose Lib "adsapi32.dll" (DriverHandle As Long) As Long
Declare Function DRV_DeviceGetFeatures Lib "adsapi32.dll" (ByVal DriverHandle As Long, lpDevFeatures As
PT_DeviceGetFeatures) As Long
Declare Function DRV_BoardTypeMapBoardName Lib "adsapi32.dll" (ByVal BoardID As Long, ByVal ExpName As String) As Long
Declare Sub DRV_GetErrorMessage Lib "adsapi32.dll" (ByVal lError As Long, ByVal lpszszErrMsg As String)
Declare Function DRV_AIConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIConfig As PT_AIConfig) As Long
Declare Function DRV_AIGetConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIGetConfig As PT_AIGetConfig) As Long
Declare Function DRV_AIBinaryIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIBinaryIn As PT_AIBinaryIn) As Long
Declare Function DRV_AIScale Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIScale As PT_AIScale) As Long
Declare Function DRV_AIVoltageIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIVoltageIn As PT_AIVoltageIn) As Long
Declare Function DRV_AIVoltageInExp Lib "adsapi32.dll" (ByVal DriverHandle As Long, AIVoltageInExp As PT_AIVoltageInExp)
As Long
Declare Function DRV_MAIConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIConfig As PT_MAIConfig) As Long
Declare Function DRV_MAIBinaryIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIBinaryIn As PT_MAIBinaryIn) As Long
Declare Function DRV_MAIVoltageIn Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIVoltageIn As PT_MAIVoltageIn) As
Long
Declare Function DRV_MAIVoltageInExp Lib "adsapi32.dll" (ByVal DriverHandle As Long, MAIVoltageInExp As
PT_MAIVoltageInExp) As Long
Declare Function DRV_TCMuxRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, TCMuxRead As PT_TCMuxRead) As Long
Declare Function DRV_AOConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOConfig As PT_AOConfig) As Long
Declare Function DRV_AOBinaryOut Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOBinaryOut As PT_AOBinaryOut) As Long
Declare Function DRV_AOVoltageOut Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOVoltageOut As PT_AOVoltageOut) As
Long
Declare Function DRV_AOScale Lib "adsapi32.dll" (ByVal DriverHandle As Long, AOScale As PT_AOScale) As Long
Declare Function DRV_DioSetPortMode Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioSetPortMode As PT_DioSetPortMode)
As Long
Declare Function DRV_DioGetConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioGetConfig As PT_DioGetConfig) As
Long
Declare Function DRV_DioReadPortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioReadPortByte As
PT_DioReadPortByte) As Long
Declare Function DRV_DioWritePortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioWritePortByte As
PT_DioWritePortByte) As Long
Declare Function DRV_DioReadBit Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioReadBit As PT_DioReadBit) As Long
Declare Function DRV_DioWriteBit Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioWriteBit As PT_DioWriteBit) As Long
Declare Function DRV_DioGetCurrentDOByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioGetCurrentDOByte As
PT_DioGetCurrentDOByte) As Long
Declare Function DRV_DioGetCurrentDOBit Lib "adsapi32.dll" (ByVal DriverHandle As Long, DioGetCurrentDOBit As
PT_DioGetCurrentDOBit) As Long
Declare Function DRV_WritePortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, WritePortByte As PT_WritePortByte) As
Long
Declare Function DRV_WritePortWord Lib "adsapi32.dll" (ByVal DriverHandle As Long, WritePortWord As PT_WritePortWord) As
Long
Declare Function DRV_ReadPortByte Lib "adsapi32.dll" (ByVal DriverHandle As Long, ReadPortByte As PT_ReadPortByte) As Long
Declare Function DRV_ReadPortWord Lib "adsapi32.dll" (ByVal DriverHandle As Long, ReadPortWord As PT_ReadPortWord) As
Long
Declare Function DRV_CounterEventStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterEventStart As
PT_CounterEventStart) As Long
Declare Function DRV_CounterEventRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterEventRead As
PT_CounterEventRead) As Long
Declare Function DRV_CounterFreqStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterFreqStart As
PT_CounterFreqStart) As Long
Declare Function DRV_CounterFreqRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterFreqRead As
PT_CounterFreqRead) As Long
Declare Function DRV_CounterPulseStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, CounterPulseStart As
PT_CounterPulseStart) As Long
Declare Function DRV_CounterReset Lib "adsapi32.dll" (ByVal DriverHandle As Long, ByVal counter As Integer) As Long
Declare Function DRV_QCounterConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterConfig As PT_QCounterConfig)
As Long
Declare Function DRV_QCounterConfigSys Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterConfigSys As
PT_QCounterConfigSys) As Long
Declare Function DRV_QCounterStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterStart As PT_QCounterStart) As
Long
Declare Function DRV_QCounterRead Lib "adsapi32.dll" (ByVal DriverHandle As Long, QCounterRead As PT_QCounterRead) As
Long
Declare Function DRV_AlarmConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, AlarmConfig As PT_AlarmConfig) As Long
Declare Function DRV_AlarmEnable Lib "adsapi32.dll" (ByVal DriverHandle As Long, AlarmEnable As PT_AlarmEnable) As Long
Declare Function DRV_AlarmCheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, AlarmCheck As PT_AlarmCheck) As Long
Declare Function DRV_AlarmReset Lib "adsapi32.dll" (ByVal DriverHandle As Long, ByVal chan As Integer) As Long
Declare Function DRV_WDTEnable Lib "adsapi32.dll" (ByVal DriverHandle As Long, WDTEnable As PT_WDTEnable) As Long
Declare Function DRV_WDTRefresh Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_WDTReset Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_GetAddress Lib "adsapi32.dll" (lpVoid As Any) As Long

' Direct I/O Functions List
Declare Function DRV_outp Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByVal ByteData As Long) As
```

```
Long
Declare Function DRV_outpw Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByVal ByteData As Long) As
Long
Declare Function DRV_inp Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByteData As Long) As Long
Declare Function DRV_inpw Lib "adsapi32.dll" (ByVal DeviceNum As Long, ByVal Port As Integer, ByteData As Long) As Long

' High speed function declaration
Declare Function DRV_FAIWatchdogConfig Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIWatchdogConfig As
PT_FAIWatchdogConfig) As Long
Declare Function DRV_FAIIntStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIIntStart As PT_FAIIntStart) As Long
Declare Function DRV_FAIIntScanStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIIntScanStart As PT_FAIIntScanStart)
As Long
Declare Function DRV_FAIDmaStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDmaStart As PT_FAIDmaStart) As Long
Declare Function DRV_FAIDmaScanStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDmaScanStart As
PT_FAIDmaScanStart) As Long
Declare Function DRV_FAIDualDmaStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDualDmaStart As
PT_FAIDualDmaStart) As Long
Declare Function DRV_FAIDualDmaScanStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDualDmaScanStart As
PT_FAIDualDmaScanStart) As Long
Declare Function DRV_FAIIntWatchdogStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIIntWatchdogStart As
PT_FAIIntWatchdogStart) As Long
Declare Function DRV_FAIDmaWatchdogStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIDmaWatchdogStart As
PT_FAIDmaWatchdogStart) As Long
Declare Function DRV_FAICheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAICheck As PT_FAICheck) As Long
Declare Function DRV_FAIWatchdogCheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAIWatchdogCheck As
PT_FAIWatchdogCheck) As Long
Declare Function DRV_FAITransfer Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAITransfer As PT_FAITransfer) As Long
Declare Function DRV_FAIStop Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_FAOIntStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOIntStart As PT_FAOIntStart) As Long
Declare Function DRV_FAODmaStart Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAODmaStart As PT_FAODmaStart) As
Long
Declare Function DRV_FAOScale Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOScale As PT_FAOScale) As Long
Declare Function DRV_FAOLoad Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOLoad As PT_FAOLoad) As Long
Declare Function DRV_FAOCheck Lib "adsapi32.dll" (ByVal DriverHandle As Long, FAOCheck As PT_FAOCheck) As Long
Declare Function DRV_FAOStop Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_ClearOverrun Lib "adsapi32.dll" (ByVal DriverHandle As Long) As Long
Declare Function DRV_EnableEvent Lib "adsapi32.dll" (ByVal DriverHandle As Long, EnableEvent As PT_EnableEvent) As Long
Declare Function DRV_CheckEvent Lib "adsapi32.dll" (ByVal DriverHandle As Long, CheckEvent As PT_CheckEvent) As Long
Declare Function DRV_AllocateDMABuffer Lib "adsapi32.dll" (ByVal DriverHandle As Long, AllocateDMABuffer As
PT_AllocateDMABuffer) As Long
Declare Function DRV_FreeDMABuffer Lib "adsapi32.dll" (ByVal DriverHandle As Long, ByVal buffer As Long) As Long

' CAN bus function declaration
Declare Function CANPortOpen Lib "ads841.dll" (ByVal DevNum As Integer, wPort As Integer, wHostID As Integer) As Long
Declare Function CANPortClose Lib "ads841.dll" (ByVal wPort As Integer) As Long
Declare Function CANInit Lib "ads841.dll" (ByVal Port As Integer, ByVal BTR0 As Integer, ByVal BTR1 As Integer, ByVal usMask
As Byte) As Long
Declare Function CANReset Lib "ads841.dll" (ByVal Port As Integer) As Long
Declare Function CANInpb Lib "ads841.dll" (ByVal Port As Integer, ByVal offset As Integer, Data As Byte) As Long
Declare Function CANOutpb Lib "ads841.dll" (ByVal Port As Integer, ByVal offset As Integer, ByVal value As Byte) As Long
Declare Function CANSetBaud Lib "ads841.dll" (ByVal Port As Integer, ByVal BTR0 As Integer, ByVal BTR1 As Integer) As Long
Declare Function CANSetAcp Lib "ads841.dll" (ByVal Port As Integer, ByVal Acp As Integer, ByVal Mask As Integer) As Long
Declare Function CANSetOutCtrl Lib "ads841.dll" (ByVal Port As Integer, ByVal OutCtrl As Integer) As Long
Declare Function CANSetNormal Lib "ads841.dll" (ByVal Port As Integer) As Long
Declare Function CANHwReset Lib "ads841.dll" (ByVal Port As Integer) As Long
Declare Function CANSendMsg Lib "ads841.dll" (ByVal Port As Integer, ByVal TxBuf As String, ByVal Wait As Long) As Long
Declare Function CANQueryMsg Lib "ads841.dll" (ByVal Port As Integer, Ready As Long, ByVal RcvBuf As String) As Long
Declare Function CANWaitForMsg Lib "ads841.dll" (ByVal Port As Integer, ByVal RcvBuf As String, ByVal uTimeValue As Long)
As Long
Declare Function CANQueryID Lib "ads841.dll" (ByVal Port As Integer, Ready As Long, IDBuf As Byte) As Long
Declare Function CANWaitForID Lib "ads841.dll" (ByVal Port As Integer, IDBuf As Byte, ByVal uTimeValue As Long) As Long
Declare Function CANEnableMessaging Lib "ads841.dll" (ByVal Port As Integer, ByVal Type1 As Integer, ByVal Enabled As Long,
ByVal AppWnd As Long, RcvBuf As String) As Long
Declare Function CANGetEventName Lib "ads841.dll" (ByVal Port As Integer, RcvBuf As Byte) As Long
```

4.4 Global.bas

```
Global Const MaxEntries = 255
Global DeviceHandle As Long
Global ptDevGetFeatures As PT_DeviceGetFeatures
Global lpDevFeatures As DEVFEATURES
Global devicelist(0 To MaxEntries) As PT_DEVLIST
Global SubDevicelist(0 To MaxEntries) As PT_DEVLIST
Global ErrCde As Long
Global szErrMsg As String * 80
Global bRun As Boolean
Global lpDioPortMode As PT_DioSetPortMode
Global lpDioWritePort As PT_DioWritePortByte
Global lpDioReadPort As PT_DioReadPortByte
Const ModeDir = 0   ' for input mode
```

5 Code listing for the Robot Client - ROBOTClient.vbp

5.1 frmRobotClient

```
Option Explicit

Private Sub Form_Load()
        cmdSend.Enabled = False
        ' set up local port and wait for connection
        tcpClient.RemoteHost = "146.230.192.36"
        tcpClient.RemotePort = 5000  ' server port
        Call tcpClient.Connect  ' connect to RemoteHost address
End Sub

Private Sub Form_Terminate()
        Call tcpClient.Close
End Sub

Private Sub Form_Resize()
        On Error Resume Next
        Call cmdSend.Move(ScaleWidth - cmdSend.Width, 0)
        Call txtSend.Move(0, 0, ScaleWidth - cmdSend.Width)
        Call txtOutput.Move(0, txtSend.Height, ScaleWidth, _
        ScaleHeight - txtSend.Height)
End Sub

Private Sub tcpClient_Connect()
        ' when connection occurs, display a message
        cmdSend.Enabled = True
        txtOutput.Text = "Connected to Web Server at IP Address: " & _
        tcpClient.RemoteHostIP & vbCrLf & "Port #: " & _
        tcpClient.RemotePort & vbCrLf & vbCrLf
End Sub

Private Sub tcpClient_DataArrival(ByVal bytesTotal As Long)
        Dim message As String
        Call tcpClient.GetData(message)  ' get data from server
        txtOutput.Text = txtOutput.Text & message & vbCrLf & vbCrLf
        txtOutput.SelStart = Len(txtOutput.Text)
        Conveyer.Text1.Text = message
End Sub

Private Sub tcpClient_Close()
        cmdSend.Enabled = False
        Call tcpClient.Close  ' server closed, client should too
        txtOutput.Text = _
                        txtOutput.Text & "Server closed connection." & vbCrLf
                        txtOutput.SelStart = Len(txtOutput.Text)
End Sub

Private Sub tcpClient_Error(ByVal Number As Integer, _
        Description As String, ByVal Scode As Long, _
        ByVal Source As String, ByVal HelpFile As String, _
        ByVal HelpContext As Long, CancelDisplay As Boolean)
        Dim result As Integer
        result = MsgBox(Source & ": " & Description, _
                vbOKOnly, "TCP/IP Error")
        End
End Sub

Private Sub cmdSend_Click()
        ' send data to server
        Call tcpClient.SendData("ClientROB >>> " & txtSend.Text)
```

```
            txtOutput.Text = txtOutput.Text & _
            "ClientROB >>> " & txtSend.Text & vbCrLf & vbCrLf
            txtOutput.SelStart = Len(txtOutput.Text)
            txtSend.Text = ""
End Sub
```

5.2 frmPCL832

```
Option Explicit

Private DDAcount(1 To 6) As Integer
Private Counter0 As Integer
Private Counter1 As Integer
Private ExitFlag As Boolean
Private RunningFlag0 As Boolean
Private RunningFlag1 As Boolean
Private Overflows As Integer
Private TotalCounts As Double

Private Sub cmdClear0_Click()
            Dim error As Integer
            Dim i As Integer

            For i = 1 To 3
   error = PCL832_get_error(0, i)
   pcl832_set_pulse 0, i, -error
  Next i
  Overflows = 0
  TotalCounts = 0
End Sub

Private Sub cmdClear1_Click()
  Dim error As Integer
Dim i As Integer

  For i = 1 To 3
   error = PCL832_get_error(1, i)
   pcl832_set_pulse 1, i, -error
  Next i
  Overflows = 0
  TotalCounts = 0
End Sub

Private Sub cmdDisp_Click()
  Dim i As Integer
  Dim error As Integer
  Dim status As String

  Do
   For i = 1 To 3
    error = PCL832_get_error(0, i)
    status = PCL832_get_status(0, i)
    txtError(i).Text = error
    txtStatus(i).Text = status
    error = PCL832_get_error(1, i)
    status = PCL832_get_status(1, i)
    txtError(i + 3).Text = error
    txtStatus(i + 3).Text = status
    DoEvents
   Next i
  Loop Until ExitFlag
End Sub

Private Sub cmdExit_Click()
```

```
  ExitFlag = True
  PCL832_reset 0
  PCL832_reset 1
  Unload Me
End Sub

Private Sub cmdGo0_Click()
  Dim i As Integer

  If Not RunningFlag0 Then
  'start running
    cmdGo0.Caption = "Stop"
    RunningFlag0 = True
    For i = 1 To 3
      DDAcount(i) = Val(txtPulses(i).Text)
    Next i
  Else
  'stop running
    cmdGo0.Caption = "Start"
    RunningFlag0 = False
    For i = 1 To 3
      DDAcount(i) = 0
    Next i
  End If
  'tmrMain.Enabled = Not tmrMain.Enabled
End Sub

Private Sub cmdGo1_Click()
  Dim i As Integer

  If Not RunningFlag1 Then
  'start running
    cmdGo1.Caption = "Stop"
    RunningFlag1 = True
    For i = 4 To 6
      DDAcount(i) = Val(txtPulses(i).Text)
    Next i
  Else
  'stop running
    cmdGo1.Caption = "Start"
    RunningFlag1 = False
    For i = 4 To 6
      DDAcount(i) = 0
    Next i
  End If
  'tmrMain.Enabled = Not tmrMain.Enabled
End Sub

Private Sub cmdLeft_Click()
  Dim pulses(1 To 21) As Integer
  Dim i As Integer

  i = 0
  Do
    DDAcount(1) = -214
    Counter0 = 0
    Do
      DoEvents
    Loop Until Counter0 > 0
    i = i + 1
    lblTotal.Caption = i
  Loop Until i = 90
  DDAcount(i) = 0
End Sub
```

```
Private Sub cmdHide_Click()
 Me.Hide
End Sub

Private Sub cmdMoveAll_Click()
 Dim i As Integer, j As Integer
 Dim deg(1 To 6) As Integer
 Dim remainder(1 To 6) As Integer

 For j = 1 To 6
  deg(j) = Int(Abs(Val(txtDeg(j).Text))) * Sgn(Val(txtDeg(j).Text))
  Select Case j
   Case 1
    'to calibrate, alter this value for each
    ' axis below
    DDAcount(j) = 212 * Sgn(deg(j))
   Case 2
    DDAcount(j) = 212 * Sgn(deg(j))
   Case 3
    DDAcount(j) = 212 * Sgn(deg(j))
   Case 4
    DDAcount(j) = 212 * Sgn(deg(j))
   Case 5
    DDAcount(j) = 212 * Sgn(deg(j))
   Case 6
    DDAcount(j) = 212 * Sgn(deg(j))
  End Select
  If Val(txtDeg(j).Text) = 0 Then
   DDAcount(j) = 0
  End If
  'set fraction of a degree
  remainder(j) = Int(DDAcount(j) * (Val(txtDeg(j).Text) - deg(j))) * Sgn(deg(j))
  deg(j) = Abs(deg(j))
 Next j
 'keep doing this until all joints are at right angles
 Do While ((DDAcount(1) <> 0) Or (DDAcount(2) <> 0) Or (DDAcount(3) <> 0) Or (DDAcount(4) <> 0) Or (DDAcount(5) <> 0) Or
(DDAcount(6) <> 0))
  Counter0 = 0
  Counter1 = 0
  Do
   DoEvents
  'wait for next dda cycle for both cards
  Loop Until (Counter0 > 0) And (Counter1 > 0)
  For j = 1 To 6
   deg(j) = deg(j) - 1
   'if turned through whole degrees
   If deg(j) = 0 Then
    'then turn through fraction of a degree
    DDAcount(j) = remainder(j)

    'if turned through whole angle and fraction, then stop
   ElseIf deg(j) = -1 Then
    DDAcount(j) = 0
   End If
  Next j
  lblDDAC0.Caption = DDAcount(1) & " " & DDAcount(2) & " " & DDAcount(3)
  lblDDAC1.Caption = DDAcount(4) & " " & DDAcount(5) & " " & DDAcount(6)
 Loop
End Sub

Private Sub cmdMoveArm_Click(Index As Integer)
 Dim i As Integer
 Dim deg As Double
 Dim frac As Double
```

```
Select Case Index
 Case 1
   i = 0
   deg = Val(txtDeg(Index).Text)
   Do While i < Abs(deg)
     DDAcount(Index) = 212 * Sgn(deg)
     Counter0 = 0
     Do
       DoEvents
     Loop Until Counter0 > 0
     i = i + 1
     'display number of steps turned this cycle
     lblTotal.Caption = DDAcount(Index)
   Loop
   DDAcount(Index) = 0
 Case 2
   i = 0
   deg = Val(txtDeg(Index).Text)
   Do While i < Abs(deg)
     DDAcount(Index) = 212 * Sgn(deg)
     Counter0 = 0
     Do
       DoEvents
     Loop Until Counter0 > 0
     i = i + 1
     lblTotal.Caption = DDAcount(Index)
   Loop
   DDAcount(Index) = 0
 Case 3
   i = 0
   deg = Val(txtDeg(Index).Text)
   Do While i < Abs(deg)
     DDAcount(Index) = 212 * Sgn(deg)
     Counter0 = 0
     Do
       DoEvents
     Loop Until Counter0 > 0
     i = i + 1
     lblTotal.Caption = DDAcount(Index)
   Loop
   DDAcount(Index) = 0
 Case 4
   i = 0
   deg = Val(txtDeg(Index).Text)
   Do While i < Abs(deg)
     DDAcount(Index) = 212 * Sgn(deg)
     Counter1 = 0
     Do
       DoEvents
     Loop Until Counter1 > 0
     i = i + 1
     lblTotal.Caption = DDAcount(Index)
   Loop
   DDAcount(Index) = 0
 Case 5
   i = 0
   deg = Val(txtDeg(Index).Text)
   Do While i < Abs(deg)
     DDAcount(Index) = 212 * Sgn(deg)
     Counter1 = 0
     Do
       DoEvents
     Loop Until Counter1 > 0
     i = i + 1
```

```
        lblTotal.Caption = DDAcount(Index)
      Loop
      DDAcount(Index) = 0
    Case 6
      i = 0
      deg = Val(txtDeg(Index).Text)
      Do While i < Abs(deg)
        DDAcount(Index) = 212 * Sgn(deg)
        Counter1 = 0
        Do
          DoEvents
        Loop Until Counter1 > 0
        i = i + 1
        lblTotal.Caption = DDAcount(Index)
      Loop
      DDAcount(Index) = 0
  End Select
End Sub

Private Sub cmdReset0_Click()
  Dim time As Integer
  Dim i As Integer

  ExitFlag = False
  time = Reset0
  For i = 1 To 3
    DDAcount(i) = Val(txtPulses(i).Text)
  Next i
  Overflows = 0
  TotalCounts = 0
End Sub

Private Sub cmdReset1_Click()
  Dim time As Integer
  Dim i As Integer

  ExitFlag = False
  time = Reset1
  For i = 4 To 6
    DDAcount(i) = Val(txtPulses(i).Text)
  Next i
  Overflows = 0
  TotalCounts = 0
End Sub

Private Sub cmdRight_Click()
  Dim i As Integer

  i = 0
  Do
    DDAcount(1) = 214
    Counter0 = 0
    Do
      DoEvents
    Loop Until Counter0 > 0
    i = i + 1
    lblTotal.Caption = i
  Loop Until i = 90
  DDAcount(1) = 0
End Sub

Private Sub cmdtest0_Click()
  Dim i As Integer

    Counter0 = 0
  For i = 1 To 3
   DDAcount(i) = Val(txtPulses(i).Text)
   pcl832_set_pulse 0, i, DDAcount(i)
  Next i
  Do
   DoEvents
  Loop Until Counter0 = 1
  For i = 1 To 3
   DDAcount(i) = 0
   pcl832_set_pulse 0, i, DDAcount(i)
  Next i
 End Sub

 Private Sub cmdTest1_Click()
  Dim i As Integer

  Counter1 = 0
  For i = 1 To 3
   DDAcount(i + 3) = Val(txtPulses(i + 3).Text)
   pcl832_set_pulse 1, i, DDAcount(i + 3)
  Next i
  Do
   DoEvents
  Loop Until Counter1 = 1
  For i = 1 To 3
   DDAcount(i + 3) = 0
   pcl832_set_pulse 0, i, DDAcount(i + 3)
  Next i
 End Sub



 Private Sub Form_Unload(Cancel As Integer)
  ExitFlag = True
  PCL832_reset 0
  PCL832_reset 1
 End Sub

 Private Sub tmrMain0_Timer()
  Dim i As Integer

  For i = 1 To 3
   pcl832_set_pulse 0, i, DDAcount(i)
  Next i
   'TotalCounts = TotalCounts + DDAcount
  Counter0 = Counter0 + 1
  lblDDAC0.Caption = Counter0
 End Sub

 Private Function Reset0() As Integer
  Dim time As Integer
  Dim i As Integer

  PCL832_reset 0
  time = PCL832_DDA_time(0, Val(txtTime(0).Text))
  tmrMain0.Interval = time
  For i = 1 To 3
   PCL832_set_gain 0, i, Val(txtGain(i).Text)
  Next i
  PCL832_DDA_enable 0
  tmrMain0.Enabled = True
  RunningFlag0 = True
  Reset0 = time
 End Function
```

```
Private Function Reset1() As Integer
 Dim time As Integer
 Dim i As Integer

 PCL832_reset 1
 time = PCL832_DDA_time(1, Val(txtTime(1).Text))
 tmrMain1.Interval = time
 For i = 1 To 3
  PCL832_set_gain 1, i, Val(txtGain(i + 3).Text)
 Next i
 PCL832_DDA_enable 1
 tmrMain1.Enabled = True
 RunningFlag1 = True
 Reset1 = time
End Function

Private Sub tmrMain1_Timer()
 Dim i As Integer

 For i = 1 To 3
  pcl832_set_pulse 1, i, DDAcount(i + 3)
 Next i
  'TotalCounts = TotalCounts + DDAcount
 Counter1 = Counter1 + 1
 lblDDAC1.Caption = Counter1
End Sub

Private Sub tmrReset_Timer()
 tmrReset.Enabled = False
 tmrMain.Enabled = True
End Sub
```

5.3 frmInvkin

```
Option Explicit
Private ArmAng(1 To 6) As Double
Private OldArmAng(1 To 6) As Double
Private ArmLen(1 To 6) As Double
Private ArmDis(1 To 6) As Double
Private P1(0 To 3) As Double
Private dx As Double, dy As Double, dz As Double
Private counter As Integer

Private Sub cmdExit_Click()
 Unload Me
End Sub

Private Sub cmdGuess_Click()
 dx = Val(InputBox("Enter X co-ord (-9 to 9):", "Input Co-ords"))
 dy = Val(InputBox("Enter Y co-ord (-9 to 9):", "Input Co-ords"))
 dz = Val(InputBox("Enter Z co-ord (-9 to 9):", "Input Co-ords"))

 txtdebug.Text = ""
 tmrFuzzy.Enabled = True
 'Fuzzy2InvKin

End Sub

Private Sub cmdGuess2_Click()
 counter = 0
 If optFuzzy.value Then
  dx = Val(txtP(0).Text)
  dy = Val(txtP(1).Text)
```

```
    dz = Val(txtP(2).Text)
    txtdebug.Text = ""
    tmrFuzzy.Enabled = True
  Else
    ConInvKin2
  End If
End Sub

Private Sub cmdInitArm_Click()
  Form1.Show
  cmdMoveArm.Enabled = True
  cmdPlayArm.Enabled = True
End Sub

Private Sub cmdMoveArm_Click()
  Dim i As Integer

  For i = 1 To 6
    'find difference between current pos, and wanted pos
    'may need to change signs here or wiring to get
    'correct direction
    Form1.txtDeg(i).Text = Str(ArmAng(i) - OldArmAng(i))
    'update current pos
    OldArmAng(i) = ArmAng(i)
  Next i
  cmdMoveArm.Enabled = False
  cmdPlayArm.Enabled = False
  cmdInitArm.Enabled = False
  'move the arm
  Form1.cmdMoveAll
  cmdMoveArm.Enabled = True
  cmdPlayArm.Enabled = True
  cmdInitArm.Enabled = True
End Sub

Private Sub cmdPathDel_Click()
  Dim i As Integer

  'decrease number of points
  NumPathPoints = NumPathPoints - 1
  CurrentPathPoint = NumPathPoints
  'check to see if last point deleted
  If NumPathPoints = 1 Then
    cmdPathDel.Enabled = False
  End If
  'remove last point
  ReDim Preserve ArmPath(1 To NumPathPoints)

  'update angles
  For i = 1 To 6
    ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
  Next i

  'draw
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  DrawPath
  lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)
End Sub

Private Sub cmdPathEdit_Click()
  Dim i As Integer

  'update angles
  For i = 1 To 6
```

```
    ArmPath(CurrentPathPoint).Angles(i) = ArmAng(i)
  Next i

  'draw
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  DrawPath
  lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)
End Sub

Private Sub cmdPathFirst_Click()
    Dim i As Integer

  'increase current point
  CurrentPathPoint = 1
  cmdPathNext.Enabled = True
  cmdPathLast.Enabled = True
  cmdPathPrev.Enabled = False
  cmdPathFirst.Enabled = False

  'update angles
  For i = 1 To 6
    ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
  Next i

  'draw
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  DrawPath
  lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)

End Sub

Private Sub cmdPathInter_Click()
  Dim num As Integer
  Dim p(0 To 3) As Double

  num = InputBox("Enter number of intermediate steps:", "Path Interpolate")
  If num > 0 Then
     p(0) = P1(0)
     p(1) = P1(1)
     p(2) = P1(2)
     p(3) = P1(3)
    Interpolate p, num
  End If
End Sub

Private Sub cmdPathLast_Click()
    Dim i As Integer

  'increase current point
  CurrentPathPoint = NumPathPoints
  cmdPathNext.Enabled = False
  cmdPathLast.Enabled = False
  cmdPathPrev.Enabled = True
  cmdPathFirst.Enabled = True

  'update angles
  For i = 1 To 6
    ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
  Next i

  'draw
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  DrawPath
  lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)

End Sub

Private Sub cmdPathLoad_Click()
  Dim i As Integer

  CommonDialog.Filter = "Path files (*.pth)|*.pth|All files (*.*)|*.*"
  CommonDialog.FileName = ""
  CommonDialog.ShowOpen
  If CommonDialog.FileName = "" Then Exit Sub
  OpenPath CommonDialog.FileName
  MsgBox "Path Loaded"

  CurrentPathPoint = 1
  cmdPathFirst.Enabled = False
  cmdPathPrev.Enabled = False
  cmdPathEdit.Enabled = True
  If NumPathPoints > 1 Then
    cmdPathDel.Enabled = True
    cmdPathNext.Enabled = True
    cmdPathLast.Enabled = True
  Else
    cmdPathDel.Enabled = False
    cmdPathNext.Enabled = False
    cmdPathLast.Enabled = False
  End If
  'update angles
  For i = 1 To 6
    ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
  Next i

  'draw
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  DrawPath
  lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)
End Sub

Private Sub cmdPathNext_Click()
  Dim i As Integer

  'increase current point
  CurrentPathPoint = CurrentPathPoint + 1
  'check if at end
  If CurrentPathPoint = NumPathPoints Then
    cmdPathNext.Enabled = False
    cmdPathLast.Enabled = False
  End If
  cmdPathPrev.Enabled = True
  cmdPathFirst.Enabled = True

  'update angles
  For i = 1 To 6
    ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
  Next i

  'draw
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  DrawPath
  lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)
End Sub
```

```
Private Sub cmdPathPlay_Click()
 Dim i As Integer

 CurrentPathPoint = 1
 'update angles
 For i = 1 To 6
  ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
 Next i

 'draw
 DrawAxis
 DrawArm ArmAng, ArmLen, ArmDis
 DrawPath
 lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)

 'start animation
 tmrAnim.Enabled = True
End Sub

Private Sub cmdPathPrev_Click()
 Dim i As Integer

 'decrease current point
 CurrentPathPoint = CurrentPathPoint - 1
 'check if at beginning
 If CurrentPathPoint = 1 Then
  cmdPathPrev.Enabled = False
  cmdPathFirst.Enabled = False
 End If
 cmdPathNext.Enabled = True
 cmdPathLast.Enabled = True
 'update angles
 For i = 1 To 6
  ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
 Next i
 'draw
 DrawAxis
 DrawArm ArmAng, ArmLen, ArmDis
 DrawPath
 lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)
End Sub

Private Sub cmdPathReset_Click()
 ReDim ArmPath(1 To 1)
 NumPathPoints = 0
 CurrentPathPoint = 0
 lblPath.Caption = "0 of 0"
 cmdPathFirst.Enabled = False
 cmdPathPrev.Enabled = False
 cmdPathNext.Enabled = False
 cmdPathLast.Enabled = False
 cmdPathDel.Enabled = False
 cmdPathEdit.Enabled = False
 DrawAxis
 DrawArm ArmAng, ArmLen, ArmDis
End Sub

Private Sub cmdPathSave_Click()
 CommonDialog.Filter = "Path files (*.pth)|*.pth|All files (*.*)|*.*"
 CommonDialog.FileName = ""
 CommonDialog.ShowSave
 If CommonDialog.FileName = "" Then Exit Sub
 SavePath CommonDialog.FileName
 MsgBox "Path Saved"
End Sub
```

```
Private Sub cmdPathStore_Click()
  Dim i As Integer

  'add new point to path
  NumPathPoints = NumPathPoints + 1
  CurrentPathPoint = NumPathPoints
  ReDim Preserve ArmPath(1 To NumPathPoints)

  'store angle information in point
  For i = 1 To 6
    ArmPath(CurrentPathPoint).Angles(i) = ArmAng(i)
  Next i
  For i = 0 To 3
    ArmPath(CurrentPathPoint).Pos(i) = P1(i)
  Next i

  'redraw with path shown
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  DrawPath
  lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)

  'enable buttons
  If NumPathPoints > 1 Then
    cmdPathFirst.Enabled = True
    cmdPathPrev.Enabled = True
    cmdPathNext.Enabled = False
    cmdPathLast.Enabled = False
    cmdPathDel.Enabled = True
  End If
  cmdPathEdit.Enabled = True
End Sub

Private Sub cmdPlayArm_Click()
  Dim i As Integer

  CurrentPathPoint = 1
  Do
    'update angles
    For i = 1 To 6
      ArmAng(i) = ArmPath(CurrentPathPoint).Angles(i)
    Next i

    'draw
    DrawAxis
    DrawArm ArmAng, ArmLen, ArmDis
    DrawPath
    lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)

    'start motion
    cmdMoveArm_Click
    CurrentPathPoint = CurrentPathPoint + 1
    'continue until whole moion completed
  Loop Until CurrentPathPoint > NumPathPoints
End Sub

Private Sub cmdReset_Click()
  counter = 0
  ArmStart
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
End Sub

Private Sub cmdRotACW_Click(Index As Integer)
  ArmAng(Index) = ArmAng(Index) - 5
```

```
' Select Case Index
'  Case 1:
'     If ArmAng(Index) < -160 Then ArmAng(Index) = -160
'  Case 2:
'     If ArmAng(Index) < -225 Then ArmAng(Index) = -225
'  Case 3:
'     If ArmAng(Index) < -45 Then ArmAng(Index) = -45
'  Case 4:
'     If ArmAng(Index) < -110 Then ArmAng(Index) = -110
'  Case 5:
'     If ArmAng(Index) < -100 Then ArmAng(Index) = -100
'  Case 6:
'     If ArmAng(Index) < -266 Then ArmAng(Index) = -265
' End Select
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
End Sub

Private Sub cmdRotCW_Click(Index As Integer)
  ArmAng(Index) = ArmAng(Index) + 5
' Select Case Index
'  Case 1:
'     If ArmAng(Index) > 160 Then ArmAng(Index) = 160
'  Case 2:
'     If ArmAng(Index) > 45 Then ArmAng(Index) = 45
'  Case 3:
'     If ArmAng(Index) > 225 Then ArmAng(Index) = 225
'  Case 4:
'     If ArmAng(Index) > 170 Then ArmAng(Index) = 170
'  Case 5:
'     If ArmAng(Index) > 100 Then ArmAng(Index) = 100
'  Case 6:
'     If ArmAng(Index) > 266 Then ArmAng(Index) = 265
' End Select
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
End Sub

Private Sub cmdStop_Click()
  tmrFuzzy.Enabled = Not tmrFuzzy.Enabled
End Sub

Private Sub Form_Load()
  picXY.Scale (-10, 10)-(10, -10)
  picXZ.Scale (-10, 10)-(10, -10)
  picYZ.Scale (10, 10)-(-10, -10)

  MakeFuzSets

  ArmStart
  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis

  'path initialization
  NumPathPoints = 0
  CurrentPathPoint = 0
  lblPath.Caption = "0 of 0"
  cmdPathFirst.Enabled = False
  cmdPathPrev.Enabled = False
  cmdPathNext.Enabled = False
  cmdPathLast.Enabled = False
End Sub

Private Sub DrawAxis()
  picXY.Cls
```

```
 picXZ.Cls
 picYZ.Cls

 'blue for X
 picXY.ForeColor = vbBlue
 picXZ.ForeColor = vbBlue
 picXY.Line (0, 0)-(2, 0)
 picXY.Line (2, 0)-(1, 1)
 picXZ.Line (0, 0)-(2, 0)
 picXZ.Line (2, 0)-(1, 1)

 'red for Y
 picXY.ForeColor = vbRed
 picYZ.ForeColor = vbRed
 picXY.Line (0, 0)-(0, 2)
 picXY.Line (0, 2)-(-1, 1)
 picYZ.Line (0, 0)-(2, 0)
 picYZ.Line (2, 0)-(1, 1)

 'yellow for Z
 picXZ.ForeColor = vbYellow
 picYZ.ForeColor = vbYellow
 picXZ.Line (0, 0)-(0, 2)
 picXZ.Line (0, 2)-(-1, 1)
 picYZ.Line (0, 0)-(0, 2)
 picYZ.Line (0, 2)-(-1, 1)

 picXY.Circle (0, 0), 5

 'reset colors
 picXY.ForeColor = vbBlack
 picXZ.ForeColor = vbBlack
 picYZ.ForeColor = vbBlack

End Sub

Private Sub DrawVect(M1() As Double, M2() As Double)
 picXY.Line (M1(0), M1(1))-(M2(0), M2(1))
 picXZ.Line (M1(0), M1(2))-(M2(0), M2(2))
 picYZ.Line (M1(1), M1(2))-(M2(1), M2(2))
End Sub

Private Sub DrawArm(Angles() As Double, lengths() As Double, dis() As Double)
 Dim P2(0 To 3) As Double
 Dim o(0 To 3) As Double
 Dim temp2(0 To 3) As Double

 Dim T1(0 To 15) As Double
 Dim T2(0 To 15) As Double
 Dim temp(0 To 15) As Double

 Dim r As Double
 Dim i As Integer

 'MakeT ArmAng(2), 0, ArmAng(1), T1
 Make0A1 ArmAng(1), ArmLen(1), ArmDis(1), T1(0)
 Make1A2 ArmAng(1), ArmLen(1), ArmDis(1), T2(0)
 'temp = t1*t2 = 0A1*1A2
 MatrixMult4b4 T1(0), T2(0), temp(0)
 'got 0A2
 GetCoords temp(0), 4, P1(0)
 DrawPoint o
 DrawPoint P1
 DrawVect o, P1
```

```
Make2A3 ArmAng(1), ArmLen(1), ArmDis(1), T2(0)
't1 = temp*t2 = 0A2*2A3
MatrixMult4b4 temp(0), T2(0), T1(0)
'got 0A3
GetCoords T1(0), 4, P2(0)
DrawPoint P2
DrawVect P1, P2

Make3A4 ArmAng(1), ArmLen(1), ArmDis(1), T2(0)
'temp = t1*t2 = 0A3*3A4
MatrixMult4b4 T1(0), T2(0), temp(0)
'got 0A4
GetCoords temp(0), 4, P1(0)
DrawPoint P1
DrawVect P2, P1

Make4A5 ArmAng(1), ArmLen(1), ArmDis(1), T2(0)
't1 = temp*t2 = 0A4*4A5
MatrixMult4b4 temp(0), T2(0), T1(0)
'got 0A5
GetCoords T1(0), 4, P2(0)
DrawPoint P2
DrawVect P1, P2
'for conInvKin2 pos

Make5A6 ArmAng(1), ArmLen(1), ArmDis(1), T2(0)
'temp = t1*t2 = 0A5*5A6
MatrixMult4b4 T1(0), T2(0), temp(0)
'got 0a6
GetCoords temp(0), 4, P1(0)
DrawPoint P1
DrawVect P2, P1

'draw manipulator
o(2) = 1
o(1) = 1
o(3) = 1
'get 1st point
MatrixMult4b1 temp(0), o(0), P2(0)
DrawVect P1, P2
o(1) = -1
'get 2nd point
MatrixMult4b1 temp(0), o(0), P2(0)
DrawVect P1, P2

r = Sqr(P1(0) ^ 2 + P1(1) ^ 2 + P1(2) ^ 2)
lblCoords.Caption = "P: (" & Format(P1(0), "##0.000") & _
            "," & Format(P1(1), "##0.000") & _
            "," & Format(P1(2), "##0.000") & ")" & vbNewLine & _
            "r =" & Format(r, "##0.00") & vbNewLine
GetCoords temp(0), 1, temp2(0)
lblCoords.Caption = lblCoords.Caption & "N: (" & Format(temp2(0), "##0.0") & _
            "," & Format(temp2(1), "##0.0") & _
            "," & Format(temp2(2), "##0.0") & ")" & vbNewLine
GetCoords temp(0), 2, temp2(0)
lblCoords.Caption = lblCoords.Caption & "O: (" & Format(temp2(0), "##0.0") & _
            "," & Format(temp2(1), "##0.0") & _
            "," & Format(temp2(2), "##0.0") & ")" & vbNewLine
GetCoords temp(0), 3, temp2(0)
lblCoords.Caption = lblCoords.Caption & "A: (" & Format(temp2(0), "##0.0") & _
            "," & Format(temp2(1), "##0.0") & _
            "," & Format(temp2(2), "##0.0") & ")"
For i = 1 To 6
  lblAng(i - 1).Caption = "A" & Str(i) & ": " & Format(ArmAng(i), "##0.00#")
Next i
```

```
End Sub

Private Sub DrawPoint(M() As Double)
 picXY.DrawWidth = 2
 picXZ.DrawWidth = 2
 picYZ.DrawWidth = 2

 picXY.PSet (M(0), M(1))
 picXZ.PSet (M(0), M(2))
 picYZ.PSet (M(1), M(2))

 picXY.DrawWidth = 1
 picXZ.DrawWidth = 1
 picYZ.DrawWidth = 1
End Sub

Private Sub ArmStart()
 'joint angles
 ArmAng(1) = 0
 ArmAng(2) = 0
 ArmAng(3) = 90
 ArmAng(4) = 0
 ArmAng(5) = 0
 ArmAng(6) = 0

 'joint parameters ai
 ArmLen(1) = 0
 ArmLen(2) = 4.318
 ArmLen(3) = -0.2032
 ArmLen(4) = 0
 ArmLen(5) = 0
 ArmLen(6) = 0

 'joint parameters di
 ArmDis(1) = 0
 ArmDis(2) = 1.4909
 ArmDis(3) = 0
 ArmDis(4) = 4.3307
 ArmDis(5) = 0
 ArmDis(6) = 0.5625
End Sub

Private Sub DrawSet(FuzSet() As Double)
 Dim i As Integer

 For i = -(Range - Hangledist) To Range Step Hangledist
  picOut.Line (i - Hangledist, FuzSet(((i - Hangledist) + Range) / Hangledist))-(i, FuzSet((i + Range) / Hangledist))
 Next i
End Sub

Private Sub Fuzzy2InvKin()
'sugeno model
 Dim temp As Double
 Dim ang1error As Double
 Dim ang2 As Double
 Dim ang3 As Double
 Dim lenError As Double
 Dim p(0 To 3) As Double
 Dim T(0 To 15) As Double

' txtdebug.Text = ""

 'get current transform
 MakeT06 ArmAng(1), ArmLen(1), ArmDis(1), T(0)
 'get current end-point
 GetCoords T(0), 4, p(0)

 'calculate errors
 ang1error = Rad2Deg(arcTan(dy, dx)) - Rad2Deg(arcTan(p(1), p(0)))
 'lenError = Sqr((dx - P1(0)) ^ 2 + (dy - P1(1)) ^ 2 + (dz - P1(2)) ^ 2)
 lenError = Sqr(dx ^ 2 + dy ^ 2 + dz ^ 2) - Sqr(P1(0) ^ 2 + P1(1) ^ 2 + P1(2) ^ 2)

 temp = Sqr((dx - P1(0)) ^ 2 + (dy - P1(1)) ^ 2 + (dz - P1(2)) ^ 2)
  txtdebug.Text = txtdebug.Text & "error = " & Format(temp, "#0.000") & vbNewLine

 If (Abs(dx - P1(0)) < tol) And (Abs(dy - P1(1)) < tol) And (Abs(dz - P1(2)) < tol) Then
  tmrFuzzy.Enabled = False
  txtdebug.Text = txtdebug.Text & "Solved in " & Str(counter) & " iterations" & vbNewLine
  txtdebug.Text = txtdebug.Text & "resulting co-ords:" & vbNewLine _
        & "x:" & Format(P1(0), "##0.00#") & vbNewLine _
        & "y:" & Format(P1(1), "##0.00#") & vbNewLine _
        & "z:" & Format(P1(2), "##0.00#") & vbNewLine _
        & "desired position: " & vbNewLine _
        & "x:" & Format(dx, "##0.00#") & vbNewLine _
        & "y:" & Format(dy, "##0.00#") & vbNewLine _
        & "z:" & Format(dz, "##0.00#")

  counter = 0
  Exit Sub
 End If
 counter = counter + 1

 'calculate angle 1
 ArmAng(1) = ArmAng(1) + FindAngleInc(ang1error)
 txtdebug.Text = txtdebug.Text & "Angle 1 is now " & Str(ArmAng(1)) & vbNewLine

 If Abs(ArmAng(1)) > 360 Then ArmAng(1) = -ArmAng(1)

 'calculate angle3
 FindLenInc lenError * 10, ang2, ang3
 ArmAng(3) = ArmAng(3) - ang3
 txtdebug.Text = txtdebug.Text & "Angle 3 is now " & Str(ArmAng(3)) & vbNewLine

 If Abs(ArmAng(3)) > 360 Then ArmAng(3) = -ArmAng(3)

  'get current transform
 MakeT06 ArmAng(1), ArmLen(1), ArmDis(1), T(0)
 'get current end-point
 'etCoords T(0), 4, p(0)

 ang2 = Rad2Deg(arcTan(dz, Sqr(dx ^ 2 + dy ^ 2))) - Rad2Deg(arcTan(p(2), Sqr(p(0) ^ 2 + p(1) ^ 2)))
 ArmAng(2) = ArmAng(2) - FindAngleInc(ang2)
 txtdebug.Text = txtdebug.Text & "Angle 2 is now " & Str(ArmAng(2)) & vbNewLine

 If Abs(ArmAng(2)) > 360 Then ArmAng(2) = -ArmAng(2)

 'draw
 DrawAxis
 DrawArm ArmAng, ArmLen, ArmDis
 temp = Sqr((dx - P1(0)) ^ 2 + (dy - P1(1)) ^ 2 + (dz - P1(2)) ^ 2)
 txtdebug.Text = txtdebug.Text & "resulting co-ords:" & vbNewLine _
        & "x:" & Format(P1(0), "##0.00#") & vbNewLine _
        & "y:" & Format(P1(1), "##0.00#") & vbNewLine _
        & "z:" & Format(P1(2), "##0.00#") & vbNewLine _
        & "strt line error is:" & Format(temp, "##0.0#")
End Sub

Private Sub ConInvKin2()
  Dim p(0 To 3) As Double
  Dim N(0 To 3) As Double
  Dim a(0 To 3) As Double
  Dim o(0 To 3) As Double
  Dim temp As Double, temp2 As Double
  Dim i As Integer
  Dim z3(0 To 3) As Double
  Dim res(0 To 3) As Double
  Dim T03(0 To 15) As Double
  Dim arm1 As Integer, arm2 As Integer, arm3 As Integer

  For i = 0 To 3
   p(i) = Val(txtP(i).Text)
   N(i) = Val(txtN(i).Text)
   a(i) = Val(txtA(i).Text)
   o(i) = Val(txtO(i).Text)
  Next i
  arm1 = chkRight.value * 2 - 1
  arm2 = chkAbove.value * 2 - 1
  arm3 = chkDown.value * 2 - 1

  For i = 0 To 2
   p(i) = p(i) - ArmDis(6) * a(i)
  Next i

  ArmAng(1) = Rad2Deg(INVKIN_FindAngle1(p(0), ArmDis(2), arm1))
  ArmAng(2) = Rad2Deg(INVKIN_FindAngle2(p(0), ArmDis(1), ArmLen(1), arm1, arm2))
  ArmAng(3) = Rad2Deg(INVKIN_FindAngle3(p(0), ArmDis(1), ArmLen(1), arm1, arm2))

  MakeT03 ArmAng(1), ArmLen(1), ArmDis(1), T03(0)
  GetCoords T03(0), 3, z3(0)
  X_Prod z3(0), a(0), res(0)
  If VectMag(res) = 0 Then
   temp = 0
  Else
   temp2 = DotProd(o(0), res(0))
   If temp2 = 0 Then
    temp2 = DotProd(N(0), res(0))
   End If
   temp = temp2 / VectMag(res)
  End If
  i = Sgn(temp) * arm3
  ArmAng(4) = Rad2Deg(INVKIN_FindAngle4(a(0), ArmAng(1), i))
  ArmAng(5) = Rad2Deg(INVKIN_FindAngle5(a(0), ArmAng(1)))
  ArmAng(6) = Rad2Deg(INVKIN_FindAngle6(N(0), o(0), ArmAng(1)))

  DrawAxis
  DrawArm ArmAng, ArmLen, ArmDis
  For i = 0 To 3
   p(i) = Val(txtP(i).Text)
  Next i
  temp = Sqr((p(0) - P1(0)) ^ 2 + (p(1) - P1(1)) ^ 2 + (p(2) - P1(2)) ^ 2)
  txtdebug.Text = "resulting co-ords:" & vbNewLine _
          & "x:" & Format(P1(0), "##0.0#") & vbNewLine _
          & "y:" & Format(P1(1), "##0.0#") & vbNewLine _
          & "z:" & Format(P1(2), "##0.0#") & vbNewLine _
          & "strt line error is:" & Format(temp, "##0.0#")
End Sub

Private Function VectMag(V() As Double) As Double
 VectMag = Sqr(V(0) ^ 2 + V(1) ^ 2 + V(2) ^ 2)
End Function

Private Sub Form_Unload(Cancel As Integer)
 Unload Form1
End Sub
```

```
Private Sub tmrAnim_Timer()
 Dim i As Integer
 Dim angErr As Double
 Dim AtPos(1 To 6) As Boolean

 'update angles
 For i = 1 To 6
  If Not AtPos(i) Then
   angErr = ArmPath(CurrentPathPoint).Angles(i) - ArmAng(i)
   'added to remove circular movement
   If angErr < -180 Then
    ArmPath(CurrentPathPoint).Angles(i) = ArmPath(CurrentPathPoint).Angles(i) + 360
   End If
   If angErr > 180 Then
    ArmPath(CurrentPathPoint).Angles(i) = ArmPath(CurrentPathPoint).Angles(i) - 360
   End If
   If angErr > 5 Then
    ArmAng(i) = ArmAng(i) + 5
   ElseIf angErr < -5 Then
    ArmAng(i) = ArmAng(i) - 5
   Else
    ArmAng(i) = ArmAng(i) + angErr
    AtPos(i) = True
   End If
  End If
 Next i

 'draw
 DrawAxis
 DrawArm ArmAng, ArmLen, ArmDis
 DrawPath
 lblPath.Caption = Str(CurrentPathPoint) & " of " & Str(NumPathPoints)

 'if at pos then move to next point
 If AtPos(1) And AtPos(2) And AtPos(3) And AtPos(4) And AtPos(5) And AtPos(6) Then
  'check if at end of path
  If CurrentPathPoint >= NumPathPoints Then
   tmrAnim.Enabled = False
  Else
   'increase path point
   CurrentPathPoint = CurrentPathPoint + 1
  End If
 End If

End Sub

Private Sub tmrConv_Timer()
 ConInvKin2
End Sub

Private Sub tmrFuzzy_Timer()
 Fuzzy3InvKin
End Sub

Private Sub DrawPath()
 Dim i As Integer

 picXY.ForeColor = vbGreen
 picXZ.ForeColor = vbGreen
 picYZ.ForeColor = vbGreen

 For i = 2 To NumPathPoints
  picXY.Line (ArmPath(i - 1).Pos(0), ArmPath(i - 1).Pos(1))- _
        (ArmPath(i).Pos(0), ArmPath(i).Pos(1))
  picXZ.Line (ArmPath(i - 1).Pos(0), ArmPath(i - 1).Pos(2))- _
```

```
        (ArmPath(i).Pos(0), ArmPath(i).Pos(2))
  picYZ.Line (ArmPath(i - 1).Pos(1), ArmPath(i - 1).Pos(2))- _
        (ArmPath(i).Pos(1), ArmPath(i).Pos(2))
 Next i

 picXY.ForeColor = vbBlack
 picXZ.ForeColor = vbBlack
 picYZ.ForeColor = vbBlack

End Sub

Public Sub Interpolate(NewPoint() As Double, NumSteps As Integer)
 Dim i As Integer
 Dim delX As Double, delY As Double, delZ As Double

 txtdebug.Text = ""
 If optFuzzy.value Then
  For i = 1 To NumSteps
   delX = (NewPoint(0) - ArmPath(NumPathPoints).Pos(0)) / (NumSteps - i + 1)
   delY = (NewPoint(1) - ArmPath(NumPathPoints).Pos(1)) / (NumSteps - i + 1)
   delZ = (NewPoint(2) - ArmPath(NumPathPoints).Pos(2)) / (NumSteps - i + 1)

   dx = ArmPath(NumPathPoints).Pos(0) + delX
   dy = ArmPath(NumPathPoints).Pos(1) + delY
   dz = ArmPath(NumPathPoints).Pos(2) + delZ
   tmrFuzzy.Enabled = True
   Do
    DoEvents
   Loop Until Not tmrFuzzy.Enabled
   cmdPathStore_Click
  Next i
 Else
  For i = 1 To NumSteps
   txtdebug.Text = txtdebug.Text & "point " & Str(i) & vbNewLine
   delX = (NewPoint(0) - ArmPath(NumPathPoints).Pos(0)) / (NumSteps - i + 1)
   txtdebug.Text = txtdebug.Text & "dx = " & Format(delX, "#0.000") & vbNewLine
   delY = (NewPoint(1) - ArmPath(NumPathPoints).Pos(1)) / (NumSteps - i + 1)
   txtdebug.Text = txtdebug.Text & "dy = " & Format(delY, "#0.000") & vbNewLine
   delZ = (NewPoint(2) - ArmPath(NumPathPoints).Pos(2)) / (NumSteps - i + 1)
   txtdebug.Text = txtdebug.Text & "dz = " & Format(delZ, "#0.000") & vbNewLine

   txtP(0).Text = Format((ArmPath(NumPathPoints).Pos(0) + delX), "#0.000")
   txtP(1).Text = Format((ArmPath(NumPathPoints).Pos(1) + delY), "#0.000")
   txtP(2).Text = Format((ArmPath(NumPathPoints).Pos(2) + delZ), "#0.000")
   txtdebug.Text = txtdebug.Text & "Desired co-ords:" & vbNewLine _
         & "(" & txtP(0).Text & "," & txtP(1).Text _
         & "," & txtP(2).Text & ")" & vbNewLine

   ConInvKin2
   'tmrConv.Enabled = True
   'Do
   ' DoEvents
   'Loop Until Not tmrConv.Enabled
   cmdPathStore_Click
   txtdebug.Text = txtdebug.Text & "Resulting co-ords:" & vbNewLine _
         & "(" & txtP(0).Text & "," & txtP(1).Text _
         & "," & txtP(2).Text & ")" & vbNewLine
  Next i
 End If
End Sub

Private Sub Fuzzy3InvKin()
 'sugeno model
 Dim temp As Double, temp2 As Double
 Dim anglerror As Double
Dim ang2 As Double
Dim ang3 As Double
Dim lenError As Double
Dim p(0 To 3) As Double
Dim T(0 To 15) As Double
Dim N(0 To 3) As Double
Dim a(0 To 3) As Double
Dim o(0 To 3) As Double
Dim i As Integer
Dim z3(0 To 3) As Double
Dim res(0 To 3) As Double
Dim T03(0 To 15) As Double
Dim arm3 As Integer

' txtdebug.Text = ""

'get current transform
MakeT06 ArmAng(1), ArmLen(1), ArmDis(1), T(0)
'get current end-point
GetCoords T(0), 4, p(0)

'calculate errors
ang1error = Rad2Deg(arcTan(dy, dx)) - Rad2Deg(arcTan(p(1), p(0)))
'lenError = Sqr((dx - P1(0)) ^ 2 + (dy - P1(1)) ^ 2 + (dz - P1(2)) ^ 2)
lenError = Sqr(dx ^ 2 + dy ^ 2 + dz ^ 2) - Sqr(P1(0) ^ 2 + P1(1) ^ 2 + P1(2) ^ 2)

temp = Sqr((dx - P1(0)) ^ 2 + (dy - P1(1)) ^ 2 + (dz - P1(2)) ^ 2)
txtdebug.Text = txtdebug.Text & "iteration " & counter & " "
 txtdebug.Text = txtdebug.Text & "error = " & Format(temp, "#0.000") & vbNewLine

If (Abs(dx - P1(0)) < tol) And (Abs(dy - P1(1)) < tol) And (Abs(dz - P1(2)) < tol) Then
 tmrFuzzy.Enabled = False
 txtdebug.Text = txtdebug.Text & "Solved in " & Str(counter) & " iterations" & vbNewLine
 txtdebug.Text = txtdebug.Text & "resulting co-ords:" & vbNewLine _
        & "x:" & Format(P1(0), "##0.00#") & vbNewLine _
        & "y:" & Format(P1(1), "##0.00#") & vbNewLine _
        & "z:" & Format(P1(2), "##0.00#") & vbNewLine _
        & "desired position: " & vbNewLine _
        & "x:" & Format(dx, "##0.00#") & vbNewLine _
        & "y:" & Format(dy, "##0.00#") & vbNewLine _
        & "z:" & Format(dz, "##0.00#")

 counter = 0
 Exit Sub
End If
counter = counter + 1

'calculate angle 1
ArmAng(1) = ArmAng(1) + FindAngleInc(ang1error)

If Abs(ArmAng(1)) > 360 Then ArmAng(1) = -ArmAng(1)

'calculate angle3
FindLenInc lenError * 10, ang2, ang3
ArmAng(3) = ArmAng(3) - ang3

If Abs(ArmAng(3)) > 360 Then ArmAng(3) = -ArmAng(3)

 'get current transform
MakeT06 ArmAng(1), ArmLen(1), ArmDis(1), T(0)
'get current end-point
'etCoords T(0), 4, p(0)

ang2 = Rad2Deg(arcTan(dz, Sqr(dx ^ 2 + dy ^ 2))) - Rad2Deg(arcTan(p(2), Sqr(p(0) ^ 2 + p(1) ^ 2)))
ArmAng(2) = ArmAng(2) - FindAngleInc(ang2)
```

```
If Abs(ArmAng(2)) > 360 Then ArmAng(2) = -ArmAng(2)

'////////////


For i = 0 To 3
  N(i) = Val(txtN(i).Text)
  a(i) = Val(txtA(i).Text)
  o(i) = Val(txtO(i).Text)
Next i
arm3 = chkDown.value * 2 - 1

 MakeT03 ArmAng(1), ArmLen(1), ArmDis(1), T03(0)
 GetCoords T03(0), 3, z3(0)
 X_Prod z3(0), a(0), res(0)
 If VectMag(res) = 0 Then
   temp = 0
 Else
   temp2 = DotProd(o(0), res(0))
   If temp2 = 0 Then
     temp2 = DotProd(N(0), res(0))
   End If
   temp = temp2 / VectMag(res)
 End If
 i = Sgn(temp) * arm3
 ArmAng(4) = Rad2Deg(INVKIN_FindAngle4(a(0), ArmAng(1), i))
 ArmAng(5) = Rad2Deg(INVKIN_FindAngle5(a(0), ArmAng(1)))
 ArmAng(6) = Rad2Deg(INVKIN_FindAngle6(N(0), o(0), ArmAng(1)))
 For i = 1 To 6
   txtdebug.Text = txtdebug.Text & "Angle " & i & " is now " & Format(ArmAng(i), "##0.00#") & vbNewLine
 Next i

 '////////
 'draw
 DrawAxis
 DrawArm ArmAng, ArmLen, ArmDis
End Sub
```

5.4 PCL832.bas

```
Option Explicit
'kallyliu@iafrica.com
'email PCB designer

Declare Sub IO_outport Lib "IO.dll" (ByVal port As Integer, ByVal value As Integer)
Declare Function IO_inport Lib "IO.dll" (ByVal port As Integer) As Integer

Private Const ADDR1 As Long = &H240
Private Const ADDR2 As Long = &H220

Public Sub PCL832_reset(card As Integer)
 If card = 0 Then
   IO_outport ADDR1 + &H1A, 0
 Else
   IO_outport ADDR2 + &H1A, 0
 End If
End Sub

Public Sub PCL832_DDA_enable(card As Integer)
 If card = 0 Then
   IO_outport ADDR1 + &H18, 0
```

```
  Else
   IO_outport ADDR2 + &H18, 0
  End If
 End Sub

 Public Function PCL832_DDA_time(card As Integer, time As Integer) As Integer
  Dim temp As Integer

  temp = Int(time / 0.512)
  If card = 0 Then
   IO_outport ADDR1 + &H4, temp
  Else
   IO_outport ADDR2 + &H4, temp
  End If
  PCL832_DDA_time = Int(temp * 0.512)
 End Function

 Public Sub PCL832_set_gain(card As Integer, channel As Integer, gain As Integer)
  Dim temp As Integer

  Select Case channel
   Case 1: temp = &H2
   Case 2: temp = &HA
   Case 3: temp = &H12
   Case Else:  Exit Sub
  End Select
  If card = 0 Then
   IO_outport ADDR1 + temp, gain
  Else
   IO_outport ADDR2 + temp, gain
  End If
 End Sub

 Public Function PCL832_get_error(card As Integer, channel As Integer) As Integer
  Dim temp As Integer
  Dim error As Integer

  Select Case channel
   Case 1: temp = &H0
   Case 2: temp = &H8
   Case 3: temp = &H10
   Case Else: Exit Function
  End Select
  If card = 0 Then
   error = IO_inport(ADDR1 + temp)
  Else
   error = IO_inport(ADDR2 + temp)
  End If
  If error > 0 Then
   PCL832_get_error = error - &H7000
  Else
   PCL832_get_error = error
  End If
 End Function

 Public Function PCL832_get_status(card As Integer, channel As Integer) As String
  Dim temp As Integer
  Dim status As Integer
  Dim out As String

  Select Case channel
   Case 1: temp = &H2
   Case 2: temp = &HA
   Case 3: temp = &H12
   Case Else: Exit Function
  End Select
  If card = 0 Then
    status = IO_inport(ADDR1 + temp)
  Else
    status = IO_inport(ADDR2 + temp)
  End If
  'out = Str((status And &H8)) & "-" & Str((status And &H2)) & Str((status And 1))
  out = Hex(status)
  PCL832_get_status = out
End Function

Public Sub pcl832_set_pulse(card As Integer, channel As Integer, pulses As Integer)
  Dim temp As Integer
  Dim t_pulse As Integer

  Select Case channel
    Case 1: temp = &H0
    Case 2: temp = &H8
    Case 3: temp = &H10
    Case Else: Exit Sub
  End Select
  If pulses < 0 Then
    t_pulse = -pulses
    t_pulse = t_pulse + &H8000
  Else
    t_pulse = pulses
  End If
  If card = 0 Then
    IO_outport ADDR1 + temp, t_pulse
  Else
    IO_outport ADDR2 + temp, t_pulse
  End If
End Sub
```

5.5 Def2.bas

```
Option Explicit

' Maths defs
Declare Sub MatrixMult4b4 Lib "Maths.dll" Alias "MATRIXMULT4B4" (a As Double, b As Double, c As Double)
Declare Sub MatrixMult4b1 Lib "Maths.dll" Alias "MATRIXMULT4B1" (a As Double, b As Double, c As Double)
Declare Sub X_Prod Lib "Maths.dll" Alias "X_PROD" (a As Double, b As Double, c As Double)
Declare Function DotProd Lib "Maths.dll" Alias "DOTPROD" (a As Double, b As Double) As Double

Declare Function Rad2Deg Lib "Maths.dll" Alias "RAD2DEG" (ByVal r As Double) As Double
Declare Function Deg2Rad Lib "Maths.dll" Alias "DEG2RAD" (ByVal d As Double) As Double

Declare Sub Mat Lib "Maths.dll" Alias "MAT" (M As Double, ByVal r As Long, ByVal c As Long, ByVal V As Double)

Declare Sub MakeTArm Lib "Maths.dll" Alias "MAKETARM" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub MakeT1 Lib "Maths.dll" Alias "MAKET1" (ang As Double, leng As Double, dis As Double, M As Double)

Declare Sub Make0A1 Lib "Maths.dll" Alias "MAKE0A1" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub Make1A2 Lib "Maths.dll" Alias "MAKE1A2" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub Make2A3 Lib "Maths.dll" Alias "MAKE2A3" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub Make3A4 Lib "Maths.dll" Alias "MAKE3A4" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub Make4A5 Lib "Maths.dll" Alias "MAKE4A5" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub Make5A6 Lib "Maths.dll" Alias "MAKE5A6" (ang As Double, leng As Double, dis As Double, M As Double)

Declare Sub MakeT03 Lib "Maths.dll" Alias "MAKET03" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub MakeT36 Lib "Maths.dll" Alias "MAKET36" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub MakeT06 Lib "Maths.dll" Alias "MAKET06" (ang As Double, leng As Double, dis As Double, M As Double)
Declare Sub GetCoords Lib "Maths.dll" Alias "GETCOORDS" (T As Double, ByVal col As Long, M As Double)
Declare Function arcTan Lib "Maths.dll" Alias "ARCTAN" (ByVal y As Double, ByVal x As Double) As Double


' Fuzzy Logic Defs
Declare Sub SFUZMakeAngleSet Lib "sfuzzy.dll" (DescSet As Double, FuzSet As Double)
Declare Function SFUZevalAngleSet Lib "sfuzzy.dll" (ByVal x As Double, FuzSet As Double) As Double
Declare Sub SFUZAngleSetAND Lib "sfuzzy.dll" (Aset As Double, Bset As Double, Cset As Double)
Declare Sub SFUZAngleSetOR Lib "sfuzzy.dll" (Aset As Double, Bset As Double, Cset As Double)
Declare Sub SFUZminTnormAngle2var Lib "sfuzzy.dll" (ByVal x As Double, Aset As Double, _
                                                   ByVal y As Double, Bset As Double, _
                                                   Cset As Double, Mset As Double)
Declare Sub SFUZmaxCnormAngle2var Lib "sfuzzy.dll" (C1 As Double, C2 As Double, Mset As Double)
Declare Function SFUZdefuzAngleSet Lib "sfuzzy.dll" (Fset As Double) As Double


' Inverse Kinematics Defs
Declare Function INVKIN_FindAngle1 Lib "OldInvKin.dll" (p As Double, ByVal d2 As Double, ByVal arm As Long) As Double
Declare Function INVKIN_FindAngle2 Lib "OldInvKin.dll" (p As Double, d As Double, a As Double, ByVal arm As Long, ByVal
elbow As Long) As Double
Declare Function INVKIN_FindAngle3 Lib "OldInvKin.dll" (p As Double, d As Double, a As Double, ByVal arm As Long, ByVal
elbow As Long) As Double
Declare Function INVKIN_FindAngle4 Lib "OldInvKin.dll" (a As Double, Angles As Double, ByVal M As Long) As Double
Declare Function INVKIN_FindAngle5 Lib "OldInvKin.dll" (a As Double, Angles As Double) As Double
Declare Function INVKIN_FindAngle6 Lib "OldInvKin.dll" (N As Double, o As Double, Angles As Double) As Double
```

5.6 Funcs.bas

```
Option Explicit

Private Const MaxCount = 10

'types
Public Type ArmPosTp
  Angles(1 To 6) As Double
  Pos(0 To 3) As Double
End Type

'path
Public ArmPath() As ArmPosTp
Public NumPathPoints As Integer
Public CurrentPathPoint As Integer

'geometric arm conditions
Public Const LEFT = -1
Public Const RIGHT = 1
Public Const ABOVE = 1
Public Const BELOW = -1
Public Const UP = -1
Public Const DOWN = 1

'fuzzy sets:
Public Const Hangledist = 2
Public Const Range = 360
Public Const NumValues = 360
Public Const tol = 0.001

Public A1Zero(0 To NumValues) As Double
Public A1SmallPos(0 To NumValues) As Double
Public A1LargePos(0 To NumValues) As Double
Public A1SmallNeg(0 To NumValues) As Double
Public A1LargeNeg(0 To NumValues) As Double
Public LenZero(0 To NumValues) As Double
Public LenSmallPos(0 To NumValues) As Double
Public LenLargePos(0 To NumValues) As Double
Public LenSmallNeg(0 To NumValues) As Double
Public LenLargeNeg(0 To NumValues) As Double

Public Sub MatVB(M() As Double, r As Integer, c As Integer, V As Double)
 M((r - 1) * 4 + (c - 1)) = V
End Sub

Public Sub MakeT(Xang As Double, Yang As Double, Zang As Double, M() As Double)
 Dim TrX(0 To 15) As Double
 Dim TrY(0 To 15) As Double
 Dim TrZ(0 To 15) As Double
 Dim temp(0 To 15) As Double
 Dim Cx As Double, Sx As Double
 Dim Cy As Double, Sy As Double
 Dim Cz As Double, Sz As Double

 Cx = Cos(Deg2Rad(Xang))
 Sx = Sin(Deg2Rad(Xang))
 Cy = Cos(Deg2Rad(Yang))
 Sy = Sin(Deg2Rad(Yang))
 Cz = Cos(Deg2Rad(Zang))
 Sz = Sin(Deg2Rad(Zang))

 Mat TrX(0), 1, 1, 1
 Mat TrX(0), 2, 2, Cx
 Mat TrX(0), 2, 3, -Sx
 Mat TrX(0), 3, 2, Sx
 Mat TrX(0), 3, 3, Cx
 Mat TrX(0), 4, 4, 1

 Mat TrY(0), 1, 1, Cy
 Mat TrY(0), 1, 3, Sy
 Mat TrY(0), 2, 2, 1
 Mat TrY(0), 3, 1, -Sy
 Mat TrY(0), 3, 3, Cy

 Mat TrZ(0), 1, 1, Cz
 Mat TrZ(0), 1, 2, -Sz
 Mat TrZ(0), 2, 1, Sz
 Mat TrZ(0), 2, 2, Cz
 Mat TrZ(0), 3, 3, 1
 Mat TrZ(0), 4, 4, 1

 MatrixMult4b4 TrX(0), TrY(0), temp(0)
 MatrixMult4b4 temp(0), TrZ(0), M(0)
End Sub

Public Sub MakeFuzSets()
 Dim DescSet(0 To 8) As Double

'angle sets
 'zero
 DescSet(0) = 3
 DescSet(1) = -10
 DescSet(2) = 0
 DescSet(3) = 0
 DescSet(4) = 1
 DescSet(5) = 10
 DescSet(6) = 0
 SFUZMakeAngleSet DescSet(0), A1Zero(0)

 'small pos
 DescSet(0) = 3
```

```
    DescSet(1) = 0
    DescSet(2) = 0
    DescSet(3) = 10
    DescSet(4) = 1
    DescSet(5) = 100
    DescSet(6) = 0
    SFUZMakeAngleSet DescSet(0), A1SmallPos(0)

    'large pos
    DescSet(0) = 3
    DescSet(1) = 10
    DescSet(2) = 0
    DescSet(3) = 100
    DescSet(4) = 1
    DescSet(5) = Range
    DescSet(6) = 1
    SFUZMakeAngleSet DescSet(0), A1LargePos(0)

    'small neg
    DescSet(0) = 3
    DescSet(1) = -100
    DescSet(2) = 0
    DescSet(3) = -10
    DescSet(4) = 1
    DescSet(5) = 0
    DescSet(6) = 0
    SFUZMakeAngleSet DescSet(0), A1SmallNeg(0)

    'large neg
    DescSet(0) = 3
    DescSet(1) = -Range
    DescSet(2) = 1
    DescSet(3) = -100
    DescSet(4) = 1
    DescSet(5) = -10
    DescSet(6) = 0
    SFUZMakeAngleSet DescSet(0), A1LargeNeg(0)

   'length sets
    'len zero
    DescSet(0) = 3
    DescSet(1) = -30
    DescSet(2) = 0
    DescSet(3) = 0
    DescSet(4) = 1
    DescSet(5) = -30
    DescSet(6) = 0
    SFUZMakeAngleSet DescSet(0), LenZero(0)

    'len small pos
    DescSet(0) = 3
    DescSet(1) = 0
    DescSet(2) = 0
    DescSet(3) = 30
    DescSet(4) = 1
    DescSet(5) = 100
    DescSet(6) = 0
    SFUZMakeAngleSet DescSet(0), LenSmallPos(0)

    'len large pos
    DescSet(0) = 2
    DescSet(1) = 30
    DescSet(2) = 0
    DescSet(3) = 100
    DescSet(4) = 1
```

```
  SFUZMakeAngleSet DescSet(0), LenLargePos(0)

  'len small neg
  DescSet(0) = 3
  DescSet(1) = -100
  DescSet(2) = 0
  DescSet(3) = -30
  DescSet(4) = 1
  DescSet(5) = 0
  DescSet(6) = 0
  SFUZMakeAngleSet DescSet(0), LenSmallNeg(0)

  'len large neg
  DescSet(0) = 2
  DescSet(1) = -100
  DescSet(2) = 1
  DescSet(3) = -30
  DescSet(4) = 0
  SFUZMakeAngleSet DescSet(0), LenLargeNeg(0)
End Sub

Public Function FindAngleInc(error As Double) As Double
  Dim c(1 To 5) As Double
  Dim temp As Double
  Dim w As Double

'rules
'1- If error is zero, increase = 0
'2- If error is small pos, increase = +10
'3- If error is large pos, increase = +100
'4- If error is small neg, increase = -10
'5- If error is large neg, increase = -100
  w = 0

'rule 1
  temp = SFUZevalAngleSet(error, A1Zero(0))
  c(1) = 0
  w = w + temp

'rule 2
  temp = SFUZevalAngleSet(error, A1SmallPos(0))
  c(2) = temp * 10
  w = w + temp

'rule 3
  temp = SFUZevalAngleSet(error, A1LargePos(0))
  c(3) = temp * 100
  w = w + temp

'rule 4
  temp = SFUZevalAngleSet(error, A1SmallNeg(0))
  c(4) = temp * -10
  w = w + temp

'rule 5
  temp = SFUZevalAngleSet(error, A1LargeNeg(0))
  c(5) = temp * -100
  w = w + temp

  temp = (c(1) + c(2) + c(3) + c(4) + c(5)) / w
  FindAngleInc = temp
End Function

Public Sub FindLenInc(error As Double, a2inc As Double, a3inc As Double)
  Dim C2(1 To 5) As Double, c3(1 To 5) As Double
  Dim temp As Double
  Dim w As Double

'rules
'1- If len error is zero, inc2 = 0, inc3 = 0
'2- If len error is small pos, inc2 = +5, inc3 = +50
'3- If len error is large pos, inc2 = +20, inc3 = +100
'4- If len error is small neg, inc2 = -5, inc3 = -50
'5- If len error is large neg, inc2 = -20, inc3 = -100
  w = 0

'rule 1
  temp = SFUZevalAngleSet(error, LenZero(0))
  c3(1) = 0
  w = w + temp

'rule 2
  temp = SFUZevalAngleSet(error, LenSmallPos(0))
  c3(2) = temp * 50
  w = w + temp

'rule 3
  temp = SFUZevalAngleSet(error, LenLargePos(0))
  c3(3) = temp * 100
  w = w + temp

'rule 4
  temp = SFUZevalAngleSet(error, LenSmallNeg(0))
  c3(4) = temp * -50
  w = w + temp

'rule 5
  temp = SFUZevalAngleSet(error, LenLargePos(0))
  c3(5) = temp * -100
  w = w + temp

  a3inc = (c3(1) + c3(2) + c3(3) + c3(4) + c3(5)) / w
End Sub

Public Sub SavePath(fname As String)
  Dim i As Integer
  Dim fn As Integer

  fn = FreeFile
  Open fname For Binary As #fn
    Put #fn, , NumPathPoints
    For i = 1 To NumPathPoints
      Put #fn, , ArmPath(i)
    Next i
  Close #fn
End Sub

Public Sub OpenPath(fname As String)
  Dim i As Integer
  Dim fn As Integer

  fn = FreeFile
  Open fname For Binary As #fn
    Get #fn, , NumPathPoints
    ReDim ArmPath(1 To NumPathPoints)
    For i = 1 To NumPathPoints
      Get #fn, , ArmPath(i)
    Next i
  Close #fn
End Sub
```

6 Code listing for the Web Client - WEBClient.vbp

6.1 frmWEBClient

```
Option Explicit
Dim messageIn As String
Dim messageOut As String
Private Sub cmdAGVAVIS_Click()
frmTCPClient.txtSend.Text = "ClientWEB-AGVAVIS"
tex1.Text = "ClientWEB-AGVAVIS"
End Sub

Private Sub cmdAGVROBOT_Click()
frmTCPClient.txtSend.Text = "ClientWEB-AGVROBOT"
tex1.Text = "ClientWEB-AGVROBOT"
End Sub
Private Sub cmdAVISAGV_Click()
frmTCPClient.txtSend.Text = "ClientWEB-AVISAGV"
tex1.Text = "ClientWEB-AVISAGV"
End Sub
Private Sub cmdCONEXIT_Click()
frmTCPClient.txtSend.Text = "ClientWEB-EXIT"
tex1.Text = "ClientWEB-EXIT"
End
End Sub
Private Sub cmdCONSTART_Click()
frmTCPClient.txtSend.Text = "ClientWEB-START"
tex1.Text = "ClientWEB-START"
End Sub

Private Sub cmdRj1p_Click()
frmTCPClient.txtSend.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
tex1.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
End Sub

Private Sub cmdRj2p_Click()
frmTCPClient.txtSend.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
tex1.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
End Sub

Private Sub cmdRj3p_Click()
frmTCPClient.txtSend.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
tex1.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
End Sub

Private Sub cmdRj4p_Click()
frmTCPClient.txtSend.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
tex1.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
End Sub

Private Sub cmdRj5p_Click()
frmTCPClient.txtSend.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
tex1.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
```

```
End Sub

Private Sub cmdRj6p_Click()
frmTCPClient.txtSend.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
tex1.Text = "ClientROB-" & j1.Text & j2.Text & j3.Text & j4.Text _
& j5.Text & j6.Text
End Sub

Private Sub cmdROBOTAGV_Click()
frmTCPClient.txtSend.Text = "ClientWEB-ROBOTAGV"
tex1.Text = "ClientWEB-ROBOTAGV"
End Sub

Private Sub cmdROBOTAVIS_Click()
frmTCPClient.txtSend.Text = "ClientROB-X" & xpos & "Y" & ypos _
& "Z" & zpos
tex1.Text = "ClientROB-X" & xpos & "Y" & ypos _
& "Z" & zpos
End Sub

Private Sub cmdRxyz_Click()
frmTCPClient.txtSend.Text = "ClientWEB-AVISROBOT"
tex1.Text = "ClientWEB-AVISROBOT"
End Sub

Private Sub Option1_Click()
Option2.Enabled = False
Option3.Enabled = False
Option4.Enabled = False
frmTCPClient.txtSend.Text = "ClientWEB-CAM1"
tex1.Text = "ClientWEB-CAM1"
End Sub

Private Sub Option2_Click()
Option1.Enabled = False
Option3.Enabled = False
Option4.Enabled = False
frmTCPClient.txtSend.Text = "ClientWEB-CAM2"
tex1.Text = "ClientWEB-CAM2"
End Sub

Private Sub Option3_Click()
Option2.Enabled = False
Option1.Enabled = False
Option4.Enabled = False
frmTCPClient.txtSend.Text = "ClientWEB-CAM3"
tex1.Text = "ClientWEB-CAM3"
End Sub

Private Sub Option4_Click()
Option2.Enabled = False
Option3.Enabled = False
Option1.Enabled = False
frmTCPClient.txtSend.Text = "ClientWEB-CAM4"
tex1.Text = "ClientWEB-CAM4"
End Sub

Private Sub Text1_Change()
Dim WEBmessage As String
Let WEBmessage = Text1.Text

End Sub
```

6.2 frmWEBTCPClient

```
Option Explicit

Private Sub Form_Load()
  cmdSend.Enabled = False
  ' set up local port and wait for connection
  tcpClient.RemoteHost = "146.230.192.36"
  tcpClient.RemotePort = 5000  ' server port
  Call tcpClient.Connect  ' connect to RemoteHost address
End Sub

Private Sub Form_Terminate()
  Call tcpClient.Close
End Sub

Private Sub Form_Resize()
  On Error Resume Next
  Call cmdSend.Move(ScaleWidth - cmdSend.Width, 0)
  Call txtSend.Move(0, 0, ScaleWidth - cmdSend.Width)
  Call txtOutput.Move(0, txtSend.Height, ScaleWidth, _
    ScaleHeight - txtSend.Height)
End Sub

Private Sub tcpClient_Connect()
  ' when connection occurs, display a message
  cmdSend.Enabled = True
  txtOutput.Text = "Connected to Web Server at IP Address: " & _
    tcpClient.RemoteHostIP & vbCrLf & "Port #: " & _
    tcpClient.RemotePort & vbCrLf & vbCrLf
End Sub

Private Sub tcpClient_DataArrival(ByVal bytesTotal As Long)
  Dim message As String
  Call tcpClient.GetData(message)  ' get data from server
  txtOutput.Text = txtOutput.Text & message & vbCrLf & vbCrLf
  txtOutput.SelStart = Len(txtOutput.Text)
  frmWEBClient.Text1.Text = message
End Sub

Private Sub tcpClient_Close()
  cmdSend.Enabled = False
  Call tcpClient.Close  ' server closed, client should too
  txtOutput.Text = _
    txtOutput.Text & "Server closed connection." & vbCrLf
  txtOutput.SelStart = Len(txtOutput.Text)
End Sub
Private Sub tcpClient_Error(ByVal Number As Integer, _
    Description As String, ByVal Scode As Long, _
    ByVal Source As String, ByVal HelpFile As String, _
    ByVal HelpContext As Long, CancelDisplay As Boolean)
  Dim result As Integer
  result = MsgBox(Source & ": " & Description, _
    vbOKOnly, "TCP/IP Error")
  End
End Sub
Private Sub cmdSend_Click()
  ' send data to server
  Call tcpClient.SendData(txtSend.Text)
  txtOutput.Text = txtOutput.Text & _
    txtSend.Text & vbCrLf & vbCrLf
  txtOutput.SelStart = Len(txtOutput.Text)
  txtSend.Text = ""
End Sub
```

7 Code listing for frame grabber card - flahptb.vbp

7.1 frmConfig

```
Private Sub chkFieldRep_Click()
  Dim nRet As Integer
  Dim Check As Integer
  Rem check the value of field replicate
  Check = frmConfig.chkScaleVideo.value
  nReplicate = Check
End Sub

Private Sub chkRemoteGrab_Click()
  Dim nRet As Integer
  Dim Check As Integer
  Rem check the value of field replicate
  Check = frmConfig.chkRemoteGrab.value
  nRemoteGrab = Check
End Sub

Private Sub chkScaleVideo_Click()
  Dim nRet As Integer
  Dim Check As Integer
  Rem Turn on/off the scale video
  Check = frmConfig.chkScaleVideo.value
  If Check = 0 Then
    nRet = FPV_SetVideoWindow(-1, -1, -1, -1, False)
  Else
    nRet = FPV_SetVideoWindow(-1, -1, -1, -1, True)
  End If
End Sub

Private Sub cmdConfigCancel_Click()
  Rem hide the config form/ activate FlashPt form
  frmConfig.Enabled = False
  frmFlashPT.Enabled = True
  frmConfig.Hide
End Sub

Private Sub cmdConfigOK_Click()
  Dim nRet As Integer
  Rem hide the config form/ activateFlashpt form /turn on video
  frmFlashPT.Enabled = True
  frmConfig.Enabled = False
  frmConfig.Hide
  nRet = FPV_VideoLive(True, ALIGN_ANY)
End Sub

Private Sub Form_Unload(Cancel As Integer)
  frmFlashPT.Enabled = True
  Unload frmConfig
End Sub

Private Sub hsbXCenter_Change()
  Dim nRet As Integer
  Rem change the X Center of the video
  frmConfig.txtXCenter = Str(frmConfig.hsbXCenter.value)
  nRet = FPV_SetInputOffset(frmConfig.hsbXCenter.value, frmConfig.hsbYCenter.value)
End Sub

Private Sub hsbYCenter_Change()
Dim nRet As Integer
```

```
    Rem Change the Y Center of the Video
    frmConfig.txtYCenter = Str(frmConfig.hsbYCenter.value)
    nRet = FPV_SetInputOffset(frmConfig.hsbXCenter.value, frmConfig.hsbYCenter.value)
End Sub

Private Sub opt200_Click()
Dim nRet As Integer
frmConfig.opt200.value = True
frmFlashPT.nAddress = 512   'in hex 200
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub

Private Sub opt220_Click()
Dim nRet As Integer
frmConfig.opt220.value = True
frmFlashPT.nAddress = 544
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub

Private Sub opt240_Click()
Dim nRet As Integer
frmConfig.opt240.value = True
frmFlashPT.nAddress = 576
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub

Private Sub opt260_Click()
Dim nRet As Integer
frmConfig.opt260.value = True
nAddress = 608
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub

Private Sub opt280_Click()
Dim nRet As Integer
frmConfig.opt280.value = True
frmFlashPT.nAddress = 640
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub

Private Sub opt2a0_Click()
Dim nRet As Integer
frmConfig.opt2a0.value = True
frmFlashPT.nAddress = 672
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub

Private Sub opt2c0_Click()
Dim nRet As Integer
frmConfig.opt2c0.value = True
frmFlashPT.nAddress = 704
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub

Private Sub opt2e0_Click()
Dim nRet As Integer
frmConfig.opt2e0.value = True
frmFlashPT.nAddress = 736
nRet = FPV_CheckSwitch(nAddress, 0)
End Sub
```

7.2 frmFlashPT

```
Public nAddress As Integer
```

```
Option Explicit
Dim nFlashDelayField As Integer
Dim nAlign As Integer
Dim nFlashType As Integer
Dim nFlashDelayLine As Integer
Dim GrabBrightness As Integer
Dim GrabContrast As Integer
Dim GrabAutoIris As Integer
Dim nBrightness As Integer
Dim nContrast As Integer
Dim nHue As Integer
Dim nSaturation As Integer
Dim BaseAddress As Integer
Dim nSharpness As Integer
Dim nReplicate As Long
Dim nRemoteGrab As Integer
Dim FlashPTB As String

Private Sub Form_Activate()
Dim nRet As Integer
Dim nScreenDepth As Integer
Dim hDC As Integer
Dim BitsPixel As Integer
Dim Planes As Integer

If (nScreenDepth = 8) Then
   nRet = FPV_SetPalette(SETPAL_SELHWND, frmFlashPT.hWnd)
   End If
End Sub

Private Sub Form_Load()
Dim nRet As Integer
Dim szInVid As String * 8
Dim nVidType As Integer, nVidStandard As Integer
Dim nVidSource As Integer, nSyncGreen As Integer
Dim nBrightnessDef As Integer, nContrastDef As Integer
Dim nHueDef As Integer, nSaturationDef As Integer
Dim szardef As String * 255
Dim szarval As String * 255
Dim I As Integer
Dim J As Integer
Dim ScreenDepth As Integer
Dim hDC As Integer
Dim BitsPixel As Integer, Planes As Integer
Dim nScreenDepth As Integer

Rem name of .ini file
FlashPTB = "FlashPTB.ini"

Rem Initialize
nRet = FPV_LoadConfig(ByVal &O0)
nRet = FPV_Init()

nRet = FPV_GetMiscParm(MISCPARM_INPUTVID, ByVal szInVid)
  If (nRet <> RET_ERROR) Then
     nVidType = Asc(Mid$(szInVid, 1, 1))
     nVidStandard = Asc(Mid$(szInVid, 2, 1))
     nVidSource = Asc(Mid$(szInVid, 3, 1))
     nSyncGreen = Asc(Mid$(szInVid, 4, 1))
  Else
     nVidType = TYPE_COMPOSITE
     nVidStandard = STANDARD_NTSC
     nVidSource = 1
     nSyncGreen = False
  End If
```

```
If nVidType = TYPE_COMPOSITE Then
  frmVideo.optComposite.value = True
  frmVideo.optSVideo.value = False
  frmVideo.optRS170.value = False
 End If
 If nVidType = TYPE_SVIDEO Then
  frmVideo.optSVideo.value = True
  frmVideo.optComposite.value = False
  frmVideo.optRS170.value = False
 End If
  If nVidType = TYPE_RS170 Then
  frmVideo.optSVideo.value = False
  frmVideo.optComposite.value = False
  frmVideo.optRS170.value = True
 End If

  If nVidStandard = STANDARD_NTSC Then
  frmVideo.optNTSC.value = True
  Else
  frmVideo.optPAL.value = True
  End If

Rem Check for 8-bit(256 Color)

'hDC = GetDC(GetDesktopWindow())

nScreenDepth = GetDeviceCaps(hDC, BitsPixel) * GetDeviceCaps(hDC, Planes)

If (nScreenDepth = 8) Then
     nRet = FPV_SetPalette(SETPAL_CREATEGRAY8, frmFlashPT.hWnd)
     Rem Twice to insure order
     nRet = FPV_SetPalette(SETPAL_SELHWND, frmFlashPT.hWnd)
     nRet = FPV_SetPalette(SETPAL_SELHWND, frmFlashPT.hWnd)
   End If

ReleaseDC GetDesktopWindow(), hDC

Rem get initial video values
     Rem setup display window
     nRet = FPV_SetVideoConfig(nVidType, nVidStandard, nVidSource, False)
   nRet = FPV_SetVideoWindow(0, 0, 320, 240, True)
   nRet = FPV_AutoWindow(frmFlashPT.hWnd, 0, 0, 0, 0, AUTOWIN_CLIP + AUTOWIN_POS + AUTOWIN_SIZE +
AUTOWIN_ADJUSTWINDOW)

   Rem set sync on green to false
SyncGreen = False

   Rem Get Default values from EEPROM
   For I = 0 To 1
      For J = 0 To 3
         DefBrightness(I, J) = FPV_GetSTVideoAdjustments(I, J, ADJUST_BRIGHTNESS)
         DefContrast(I, J) = FPV_GetSTVideoAdjustments(I, J, ADJUST_CONTRAST)
         DefSaturation(I, J) = FPV_GetSTVideoAdjustments(I, J, ADJUST_SATURATION)
         DefHue(I, J) = FPV_GetSTVideoAdjustments(I, J, ADJUST_HUE)
      Next J
   Next I

   Rem Get Initial values for Video
   nBrightnessDef = FPV_GetSTVideoAdjustments(nVidStandard, nVidType, ADJUST_BRIGHTNESS)
   frmVideo.vsbBrightness.value = nBrightnessDef
   frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
   nContrastDef = FPV_GetSTVideoAdjustments(nVidStandard, nVidType, ADJUST_CONTRAST)
   frmVideo.vsbContrast.value = nContrastDef
   frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
   nHueDef = FPV_GetSTVideoAdjustments(nVidStandard, nVidType, ADJUST_HUE)
```

```
frmVideo.vsbHue.value = nHueDef
frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)
nSaturationDef = FPV_GetSTVideoAdjustments(nVidStandard, nVidType, ADJUST_SATURATION)
frmVideo.vsbSaturation.value = nSaturationDef
frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)
Rem set Default values
frmGrab.optnone = True
frmConfig.chkScaleVideo.value = 1
frmConfig.chkFieldRep.value = 1
frmConfig.opt200 = True
frmGrab.optAny = True
Rem load settings from .ini file
Rem if no .ini then default values will be used
Rem load video standard
nRet = FPV_GetPrivateProfileString("Video", "NTSC", szardef, szarval, 80, FlashPTB)
If Val(szarval) = 1 Then
  nVidStandard = STANDARD_PAL
  frmVideo.optPAL.value = True
ElseIf Val(szarval) = 0 Then
  nVidStandard = STANDARD_NTSC
  frmVideo.optNTSC.value = True
End If
Rem -1 = True
Rem load video type
nRet = FPV_GetPrivateProfileString("Video", "Composite", szardef, szarval, 80, FlashPTB)
If Val(szarval) = -1 Then
  frmVideo.optComposite.value = True
  nVidType = TYPE_COMPOSITE
  nVidSource = 1
End If
nRet = FPV_GetPrivateProfileString("Video", "RGB", szardef, szarval, 80, FlashPTB)
nRet = FPV_GetPrivateProfileString("Video", "SVideo", szardef, szarval, 80, FlashPTB)
If Val(szarval) = -1 Then
  frmVideo.optSVideo.value = True
  nVidType = TYPE_SVIDEO
  nVidSource = 1
End If
nRet = FPV_GetPrivateProfileString("Video", "RS170", szardef, szarval, 80, FlashPTB)
If Val(szarval) = -1 Then
  frmVideo.optRS170.value = True
  nVidType = TYPE_RS170
  nVidSource = 1
End If

Rem Load the Brightness value from .ini File
szardef = Str(nBrightnessDef)
nRet = FPV_GetPrivateProfileString("Video", "Brightness", szardef, szarval, 80, FlashPTB)
frmVideo.vsbBrightness.value = Val(szarval)
frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
nBrightness = frmVideo.vsbBrightness.value
nRet = FPV_SetVideoAdjustments(ADJUST_BRIGHTNESS, frmVideo.vsbBrightness.value)

Rem Load the Contrast value from .ini
szardef = Str(nContrastDef)
nRet = FPV_GetPrivateProfileString("Video", "Contrast", szardef, szarval, 80, FlashPTB)
frmVideo.vsbContrast.value = Val(szarval)
frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
nContrast = frmVideo.vsbContrast.value
nRet = FPV_SetVideoAdjustments(ADJUST_CONTRAST, frmVideo.vsbContrast.value)

Rem Load the Hue value from .ini
szardef = Str(nHueDef)
nRet = FPV_GetPrivateProfileString("Video", "Hue", szardef, szarval, 80, FlashPTB)
frmVideo.vsbHue.value = Val(szarval)
frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)
```

```
    nHue = frmVideo.vsbHue.value
    nRet = FPV_SetVideoAdjustments(ADJUST_HUE, nHue)

    Rem load the saturation value from .ini
    szardef = Str(nSaturationDef)
    nRet = FPV_GetPrivateProfileString("Video", "Saturation", szardef, szarval, 80, FlashPTB)
    frmVideo.vsbSaturation.value = Val(szarval)
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)
    nSaturation = frmVideo.vsbSaturation.value
    nRet = FPV_SetVideoAdjustments(ADJUST_SATURATION, nSaturation)

    Rem load the sharpness value from .ini
    szardef = Str(0)
    nRet = FPV_GetPrivateProfileString("Video", "Sharpness", szardef, szarval, 80, FlashPTB)
    frmVideo.vsbSharpness.value = Val(szarval)
    frmVideo.txtSharpness.Text = Str(frmVideo.vsbSharpness.value)
    nSharpness = frmVideo.vsbSharpness.value
    nRet = FPV_SetVideoAdjustments(ADJUST_SHARPNESS, nSharpness)

  Rem Setup Grab settings from .ini file/if no .ini file then nothing will happen
    szardef = Str(nBrightnessDef)
    nRet = FPV_GetPrivateProfileString("Grab", "Brightness", szardef, szarval, 80, FlashPTB)
    frmGrab.hsbGrabBrightness.value = Val(szarval)
    frmGrab.txtGrabBrightness.Text = Str(frmGrab.hsbGrabBrightness.value)
    szardef = Str(nContrastDef)
    nRet = FPV_GetPrivateProfileString("Grab", "Contrast", szardef, szarval, 80, FlashPTB)
    frmGrab.hsbGrabContrast.value = Val(szarval)
    frmGrab.txtGrabContrast.Text = Str(frmGrab.hsbGrabContrast.value)
    szardef = Str(0)
    nRet = FPV_GetPrivateProfileString("Grab", "Field", szardef, szarval, 80, FlashPTB)
    frmGrab.hsbGrabField.value = Val(szarval)
    frmGrab.txtGrabField.Text = Str(frmGrab.hsbGrabField.value)
    nRet = FPV_GetPrivateProfileString("Grab", "Line", szardef, szarval, 80, FlashPTB)
    frmGrab.hsbGrabLine.value = Val(szarval)
    frmGrab.txtGrabLine.Text = Str(frmGrab.hsbGrabLine.value)

    Rem Set the Align for the grab from the .ini file
    nRet = FPV_GetPrivateProfileString("Grab", "Even", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmGrab.optEven = True
      nAlign = ALIGN_EVEN
    End If
    nRet = FPV_GetPrivateProfileString("Grab", "Odd", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmGrab.optOdd = True
      nAlign = ALIGN_ODD
    End If
    nRet = FPV_GetPrivateProfileString("Grab", "Any", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmGrab.optAny = True
      nAlign = ALIGN_ANY
    End If
    Rem Set the type for the grab from the .ini file
    nRet = FPV_GetPrivateProfileString("Grab", "None", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmGrab.optnone = True
      nAlign = FLASHTYPE_NONE
    End If
    nRet = FPV_GetPrivateProfileString("Grab", "Universal", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmGrab.optUniversal = True
      nAlign = FLASHTYPE_UNIVERSAL
    End If
    nRet = FPV_GetPrivateProfileString("Grab", "Dual", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
```

```
        frmGrab.optDual = True
        nAlign = FLASHTYPE_DUALFIELD
      End If
        nRet = FPV_GetPrivateProfileString("Grab", "CCD4000", szardef, szarval, 80, FlashPTB)
      If Val(szarval) = -1 Then
        frmGrab.optCCD = True
        nAlign = FLASHTYPE_CCD4000
      End If
      Rem load base address

      nRet = FPV_GetPrivateProfileString("Address", "Serial", szardef, szarval, 80, FlashPTB)
      nAddress = Val(szarval)

      Rem set Config options from .ini file
      Rem read in Xcenter value
      szardef = Str(0)
      nRet = FPV_GetPrivateProfileString("Config", "XCenter", szardef, szarval, 80, FlashPTB)
      frmConfig.hsbXCenter.value = Val(szarval)
      frmConfig.txtXCenter.Text = Str(frmConfig.hsbXCenter.value)

      Rem read in YCenter value
      szardef = Str(0)
      nRet = FPV_GetPrivateProfileString("Config", "YCenter", szardef, szarval, 80, FlashPTB)
      frmConfig.hsbYCenter.value = Val(szarval)
      frmConfig.txtYCenter.Text = Str(frmConfig.hsbYCenter.value)

      Rem read in scale video value
      szardef = Str(1)
      frmConfig.chkScaleVideo.value = 1
      nRet = FPV_GetPrivateProfileString("Config", "ScaleVideo", szardef, szarval, 80, FlashPTB)
      If Val(szarval) = 0 Then
        frmConfig.chkScaleVideo.value = 0
      End If
      Rem read in field rep value
      szardef = Str(0)
       frmConfig.chkFieldRep.value = 0
      nRet = FPV_GetPrivateProfileString("Config", "FieldRep", szardef, szarval, 80, FlashPTB)
      If Val(szarval) = 1 Then
        frmConfig.chkFieldRep.value = 1
      End If
    Rem get serial address
    nRet = FPV_GetPrivateProfileString("Address", "x200", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmConfig.opt200 = True
      nAddress = 512
    End If
    nRet = FPV_GetPrivateProfileString("Address", "x220", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmConfig.opt220 = True
      nAddress = 544
    End If
    nRet = FPV_GetPrivateProfileString("Address", "x240", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmConfig.opt240 = True
      nAddress = 576
    End If
    nRet = FPV_GetPrivateProfileString("Address", "x260", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmConfig.opt260 = True
      nAddress = 608
    End If
    nRet = FPV_GetPrivateProfileString("Address", "x280", szardef, szarval, 80, FlashPTB)
    If Val(szarval) = -1 Then
      frmConfig.opt280 = True
      nAddress = 640
```

```
End If
nRet = FPV_GetPrivateProfileString("Address", "x2a0", szardef, szarval, 80, FlashPTB)
If Val(szarval) = -1 Then
  frmConfig.opt2a0 = True
  nAddress = 672
End If
nRet = FPV_GetPrivateProfileString("Address", "x2c0", szardef, szarval, 80, FlashPTB)
If Val(szarval) = -1 Then
  frmConfig.opt2c0 = True
  nAddress = 704
End If
nRet = FPV_GetPrivateProfileString("Address", "x2e0", szardef, szarval, 80, FlashPTB)
If Val(szarval) = -1 Then
  frmConfig.opt2e0 = True
  nAddress = 736
End If
nRet = FPV_CheckSwitch(nAddress, 0)

Rem set Freeze/Live menu option accordingly
mnuFreeze.Visible = True
mnuLive.Visible = False
Rem set reset(on Video Setup) to false
frmVideo.cmdVideoReset.Enabled = True

Rem set windows to specific location on screen
frmFlashPT.Left = 1440
frmFlashPT.Top = 1440
frmVideo.Left = 6735
frmVideo.Top = 1440

Rem Set Video Live
nRet = FPV_VideoLive(True, ALIGN_ANY)

Timer1.Enabled = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
Dim nRet As Integer

Rem Cleanup actually makes sure video is off and
Rem AutoWindow is disabled
nRet = FPV_Cleanup()
End
End Sub

Private Sub mnuAbout_Click()
  Rem Show the About Form/disable the FlashPT form
  frmFlashPT.Enabled = False
  frmAbout.Enabled = True
  frmAbout.Show
End Sub

Private Sub mnuConfiguration_Click()
  Rem Show the Configuration form/Disable FlashPT form
  frmConfig.Show
  frmFlashPT.Enabled = False
  frmConfig.Enabled = True
End Sub

Private Sub mnuCopy_Click()

  Dim nRet As Integer
  Dim nX As Integer
  Dim nY As Integer
  Dim nWidth As Integer
```

```
    Dim nHeight As Integer

    Rem Put copy of current image on the Clipboard
    nRet = FPV_GetVideoRect(DESTRECT, nX, nY, nWidth, nHeight)
    nRet = FPV_VideoLive(False, ALIGN_ANY)
    nRet = FPV_WaitMS(32)
    nRet = FPV_ScreenToDIB(nX, nY, nWidth, nHeight, STD_CLIPBOARD, ByVal &O0)
    nRet = FPV_VideoLive(True, ALIGN_ANY)

  End Sub

  Private Sub mnuExit_Click()
    Dim nRet As Integer
    Rem    exit the program
    nRet = FPV_VideoLive(False, ALIGN_ANY)
    nRet = FPV_AutoWindow(0, 0, 0, 0, 0)
    nRet = FPV_Cleanup()
    Unload Me
    End
  End Sub

  Private Sub mnuFreeze_Click()
    Dim nRet As Integer
    Dim nFlashParm As Integer

    Rem If FlashType Universal or Dual Then Adjust Brightness/contrast
    If ((nFlashType = FLASHTYPE_UNIVERSAL) Or (nFlashType = FLASHTYPE_DUALFIELD)) Then
      nRet = FPV_SetVideoAdjustments(ADJUST_BRIGHTNESS, GrabBrightness)
      nRet = FPV_SetVideoAdjustments(ADJUST_CONTRAST, GrabContrast)
    End If

    Rem do grab if remote grab is on
    If ((frmGrab.optDual.value = -1) And (frmConfig.chkRemoteGrab.value = 1)) Then
      nFlashParm = FLASHTYPE_WAITSWITCH
    End If

    Rem If the Field Rep is Checked then OR flashtype with FieldRep
    If frmConfig.chkFieldRep.value = 1 Then
      nFlashParm = nFlashType Or FLASHFLAG_FIELDREP Or FLASHFLAG_REDGESMOOTH
    Else
      Rem Otherwise dont dont change parameters
      nFlashParm = nFlashType Or nFlashParm
    End If

      Rem do the grab
    nRet = FPV_VideoGrab(nAlign, nFlashParm, nFlashDelayField, nFlashDelayLine)

    Rem adjust menus
    mnuFreeze.Visible = False
    mnuLive.Visible = True
  End Sub

  Private Sub mnuGrab_Click()
    Rem Show the Grab form/disable FlashPT form
    frmGrab.Show
    frmFlashPT.Enabled = False
    frmGrab.Enabled = True
  End Sub

  Private Sub mnuLive_Click()
    Rem turn video to live again
   Dim nRet As Integer
    nRet = FPV_VideoLive(True, ALIGN_ANY)
    mnuFreeze.Visible = True
    mnuLive.Visible = False
```

```
End Sub

Private Sub mnuLoadImage_Click()
   Dim nRet
   Rem show the load image form/ plus turn off video live
   nRet = FPV_VideoLive(False, ALIGN_ANY)
   frmLoadImage.Enabled = True
   frmFlashPT.Enabled = False
   frmLoadImage.Show
   End Sub

Private Sub mnuPrint_Click()
   Rem show the print form
   frmPrint.Show
  End Sub

Private Sub mnuSaveImage_Click()
   Dim nRet As Integer
   Rem Show the Save Image Form
   nRet = FPV_VideoLive(False, ALIGN_ANY)
   frmSaveImage.Enabled = True
   frmFlashPT.Enabled = False
   frmSaveImage.Show

End Sub

Private Sub mnuSaveSettings_Click()
   Dim nRet As Integer
   Dim nVal As Integer

   Rem save settings to .ini file
   Rem save video standard
   nVal = frmVideo.optNTSC.value
   nRet = FPV_WritePrivateProfileString("Video", "NTSC", nVal, FlashPTB)
   nVal = frmVideo.optPAL.value
   nRet = FPV_WritePrivateProfileString("Video", "PAL", nVal, FlashPTB)
   Rem save video type
   nVal = frmVideo.optComposite.value
   nRet = FPV_WritePrivateProfileString("Video", "Composite", nVal, FlashPTB)
   nVal = frmVideo.optSVideo.value
   nRet = FPV_WritePrivateProfileString("Video", "SVideo", nVal, FlashPTB)
   nVal = frmVideo.optRS170.value
   nRet = FPV_WritePrivateProfileString("Video", "RS170", nVal, FlashPTB)
   Rem save serial address
   nVal = frmConfig.opt200.value
   nRet = FPV_WritePrivateProfileString("Address", "x200", nVal, FlashPTB)
   nVal = frmConfig.opt220.value
   nRet = FPV_WritePrivateProfileString("Address", "x220", nVal, FlashPTB)
   nVal = frmConfig.opt240.value
   nRet = FPV_WritePrivateProfileString("Address", "x240", nVal, FlashPTB)
   nVal = frmConfig.opt260.value
   nRet = FPV_WritePrivateProfileString("Address", "x260", nVal, FlashPTB)
   nVal = frmConfig.opt280.value
   nRet = FPV_WritePrivateProfileString("Address", "x280", nVal, FlashPTB)
   nVal = frmConfig.opt2a0.value
   nRet = FPV_WritePrivateProfileString("Address", "x2a0", nVal, FlashPTB)
   nVal = frmConfig.opt2c0.value
   nRet = FPV_WritePrivateProfileString("Address", "x2c0", nVal, FlashPTB)
   nVal = frmConfig.opt2e0.value
   nRet = FPV_WritePrivateProfileString("Address", "x2e0", nVal, FlashPTB)

   Rem save video adjustments
   nVal = frmVideo.vsbBrightness.value
   nRet = FPV_WritePrivateProfileString("Video", "Brightness", nVal, FlashPTB)
   nVal = frmVideo.vsbContrast.value
```

```
    nRet = FPV_WritePrivateProfileString("Video", "Contrast", nVal, FlashPTB)
    nVal = frmVideo.vsbHue.value
    nRet = FPV_WritePrivateProfileString("Video", "Hue", nVal, FlashPTB)
    nVal = frmVideo.vsbSaturation.value
    nRet = FPV_WritePrivateProfileString("Video", "Saturation", nVal, FlashPTB)
    nVal = frmVideo.vsbSharpness.value
    nRet = FPV_WritePrivateProfileString("Video", "Sharpness", nVal, FlashPTB)
    nVal = frmVideo.vsbIrisLevel.value
    nRet = FPV_WritePrivateProfileString("Video", "Iris Level", nVal, FlashPTB)
    Rem save Grab settings
    nVal = frmGrab.hsbGrabBrightness.value
    nRet = FPV_WritePrivateProfileString("Grab", "Brightness", nVal, FlashPTB)
    nVal = frmGrab.hsbGrabContrast.value
    nRet = FPV_WritePrivateProfileString("Grab", "Contrast", nVal, FlashPTB)
    nVal = frmGrab.hsbGrabField.value
    nRet = FPV_WritePrivateProfileString("Grab", "Field", nVal, FlashPTB)
    nVal = frmGrab.hsbGrabLine.value
    nRet = FPV_WritePrivateProfileString("Grab", "Brightness", nVal, FlashPTB)
    nVal = frmGrab.optEven.value
    nRet = FPV_WritePrivateProfileString("Grab", "Even", nVal, FlashPTB)
    nVal = frmGrab.optOdd.value
    nRet = FPV_WritePrivateProfileString("Grab", "Odd", nVal, FlashPTB)
    nVal = frmGrab.optAny.value
    nRet = FPV_WritePrivateProfileString("Grab", "Any", nVal, FlashPTB)
    nVal = frmGrab.optnone.value
    nRet = FPV_WritePrivateProfileString("Grab", "None", nVal, FlashPTB)
    nVal = frmGrab.optUniversal.value
    nRet = FPV_WritePrivateProfileString("Grab", "Universal", nVal, FlashPTB)
    nVal = frmGrab.optDual.value
    nRet = FPV_WritePrivateProfileString("Grab", "Dual", nVal, FlashPTB)
    nVal = frmGrab.optCCD.value
    nRet = FPV_WritePrivateProfileString("Grab", "CCD4000", nVal, FlashPTB)
    Rem Save configure settings
    nVal = frmConfig.chkScaleVideo.value
    nRet = FPV_WritePrivateProfileString("Config", "ScaleVideo", nVal, FlashPTB)
    nVal = frmConfig.hsbXCenter.value
    nRet = FPV_WritePrivateProfileString("Config", "XCenter", nVal, FlashPTB)
    nVal = frmConfig.hsbYCenter.value
    nRet = FPV_WritePrivateProfileString("Config", "YCenter", nVal, FlashPTB)
    nVal = frmConfig.chkFieldRep.value
    nRet = FPV_WritePrivateProfileString("Config", "FieldRep", nVal, FlashPTB)
    Rem save base address
    nVal = nAddress
    nRet = FPV_WritePrivateProfileString("Address", "Serial", nVal, FlashPTB)

End Sub

Private Sub mnuVideoSetup_Click()
  Rem show the video setup form/disable FlashPt form
  frmVideo.Show
  frmVideo.Enabled = True
  frmFlashPT.Enabled = False
End Sub

Private Sub Timer1_Timer()
Dim nRet As Integer
nRet = FPV_VideoLive(False, ALIGN_ANY)
frmFlashPT.Enabled = False

  Dim nX As Integer
  Dim nY As Integer
  Dim nWidth As Integer
  Dim nHeight As Integer
  Dim TheFile As String
  Rem show save image form/turn off flashpt form
```

```
    frmSaveImage.Hide
    frmFlashPT.Show
    Rem the file selected by user
    Open "c:\Inetpub\wwwroot\webcam32.jpg" For Output As #1
    nRet = FPV_GetVideoRect(DESTRECT, nX, nY, nWidth, nHeight)
    nRet = FPV_SaveFile("c:\Inetpub\wwwroot\webcam32.jpg", nX, nY, nWidth, nHeight, 24, 0, ByVal 0&, -1)
    Close #1
    Rem disable save image / activate  flashpt form/ set menu accordingly

    frmFlashPT.Enabled = True
    frmFlashPT.mnuFreeze.Visible = False

    nRet = FPV_VideoLive(True, ALIGN_ANY)
    mnuFreeze.Visible = True
    mnuLive.Visible = False

End Sub
```

7.3 frmGrab

```
Private Sub cmdGrabOK_Click()
    Rem grab Setup menu
    Rem hide Grab Setup menu/activate FlashPt form again

    frmFlashPT.Enabled = True
    frmGrab.Enabled = False
    frmGrab.Hide
End Sub

Private Sub cmdGrabReset_Click()
    Dim nGrabBrightnessDef As Integer
    Dim nGrabContrastDef As Integer
    Dim nVidType As Integer
    Dim nVidStandard As Integer
    Rem Reset the Grab value from their initial values
    nVidType = FPV_GetVideoType()
    nVidStandard = FPV_GetVideoStandard()
    nGrabBrightnessDef = FPV_GetSTVideoAdjustments(nVidStandard, nVidType, ADJUST_BRIGHTNESS)
    frmGrab.hsbGrabBrightness.value = nGrabBrightnessDef
    frmGrab.txtGrabBrightness.Text = Str(frmGrab.hsbGrabBrightness.value)
    nGrabContrastDef = FPV_GetSTVideoAdjustments(nVidStandard, nVidType, ADJUST_CONTRAST)
    frmGrab.hsbGrabContrast.value = nGrabContrastDef
    frmGrab.txtGrabContrast.Text = Str(frmGrab.hsbGrabContrast.value)

End Sub

Private Sub Field_Click()
    nReplicate.value
End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmFlashPT.Enabled = True
    Unload frmGrab
End Sub

Private Sub hsbGrabAutoIris_Change()
    Rem change the value of the grab auto iris value appropiately
    frmGrab.txtGrabAutoIris.Text = Str(frmGrab.hsbGrabAutoIris.value)
    GrabAutoIris = frmGrab.hsbGrabAutoIris.value
End Sub

Private Sub hsbGrabBrightness_Change()
    Rem change the brightness of the grab accordingly
```

```
   frmGrab.txtGrabBrightness.Text = Str(frmGrab.hsbGrabBrightness.value)
   GrabBrightness = frmGrab.hsbGrabBrightness.value
End Sub

Private Sub hsbGrabContrast_Change()
   Rem change the contrast of the grab accordingly
   frmGrab.txtGrabContrast.Text = Str(frmGrab.hsbGrabContrast.value)
   GrabContrast = frmGrab.hsbGrabContrast.value
End Sub

Private Sub hsbGrabField_Change()
   Rem change the number of fields accordingly
   frmGrab.txtGrabField.Text = Str(frmGrab.hsbGrabField.value)
   nFlashDelayField = frmGrab.hsbGrabField.value
End Sub

Private Sub hsbGrabLine_Change()
   Rem change the number of lines accrordingly
   frmGrab.txtGrabLine.Text = Str(frmGrab.hsbGrabLine.value)
   nFlashDelayLine = frmGrab.hsbGrabLine.value
End Sub

Private Sub Option1_Click()

End Sub

Private Sub optAny_Click()
   Rem set the ALIGN_ANY to true
   nAlign = ALIGN_ANY
   optEven.value = False
   optAny.value = True
   optOdd.value = False
End Sub

Private Sub optCCD_Click()
   Rem set the flashtype to ccd4000
   nFlashType = FLASHTYPE_CCD4000
End Sub

Private Sub optDual_Click()
   Rem set the flashtype to dualfield
   nFlashType = FLASHTYPE_DUALFIELD
End Sub

Private Sub optEven_Click()
   Rem set the ALIGN_EVEN to true
   nAlign = ALIGN_EVEN
   optEven.value = True
   optAny.value = False
   optOdd.value = False
End Sub

Private Sub optnone_Click()
   Rem set the value of Flashtype to none
   nFlashType = FLASHTYPE_NONE
End Sub

Private Sub optOdd_Click()
   Rem set ALIGN_ODD to true
   nAlign = ALIGN_ODD
   optEven.value = False
   optAny.value = False
   optOdd.value = True
End Sub
```

```
Private Sub optUniversal_Click()
   Rem set flashtype to universal
   nFlashType = FLASHTYPE_UNIVERSAL
End Sub
```

7.4 frmLoadImage

```
Private Sub cmdLoadCancel_Click()
   Rem cancel loading an image
   Rem hide load image form view activate flashpt form
   frmLoadImage.Hide
   frmFlashPT.Enabled = True
   frmLoadImage.Enabled = False
End Sub

Private Sub cmdLoadOK_Click()
Dim nRet As Integer
Dim nX As Integer
Dim nY As Integer
Dim nWidth As Integer
Dim nHeight As Integer

Rem load an image
Rem show load image form
frmLoadImage.Hide
frmFlashPT.Show
Rem get size of video form(flashpt)
nRet = FPV_GetVideoRect(DESTRECT, nX, nY, nWidth, nHeight)
Rem load appropiate file
nRet = FPV_LoadFile(dirLook.Path + "\" + filFind.FileName, nX, nY, nWidth, nHeight, 0, ByVal 0&, -1)
Rem freeze video/change menu accordingly
nRet = FPV_VideoLive(False, ALIGN_ANY)
frmFlashPT.mnuFreeze.Visible = False
frmFlashPT.mnuLive.Visible = True

frmFlashPT.Enabled = True
frmLoadImage.Enabled = False
End Sub

Private Sub dirLook_Change()
   Rem show files in current directory
   filFind.Path = dirLook.Path
End Sub

Private Sub drvChange_Change()
   Rem show directories in current drive
   dirLook.Path = drvChange.Drive
End Sub

Private Sub filFind_DblClick()
   Rem if double click on file then load file
   cmdLoadOK_Click
End Sub


Private Sub Form_Unload(Cancel As Integer)
   frmFlashPT.Enabled = True
   Unload frmLoadImage
End Sub
```

7.5 frmPrint

```
Private Sub cmdPrintCancel_Click()
   Rem  Hide Print Window, Enable FlashPT window
   frmPrint.Hide
   frmPrint.Enabled = False
   frmFlashPT.Enabled = True
End Sub

Private Sub cmdPrintOK_Click()

   Rem Print from the Clipboard
   frmPrint.Enabled = True
   frmFlashPT.Enabled = False
   Rem Check if Clipboard is empty
   Rem If not print otherwise show warning
   If Clipboard.GetFormat(vbCFBitmap) Then
   frmFlashPT.Picture = Clipboard.GetData(0) 'Loads appropiate type
   frmFlashPT.PrintForm    'Prints the form
   Else
      frmWarning.Show
   End If
   Rem Hide Print window
   frmPrint.Hide
   frmPrint.Enabled = False
   frmFlashPT.Enabled = True
End Sub

Private Sub Form_Unload(Cancel As Integer)
   frmFlashPT.Enabled = True
   Unload frmPrint
End Sub
```

7.6 frmSaveImage

```
Private Sub cmdSaveCancel_Click()
   Rem dont save anything/ hide save image form
   frmSaveImage.Hide
   frmSaveImage.Enabled = False
   frmFlashPT.Enabled = True
End Sub

Private Sub cmdSaveOK_Click()
   Dim nRet As Integer
   Dim nX As Integer
   Dim nY As Integer
   Dim nWidth As Integer
   Dim nHeight As Integer
   Dim TheFile As String
   Rem show save image form/turn off flashpt form
   frmSaveImage.Hide
   frmFlashPT.Show
   Rem the file selected by user
   Open frmSaveImage.dirSaveFile.Path + "\" + frmSaveImage.txtSaveImageName.Text For Output As #1
   nRet = FPV_GetVideoRect(DESTRECT, nX, nY, nWidth, nHeight)
   nRet = FPV_SaveFile(frmSaveImage.dirSaveFile.Path + "\" + frmSaveImage.txtSaveImageName.Text, nX, nY, nWidth, nHeight, 24,
0, ByVal 0&, -1)
   Close #1
   Rem disable save image / activate  flashpt form/ set menu accordingly
   frmSaveImage.Enabled = False
   frmFlashPT.Enabled = True
   frmFlashPT.mnuFreeze.Visible = False
   frmFlashPT.mnuLive.Visible = True
End Sub

Private Sub dirSaveFile_Change()
   Rem show files in current directory
   filFindSave.Path = dirSaveFile.Path
End Sub

Private Sub drvSaveFile_Change()
   Rem show current directories in current drive
   dirSaveFile.Path = drvSaveFile.Drive
End Sub

Private Sub filFindSave_Click()
   Rem name the file to be saved
   txtSaveImageName.Text = filFindSave.FileName
End Sub

Private Sub Form_Unload(Cancel As Integer)
   frmFlashPT.Enabled = True
   Unload frmSaveImage
End Sub
```

7.7 frmVideo

```
Private Sub chkGreenSync_Click()
   Dim nRet As Integer
   Dim nStandard As Integer

   nStandard = FPV_GetVideoStandard()
   If SyncGreen = False Then
      SyncGreen = True
   ElseIf SyncGreen = True Then
      SyncGreen = False
   End If
   nRet = FPV_SetVideoConfig(TYPE_RGB, nStandard, 0, SyncGreen)
End Sub

Private Sub cmdOK_Click()
   Rem hide menu activate flashpt form
   frmFlashPT.Enabled = True
   frmVideo.Enabled = False
   frmVideo.Hide
End Sub

Private Sub cmdVideoReset_Click()

   Dim nVidType As Integer
   Dim nVidStandard As Integer
   Dim nRet As Integer
   Rem Reset the Video values to their default values
   nVidType = FPV_GetVideoType()
   nVidStandard = FPV_GetVideoStandard()
   nVidSource = FPV_GetVideoSource()
   nRet = FPV_SetVideoConfig(nVidType, nStandard, nSource, False)
    nRet = FPV_SetVideoAdjustments(ADJUST_BRIGHTNESS, DefBrightness(nVidStandard, nVidType))
   frmVideo.vsbBrightness.value = DefBrightness(nVidStandard, nVidType)
   frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)

   nRet = FPV_SetVideoAdjustments(ADJUST_CONTRAST, DefContrast(nVidStandard, nVidType))
   frmVideo.vsbContrast.value = DefContrast(nVidStandard, nVidType)
   frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)

   nRet = FPV_SetVideoAdjustments(ADJUST_HUE, DefHue(nVidStandard, nVidType))
```

```
    frmVideo.vsbHue.value = DefHue(nVidStandard, nVidType)
    frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)

    nRet = FPV_SetVideoAdjustments(ADJUST_SATURATION, DefSaturation(nVidStandard, nVidType))
    frmVideo.vsbSaturation.value = DefSaturation(nVidStandard, nVidType)
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)

End Sub

Private Sub Form_Unload(Cancel As Integer)
    frmFlashPT.Enabled = True
    Unload frmVideo
End Sub

Private Sub optRS170_Click()
    Dim nRet As Integer
    Dim nStandard As Integer

    nStandard = FPV_GetVideoStandard()
    nRet = FPV_SetVideoConfig(TYPE_RS170, nStandard, 1, False)

    nRet = FPV_GetVideoAdjustments(ADJUST_BRIGHTNESS)
    frmVideo.vsbBrightness.value = nRet
    frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_CONTRAST)
    frmVideo.vsbContrast.value = nRet
    frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_HUE)
    frmVideo.vsbHue.value = nRet
    frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_SATURATION)
    frmVideo.vsbSaturation.value = nRet
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)

    frmVideo.optRS170.value = True

End Sub

Private Sub optComposite_Click()
    Dim nRet As Integer
    Dim nStandard As Integer

    Rem set values back to default values
    nStandard = FPV_GetVideoStandard()
    nRet = FPV_SetVideoConfig(TYPE_COMPOSITE, nStandard, 1, False)

    nRet = FPV_GetVideoAdjustments(ADJUST_BRIGHTNESS)
    frmVideo.vsbBrightness.value = nRet
    frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_CONTRAST)
    frmVideo.vsbContrast.value = nRet
    frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_HUE)
    frmVideo.vsbHue.value = nRet
    frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_SATURATION)
    frmVideo.vsbSaturation.value = nRet
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)

    frmVideo.optComposite.value = True
    frmVideo.optRS170.value = False
    frmVideo.optSVideo = False
    frmVideo.vsbSaturation.Enabled = True
    frmVideo.vsbHue.Enabled = True
    frmVideo.chkGreenSync.value = False
    frmVideo.chkGreenSync.Enabled = False
    frmVideo.vsbSharpness.Enabled = True

    End Sub

Private Sub optGreenSync_Click()
    Dim nRet As Integer
    Dim nStandard As Integer

    nStandard = FPV_GetVideoStandard()
    nRet = FPV_SetVideoConfig(TYPE_RGB, nStandard, 0, True)
End Sub

Private Sub optNTSC_Click()
    Dim nVidType As Integer
    Dim nVidSource As Integer
    Dim nRet As Integer
    Rem set the standard to NTSC
    nVidType = FPV_GetVideoType()
    nVidSource = FPV_GetVideoSource()
    nRet = FPV_SetVideoConfig(nVidType, STANDARD_NTSC, nVidSource, SyncGreen)
    nRet = FPV_GetVideoAdjustments(ADJUST_BRIGHTNESS)
    frmVideo.vsbBrightness.value = nRet
    frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_CONTRAST)
    frmVideo.vsbContrast.value = nRet
    frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_HUE)
    frmVideo.vsbHue.value = nRet
    frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_SATURATION)
    frmVideo.vsbSaturation.value = nRet
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)

End Sub

Private Sub optPAL_Click()
    Dim nRet As Integer
    Dim nVidType As Integer
    Dim nVidSource As Integer
    Rem set the standard to PAL
    nVidType = FPV_GetVideoType()
    nVidSource = FPV_GetVideoSource()
    nRet = FPV_SetVideoConfig(nVidType, STANDARD_PAL, nVidSource, SyncGreen)
    nRet = FPV_GetVideoAdjustments(ADJUST_BRIGHTNESS)
    frmVideo.vsbBrightness.value = nRet
    frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_CONTRAST)
    frmVideo.vsbContrast.value = nRet
    frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_HUE)
    frmVideo.vsbHue.value = nRet
    frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)

    nRet = FPV_GetVideoAdjustments(ADJUST_SATURATION)
    frmVideo.vsbSaturation.value = nRet
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)

End Sub

Private Sub optSVideo_Click()
Dim nRet As Integer
Dim nStandard As Integer
    Rem set type to svideo
    nStandard = FPV_GetVideoStandard()
    nRet = FPV_SetVideoConfig(TYPE_SVIDEO, nStandard, 1, False)
    Rem adjust according to SVIDEO
    nRet = FPV_GetVideoAdjustments(ADJUST_BRIGHTNESS)
    frmVideo.vsbBrightness.value = nRet
    frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_CONTRAST)
    frmVideo.vsbContrast.value = nRet
    frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_HUE)
    frmVideo.vsbHue.value = nRet
    frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)
    nRet = FPV_GetVideoAdjustments(ADJUST_SATURATION)
    frmVideo.vsbSaturation.value = nRet
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)

    frmVideo.vsbSharpness.Enabled = False
End Sub

Private Sub vsbBrightness_Change()
    Dim nRet As Integer

    Rem set brightness change
    nRet = FPV_SetVideoAdjustments(ADJUST_BRIGHTNESS, frmVideo.vsbBrightness.value)
    frmVideo.txtBrightness.Text = Str(frmVideo.vsbBrightness.value)
    nBrightness = frmVideo.vsbBrightness.value
    frmVideo.cmdVideoReset.Enabled = True
End Sub

Private Sub vsbContrast_Change()
Dim nRet As Integer

    Rem set contrast change
    nRet = FPV_SetVideoAdjustments(ADJUST_CONTRAST, frmVideo.vsbContrast.value)
    frmVideo.txtContrast.Text = Str(frmVideo.vsbContrast.value)
    nContrast = frmVideo.vsbContrast.value
    frmVideo.cmdVideoReset.Enabled = True
End Sub

Private Sub vsbHue_Change()
Dim nRet As Integer
    Rem set the hue change
    nRet = FPV_SetVideoAdjustments(ADJUST_HUE, frmVideo.vsbHue.value)
    frmVideo.txtHue.Text = Str(frmVideo.vsbHue.value)
    frmVideo.cmdVideoReset.Enabled = True
End Sub


Private Sub vsbSaturation_Change()
    Dim nRet As Integer
    Rem set the saturation change
    nRet = FPV_SetVideoAdjustments(ADJUST_SATURATION, frmVideo.vsbSaturation.value)
    frmVideo.txtSaturation.Text = Str(frmVideo.vsbSaturation.value)
    frmVideo.cmdVideoReset.Enabled = True
End Sub



Private Sub vsbSharpness_Change()
   Dim nRet As Integer
    Rem set the sharpess change
    nRet = FPV_SetVideoAdjustments(ADJUST_SHARPNESS, frmVideo.vsbSharpness.value)
    frmVideo.txtSharpness.Text = Str(frmVideo.vsbSharpness.value)
    frmVideo.cmdVideoReset.Enabled = True
End Sub
```

7.8 fp3d.bas

```
'// FP3D.BAS FlashPoint 3D API
'// Public prototypes and declarations for fp3d32.dll
'//


'// ********** Global Constants ***********************

'// Libary return values
Global Const RET_SUCCESS = 0
Global Const RET_ERROR = -1
Global Const RET_INICONFIG = 0
Global Const RET_DEFCONFIG = 1
Global Const RET_EEPROMCONFIG = 2
Global Const RET_AWFULLDCI = 0
Global Const RET_AWPARTIALDCI = 1
Global Const RET_AWNODCI = 2
Global Const RET_UNSUPPORTED = 1
Global Const RET_TIMEOUT = 1

'// Input video sizes
Global Const NTSCVIDEOWIDTH = 640
Global Const NTSCVIDEOHEIGHT = 480
Global Const PALVIDEOWIDTH = 760
Global Const PALVIDEOHEIGHT = 570

'// Video standards
Global Const STANDARD_NTSC = 0
Global Const STANDARD_PAL = 1
Global Const FP_STANDARDS = 2

'// Video types
Global Const TYPE_COMPOSITE = 0
Global Const TYPE_SVIDEO = 1
Global Const TYPE_RGB = 2
Global Const TYPE_BETA = 3
Global Const TYPE_RS170 = 4
Global Const TYPE_PROGRESSIVESCAN = &H80
Global Const MAXVITYPE = 4
Global Const FP_TYPES = 5

'// IRQ type for FPV_EnableIRQ() call
Global Const IRQ_VIDEO = 0
Global Const IRQ_VGA = 1

'// Rect types used in calls to FPV_GetVideoRect()
Global Const ACQRECT = 0
Global Const SOURCERECT = 1
Global Const DESTRECT = 2

'// I2C Addresses
Global Const MC44011ADDRW = &H8A
Global Const MC44011ADDRR = &H8B
Global Const EEPROMADDRW = &HA0
Global Const EEPROMADDRR = &HA1
Global Const TDA8444ADDRW = &H48
Global Const PCF8574CTRLADDRW = &H40
Global Const PCF8574CTRLADDRR = &H41
Global Const PCF8574DATAADDRW = &H42
Global Const PCF8574DATAADDRR = &H43
Global Const MV_DATA = &H44
Global Const MV_CONTROL = &H46
Global Const SAA7110ADDRW = &H9C
Global Const SAA7110ADDRR = &H9D

'// Special Flash delays
Global Const FLASHTYPE_NONE = 0
Global Const FLASHTYPE_UNIVERSAL = 1
Global Const FLASHTYPE_CCD4000 = 2
Global Const FLASHTYPE_DUALFIELD = 3
Global Const FLASHTYPE_DUALPULSE = 4
Global Const FLASHTYPE_SONYLTE = 5
Global Const FLASHTYPE_WAITSWITCH = 6
Global Const FLASHTYPE_CANON = 7
Global Const FLASHTYPE_WAITFLASH = 8
Global Const FLASHTYPE_WAITFLASHFIELD = 9
Global Const FLASHTYPE_ASYNCRESET = 10  'Pulse low
Global Const FLASHTYPE_ASYNCRESETH = 11  'Pulse high

'// Flash flags OR'ed with type
Global Const FLASHFLAG_FIELDREP = &H8000
Global Const FLASHFLAG_FIELDREPAVG = &H800
Global Const FLASHFLAG_REDGESMOOTH = &H4000
Global Const FLASHFLAG_FIELDREPO = &H2000
Global Const FLASHFLAG_FORCELIVE = &H1000


'// Field alignment values
Global Const ALIGN_EVEN = 0
Global Const ALIGN_ODD = 1
Global Const ALIGN_ANY = 2
Global Const ALIGN_NONE = 3
Global Const ALIGNFLAG_NOREFRESH = &H8000
Global Const ALIGNFLAG_LTE = &H4000

'// VGA Mode depth flag
Global Const VGAMODE_VESA = -1

'// Copy direction flag
Global Const COPYDIR_TOVGA = 0
Global Const COPYDIR_FROMVGA = 1
Global Const COPYDIR_FROMOFFSCREEN = 2
Global Const COPYDIR_FROMYUV = 4
Global Const COPYDIR_FROMYUV_PACKED = 4
Global Const COPYDIR_FROMYUV_UNPACKED = 5
Global Const COPYDIR_TOYUV = 6
Global Const COPYDIR_TOYUV_PACKED = 6
Global Const COPYDIR_TOYUV_UNPACKED = 7
Global Const COPYDIR_TODISPLAYMEM = 8
Global Const COPYDIR_FROMDISPLAYMEM = 9
Global Const COPYDIR_BLENDOVERLAYVIDEO = 10
Global Const COPYDIR_FROMOFFSCREENYUV = 11
Global Const COPYDIR_TOOFFSCREENYUV = 12

'// SaveFile flags
Global Const SAVEFILE_RLE = &H100
Global Const SAVEFILE_GRAY8 = &H200
Global Const SAVEFILE_FLIPX = &H400
Global Const SAVEFILE_COLOR8 = &H800
Global Const SAVEFILE_FLIPY = &H1000
Global Const SAVEFILE_FROMOFFSCREEN = &H8000
Global Const SAVEFILE_FROMYUV = &H2000
Global Const SAVEFILE_BLENDOVERLAYVIDEO = &H4000

'// Special RGB-YUV conversion constants
Global Const YUV_FULLSCALE = &H1000 + 24    '   /* YUV444, full scale */
Global Const YUV_CCIR601 = &H2000 + 24      '   /* YUV444, CCIR601 */
Global Const YUV_DEPTH = YUV_CCIR601        '   /* Archaic. Use YUV_CCIR601 */
Global Const YUVFLAG_PACKED = &H4000        ' /* Specifies that YUV is packed
Global Const YUVCONV_24 = &H8000
Global Const YUVCONV_16 = &H8
Global Const YUVCONV_8 = &H4
Global Const COLORDEPTH_YUV = &H5000 + 24

'// Video adjustment types
Global Const ADJUST_HUE = 0
Global Const ADJUST_SATURATION = 1
Global Const ADJUST_BRIGHTNESS = 2
Global Const ADJUST_CONTRAST = 3
Global Const ADJUST_REDBRIGHT = 4
Global Const ADJUST_BLUEBRIGHT = 5
Global Const ADJUST_SHARPNESS = 6
Global Const ADJUST_COMPMONO = 7
Global Const ADJUST_SYNCMODE = 8
Global Const ADJUST_FILTER = 9
Global Const ADJUST_GREENBRIGHT = 10
Global Const ADJUST_GAIN = 11
Global Const ADJUST_REDCONTRAST = 12
Global Const ADJUST_BLUECONTRAST = 13
Global Const ADJUST_GREENCONTRAST = 14

'// MIN and MAX adjust vals
Global Const MINADJUSTVAL = 0
Global Const MAXADJUSTVAL = &H3F
'// IRIS level is 0-4095
Global Const MAXIRISVAL = &HFFF

'// Windows DCI window watch types for FPV_AutoWindow
Global Const AUTOWIN_FREEZE = &H1
Global Const AUTOWIN_CLIP = &H2
Global Const AUTOWIN_POS = &H4
Global Const AUTOWIN_SIZE = &H8
Global Const AUTOWIN_ASPECT = &H10
Global Const AUTOWIN_ADJUSTWINDOW = &H20
Global Const AUTOWIN_FORCEPAINT = &H40
Global Const AUTOWIN_OVERSIZE = &H80
Global Const AUTOWIN_REFRESH = &H100
Global Const AUTOWIN_TOPMOST = &H200
Global Const AUTOWIN_LIVEDACTWIN = &H400
Global Const AUTOWIN_LIVEDACTAPP = &H800
Global Const AUTOWIN_FORCEAW1 = &H1000
Global Const AUTOWIN_FORCEAW2 = &H2000
Global Const AU0TWIN_MASKBLEEDFIX = &H4000
Global Const AUTOWIN_DCIFREEZEFIX = &H8000
Global Const AUTOWIN_DRAWKEY = &H10000

'// Screen to DIB flags
Global Const STD_ONSCREEN = &H0
Global Const STD_CLIPBOARD = &H1
Global Const STD_GRAY8 = &H200
Global Const STD_OFFSCREEN = &H8000
Global Const STD_OVERLAY = &H4000

'// MaskDraw OPs
Global Const MD_INIT = 0
Global Const MD_CLEANUP = 1
Global Const MD_SETUP = 2
Global Const MD_RESET = 3
Global Const MD_UPDATE = 4
Global Const MD_LINE = 5
Global Const MD_UNLINE = 6
Global Const MD_BOX = 7
Global Const MD_UNBOX = 8
Global Const MD_CROSSHAIR = 9
Global Const MD_UNCROSSHAIR = 10
Global Const MD_CIRCLE = 11
Global Const MD_UNCIRCLE = 12
Global Const MD_RECT = 13
Global Const MD_UNRECT = 14

'// Get/Set Misc indexes
'// Return connected video input type - type,standard,source,sog
'// in byte0,byte1,byte2,byte3
Global Const MISCPARM_INPUTVID = &H1
Global Const MISCPARM_AWSELBOX = &H2
Global Const MISCPARM_MEMPTR = &H3
Global Const MISCPARM_MEMSIZE = &H4
Global Const MISCPARM_MEMFRAMES = &H5
Global Const MISCPARM_SINGLEFIELD = &H6     ' ALIGN_EVEN, ALIGN_ODD or ALIGN_ANY
Global Const MISCPARM_MEMPHYSADDR = &H7
Global Const MISCPARM_MEMOFFSET = &H8
Global Const MISCPARM_MEMCOPYOFFSET = &H9
Global Const MISCPARM_VIDEOLOCK = &HA
Global Const MISCPARM_AWBASEMASK = &HB
Global Const MISCPARM_OUTSWITCH = &HC
Global Const MISCPARM_WSTIMEOUT = &HD
Global Const MISCPARM_PIXBUFDEPTH = &HE
Global Const MISCPARM_RESERVED = &HF
Global Const MISCPARM_ALWAYSLIVE = &H10
Global Const MISCPARM_LTEMS = &H11
Global Const MISCPARM_CAMERATYPE = &H12
Global Const MISCPARM_SYNCMODE = &H13
Global Const MISCPARM_WAITFLASHPARMS = &H14

Global Const MISCPARM_AGCENABLE = &H15
Global Const MISCPARM_POWEROUT = &H16
Global Const MISCPARM_CLEARSWITCH = &H17
Global Const MISCPARM_INVERTVIDEO = &H18
Global Const MISCPARM_SELBOXWIDTH = &H19
Global Const MISCPARM_SELBOXCOLOR = &H1A

Global Const MISCPARM_UPDATEMASK = &H1B
Global Const MISCPARM_AWKILLPAINTMSG = &H1C
Global Const MISCPARM_ONEFIELD = &H1D
Global Const MISCPARM_NONINTERLACED = &H1E
Global Const MISCPARM_MIRRORVIDEO = &H1F
Global Const MISCPARM_FRAMEMASK = &H20
Global Const MISCPARM_IOTRIGGERS = &H21
Global Const MISCPARM_WINDOWNUMBER = &H22
Global Const MISCPARM_LATCHINSWITCH = &H23
Global Const MISCPARM_CURCONSECFRAME = &H24
Global Const MISCPARM_MICROPHWRESET = &H25
Global Const MISCPARM_CURCONSECFIELD = &H26
Global Const MISCPARM_KEYMODE = &H27
Global Const MISCPARM_KEYCOLORLOW = &H28
Global Const MISCPARM_KEYCOLORHIGH = &H29
Global Const MISCPARM_YUVCAPTUREWIDTH = &H2B
Global Const MISCPARM_YUVCAPTUREHEIGHT = &H2C
Global Const MISCPARM_YUVCAPTUREPITCH = &H2E
Global Const MISCPARM_CONVERT_LUMAONLY = &H2D
Global Const MISCPARM_BOARD = &H2E
Global Const MISCPARM_CCIR601 = &H30
Global Const MISCPARM_NOVIDOUTRESETONTERMINATE = &H31
Global Const MISCPARM_PICVER = &H32
Global Const MISCPARM_INPUTLATCH = &H33
Global Const MISCPARM_FORCEONEFIELD = &H34
Global Const MISCPARM_MAXVIDSOURCES = &H35
Global Const MISCPARM_SCALEMODE = &H36
Global Const MISCPARM_FORCEBOARD = &H100

'// SetPalette indexes
Global Const SETPAL_CREATEGRAY8 = &H0
Global Const SETPAL_DELGRAY8 = &H1
Global Const SETPAL_SETCURRENT = &H2
Global Const SETPAL_SELHWND = &H3
Global Const SETPAL_DESELHWND = &H4
Global Const SETPAL_SELHDC = &H5
Global Const SETPAL_DESELHDC = &H6
Global Const SETPAL_ILUT = &H7
Global Const SETPAL_RS170 = &H8
Global Const SETPAL_ILUT2 = &H9
Global Const SETPAL_ILUT3 = &HA
Global Const SETPAL_ILUT4 = &HB

'// MISCREG Indexes
Global Const MISCREG_MMIO = &H1
Global Const MISCREG_PORTIO = &H2
Global Const MISCREG_BIOS = &H3

'// Board config flags
Global Const CFG_INTPOLFIX = &H1
Global Const CFG_NOSERIAL = &H2
Global Const CFG_PROGFREQ = &H4
Global Const CFG_IRISCTRL = &H8
Global Const CFG_GRAY8 = &H10
Global Const CFG_NORGB = &H20
Global Const CFG_MV = &H40
Global Const CFG_INVFLASH = &H80
Global Const CFG_MICROP = &H100
Global Const CFG_RS170 = &H200
Global Const CFG_VIDOUT = &H400

'// Camera Types
Global Const CAMTYPE_DC330 = &H1
Global Const CAMTYPE_SONY950 = &H2
Global Const CAMTYPE_DAGESLG = &H3
Global Const GENERICMASTER = &H4000

'// EEPROM byte count
Global Const EEPROMSIZE = 256

' key constants
Global Const KEYMODE_NONE = 0
Global Const KEYMODE_COLORKEY = 1
Global Const KEYMODE_CHROMAKEY = 2
Global Const KEYMODE_DESKTOP_ONLY = 3

' YUV conversion flags
Global Const YUVCONV_LUMAONLY = &H1          ' Source/dest is luminance only
Global Const YUVCONV_CCIR601 = &H2          ' Source/dest is compressed YUV422 CCIR601 (else assumes uncompressed
YUV444)
Global Const YUVCONV_PACKED = &H4            ' Source/dest YUV is packed 422
Global Const YUVCONV_OUT32 = &H8

' FPV_SetCaptureBuffer defines
Global Const CAPBUF_PTRTOOFFSET = 0
Global Const CAPBUF_SETVIDEO = 1
Global Const CAPBUF_SETVIDEODEFAULTS = 2
Global Const CAPBUF_SETOVERLAY = 3
Global Const CAPBUF_SETOVERLAYDEFAULTS = 4
Global Const CAPBUF_SINGLEBUFFER = 5
Global Const CAPBUF_DOUBLEBUFFER = 6
Global Const CAPBUF_TRIPLEBUFFER = 7
Global Const CAPBUF_DEFAULTBUFFER = 8


' camera types
Global Const CUSTOMSYNC = &H8000
Global Const CS_WENRISE = &H0   'WEN Polarity
Global Const CS_WENFALL = &H10
Global Const CS_INTPIXCLK = &H0   'PIX clk
Global Const CS_EXTPIXCLK = &H20
Global Const CS_PLLADCLK = &H0   'ADCLK
Global Const CS_7110ADCLK = &H40
Global Const CS_TWOSYNC = &H80  'Sync on both edges of WEN
Global Const CS_HSCTLHREFPLL = &H0   'HSYNC control
Global Const CS_HSCTL7110 = &H100
Global Const CS_HSCTLEXT = &H200
Global Const CS_HSCTLSSEP = &H300
Global Const CS_VSCTLSSEP = &H0   'VSYNC control
Global Const CS_VSCTL7110 = &H400
Global Const CS_VSCTLEXT = &H800
Global Const CS_VSCTLEXTPS = &HC00


'// ********* TYPEDEFS *****************

'// Version info struct
Type VERSIONINFO
      wLibVersion As Integer
      wBetaVersion As Integer
      wDebugStatus As Integer
      wBusType As Integer
      wVGAMem As Integer
      wSClk As Integer
      wBWidthLimit As Integer
      wBIOSVersion As Integer
      dwBoardCfg As Long
      wBoardRev As Integer
      dwSerialNum As Long
      wScalerRev As Integer
      wDecoderType As Integer
      wEEPROMVer As Integer
      wReserved(15) As Integer
End Type


'// File resolution info struct
Type FILERESINFO
      wWidth As Integer
      wHeight As Integer
      wDepth As Integer
      wFlags As Integer
End Type

'// AutoWindow info struct
Type AUTOWININFO
      hWnd As Long
      hRsv As Integer
      lpMaskBuf As Long
      wMaskWidth As Integer
      wMaskHeight As Integer
      wOccluded As Integer
      bAWLive As Integer
      lpRefreshBuf As Long
      wMaskDelay As Integer
      hWinWatch As Long
      wReserved(18) As Integer
```

```
End Type

Type OVLINIT
  lpSurface As Long
  dwPitch As Long
  dwWidth As Long
  dwHeight As Long
  dwReserved(12) As Long
End Type

'/ Allows setting video to different surfaces
Type CAPTUREBUFFERINFO
  lpBuffer As Long
  dwOffset As Long
  wBufferNum As Integer
  wPitch As Integer
  dwReserved(12) As Long
End Type

'// *********** Prototypes ***********************

'// Library routines return 0 or value for success, <0 for failure

'// Locate the FPV board, don't change any regs
Declare Function FPV_Locate Lib "fp3d32.dll" () As Integer
'// Init the board using the currently loaded configuration
Declare Function FPV_Init Lib "fp3d32.dll" () As Integer
'// Shut down in preparation for exit
Declare Function FPV_Cleanup Lib "fp3d32.dll" () As Integer

'// Load configuration from .INI file.  A NULL file name loads the default vals
Declare Function FPV_LoadConfig Lib "fp3d32.dll" (szIniFile As Any) As Integer
'// Save configuration to .INI file
Declare Function FPV_SaveConfig Lib "fp3d32.dll" (ByVal szIniFile As String) As Integer

'// Save current configuration to the EEPROM
'// Developers should NOT make this call
Declare Function FPV_SaveConfigToEEPROM Lib "fp3d32.dll" (ByVal wValid As Integer, ByVal wBoardRev As Integer, ByVal
dwSerialNum As Long, ByVal dwBoardCfg As Long) As Integer

'// Write an I2C register
Declare Function FPV_SetI2CReg Lib "fp3d32.dll" (ByVal nAddr As Integer, ByVal nSubAddr As Integer, ByVal nVal As Integer) As
Integer
'// Read an I2C register
Declare Function FPV_GetI2CReg Lib "fp3d32.dll" (ByVal nAddr As Integer, ByVal nSubAddr As Integer) As Integer
'// Write a viper register
Declare Function FPV_SetViperReg Lib "fp3d32.dll" (ByVal nRegNum As Integer, ByVal nVal As Integer) As Integer
'// Read a viper register
Declare Function FPV_GetViperReg Lib "fp3d32.dll" (ByVal nRegNum As Integer) As Integer
'// Set the W32 read segment
Declare Function FPV_SetW32ReadSeg Lib "fp3d32.dll" (ByVal wSeg As Integer) As Integer
'// Set/Get IMA regs
Declare Function FPV_SetIMAReg Lib "fp3d32.dll" (ByVal nReg As Integer, ByVal nVal As Integer) As Integer
Declare Function FPV_GetIMAReg Lib "fp3d32.dll" (ByVal nReg As Integer) As Integer
'//Send a Viper FS
Declare Function FPV_SendViperFS Lib "fp3d32.dll" (ByVal nAlign As Integer) As Integer
'//Set Viper in and out modes
Declare Function FPV_SetColorSpace Lib "fp3d32.dll" (ByVal nInSpace As Integer, ByVal nOutSpace As Integer) As Integer

'// Return version info
Declare Function FPV_GetVersionInfo Lib "fp3d32.dll" (pVInfo As VERSIONINFO) As Integer

'// Set the VGA mode (DOS only)
Declare Function FPV_SetVGAMode Lib "fp3d32.dll" (ByVal nMode As Integer, ByVal nBitsPerPix As Integer) As Integer
'// Return the VGA mode (DOS only)
```

```
Declare Function FPV_GetVGAMode Lib "fp3d32.dll" (ByVal bVESA As Integer) As Integer

'// Convert pixel vals to a new resolution
Declare Function FPV_ConvertPixel Lib "fp3d32.dll" (ByVal nWidth As Integer, ByVal nDepthCur As Integer, ByVal nDepthNew As Integer, lpValCur As Any, lpValNew As Any, lpCMap As Any) As Integer
'// Set a VGA pixel
Declare Function FPV_SetVGAPixel Lib "fp3d32.dll" (ByVal nX As Integer, ByVal nY As Integer, ByVal lVal As Long) As Integer
'// Set a VGA rectangle to a uniform value
Declare Function FPV_SetVGARect Lib "fp3d32.dll" (ByVal nX As Integer, ByVal nY As Integer, ByVal nWidth As Integer, ByVal nHeight As Integer, ByVal lVal As Long) As Integer
'// Copy a VGA rectangle to or from main memory
'// Direction 0==to VGA 1==from VGA
Declare Function FPV_CopyVGARect Lib "fp3d32.dll" (ByVal nX As Integer, ByVal nY As Integer, ByVal nWidth As Integer, ByVal nHeight As Integer, hpPixBuf As Any, ByVal nPitch As Integer, ByVal nCopyDir As Integer) As Integer
'// Create a DIB from a screen rectangle
Declare Function FPV_ScreenToDIB Lib "fp3d32.dll" (ByVal nX As Integer, ByVal nY As Integer, ByVal nWidth As Integer, ByVal nHeight As Integer, ByVal nFlags As Integer, hpDIBBuf As Any) As Integer

'// Set the video type and standard
'// Config types are: 0 source, 1 standard, 2 type
Declare Function FPV_SetVideoConfig Lib "fp3d32.dll" (ByVal nType As Integer, ByVal nStandard As Integer, ByVal nSource As Integer, ByVal bSyncOnGreen As Integer) As Integer
'// Get the video type, standard, or source
Declare Function FPV_GetVideoType Lib "fp3d32.dll" () As Integer
Declare Function FPV_GetVideoStandard Lib "fp3d32.dll" () As Integer
Declare Function FPV_GetVideoSource Lib "fp3d32.dll" () As Integer

'// Get or set a misc value
Declare Function FPV_GetMiscParm Lib "fp3d32.dll" (ByVal nIndex As Integer, lpRetBuf As Any) As Integer
Declare Function FPV_SetMiscParm Lib "fp3d32.dll" (ByVal nIndex As Integer, ByVal lVal As Long) As Integer

'// Get or set a misc reg value
Declare Function FPV_SetMiscReg Lib "fp3d32.dll" (ByVal nIndex As Integer, ByVal dwReg As Long, ByVal dwVal As Long) As Integer
Declare Function FPV_GetMiscReg Lib "fp3d32.dll" (ByVal nIndex As Integer, ByVal dwReg As Long) As Integer

'// Set an 8 bit palette
Declare Function FPV_SetPalette Lib "fp3d32.dll" (ByVal nType As Integer, ByVal lHandle As Long) As Integer

'// Set input(source) video rect
Declare Function FPV_SetInputWindow Lib "fp3d32.dll" (ByVal nSrcX As Integer, ByVal nSrc As Integer, ByVal nSrcWidth As Integer, ByVal nSrcHeight As Integer) As Integer
'// Set X and Y offset into the input(source) window
Declare Function FPV_SetInputOffset Lib "fp3d32.dll" (ByVal nOffX As Integer, ByVal nOffY As Integer) As Integer

'// Set video window size and position
Declare Function FPV_SetVideoWindow Lib "fp3d32.dll" (ByVal nDstX As Integer, ByVal nDst As Integer, ByVal nDstWidth As Integer, ByVal nDstHeight As Integer, ByVal bScale As Integer) As Integer
Declare Function FPV_VideoOffscreen Lib "fp3d32.dll" (ByVal nDstWidth As Integer, ByVal nDstHeight As Integer, ByVal nDstDepth As Integer, ByVal bScale As Integer) As Integer

'// Set video acquisition rectangle
Declare Function FPV_SetAcqRect Lib "fp3d32.dll" (ByVal nAcqX As Integer, ByVal nAcqY As Integer, ByVal nAcqWidth As Integer, ByVal nAcqHeight As Integer) As Integer
'// Set and get the video x and y delays
Declare Function FPV_SetXYDelay Lib "fp3d32.dll" (ByVal nXDelay As Integer, ByVal nYDelay As Integer) As Integer
Declare Function FPV_GetXDelay Lib "fp3d32.dll" () As Integer
Declare Function FPV_GetYDelay Lib "fp3d32.dll" () As Integer

'// Get various video rects
Declare Function FPV_GetVideoRect Lib "fp3d32.dll" (ByVal nRectType As Integer, pnX As Integer, pnY As Integer, pnWidth As Integer, pnHeight As Integer) As Integer

'// Vsync/LSync operations
'// Get the current field type
```

```
Declare Function FPV_GetFieldType Lib "fp3d32.dll" () As Integer
'// Wait for a particular field type
Declare Function FPV_WaitFieldType Lib "fp3d32.dll" (ByVal nFieldType As Integer) As Integer
'// Wait a specified number of fields
Declare Function FPV_WaitVSync Lib "fp3d32.dll" (ByVal nSyncCnt As Integer) As Integer
'// Wait a specified number of lines
Declare Function FPV_WaitHSync Lib "fp3d32.dll" (ByVal nSyncCnt As Integer) As Integer
'// Wait a specified number of milliseconds
Declare Function FPV_WaitMS Lib "fp3d32.dll" (ByVal nMSCnt As Integer) As Integer

'// Adjust video parms
'// Adjustment types are: 0==hue 1==sat, 3==bright, 4==contrast
Declare Function FPV_SetVideoAdjustments Lib "fp3d32.dll" (ByVal nType As Integer, ByVal nVal As Integer) As Integer
'// Get current video parms
Declare Function FPV_GetVideoAdjustments Lib "fp3d32.dll" (ByVal nType As Integer) As Integer
'// Get video parms for any standard or type
Declare Function FPV_GetSTVideoAdjustments Lib "fp3d32.dll" (ByVal nStandard As Integer, ByVal nVideoType As Integer, ByVal
nType As Integer) As Integer

'// Freeze video with or without flash
'// Alignment values: ALIGN_ANY  ALIGN_ODD  ALIGN_EVEN
'// Flash delay values: FLASHTYPE_NONE FLASHTYPE_AUTO FLASHTYPE_MANUAL
Declare Function FPV_VideoGrab Lib "fp3d32.dll" (ByVal nAlign As Integer, ByVal nFlashType As Integer, ByVal nFlashDelayF As
Integer, ByVal nFlashDelayL As Integer) As Integer
'// Start live video
Declare Function FPV_VideoLive Lib "fp3d32.dll" (ByVal bLive As Integer, ByVal nAlign As Integer) As Integer
'// Get live status
Declare Function FPV_GetLiveStatus Lib "fp3d32.dll" () As Integer

'// Enable or Disable the video mask ram
Declare Function FPV_EnableVideoMask Lib "fp3d32.dll" (ByVal bEnable As Integer) As Integer
'// Set the video mask ram
Declare Function FPV_SetVideoMask Lib "fp3d32.dll" (ByVal nX As Integer, ByVal nY As Integer, ByVal nWidth As Integer, ByVal
nHeight As Integer, ByVal nPitch As Integer, lpMaskBuf As Any) As Integer
'// Change the mask write delay
Declare Function FPV_SetMaskDelay Lib "fp3d32.dll" (ByVal nMDelay As Integer) As Integer
'// Set the mask based on a key color
Declare Function FPV_VGARectToMask Lib "fp3d32.dll" (ByVal dwKeyColor As Long, ByVal nX As Integer, ByVal nY As Integer,
ByVal nWidth As Integer, ByVal nHeight As Integer, lpMaskBuf As Any, ByVal nPitch As Integer) As Integer
'// Draw into video mask
Declare Function FPV_MaskDraw Lib "fp3d32.dll" (ByVal nOp As Integer, ByVal nX1 As Integer, ByVal nY1 As Integer, ByVal nX2
As Integer, ByVal nY2 As Integer, ByVal dwParam1 As Long, ByVal dwParam2 As Long) As Integer

'// Check external switch
Declare Function FPV_CheckSwitch Lib "fp3d32.dll" (ByVal nBaseAddr As Integer, ByVal nTimeOutMS As Integer) As Integer

'// Enable or disable VGA or video VSync
Declare Function FPV_EnableIRQ Lib "fp3d32.dll" (ByVal nIRQType As Integer, ByVal nIRQNum As Integer, ByVal bEnable As
Integer, lpISR As Any) As Integer
'// Clear an IRQ
Declare Function FPV_ClearIRQ Lib "fp3d32.dll" (ByVal nIRQType As Integer, ByVal nIRQNum As Integer) As Integer
'// Return the IRQ used
Declare Function FPV_GetIRQNumber Lib "fp3d32.dll" () As Integer

'// Serial port functions
Declare Function FPV_SetupSerial Lib "fp3d32.dll" (ByVal nBaseIO As Integer, ByVal nIRQ As Integer, ByVal nBaud As Integer,
ByVal nDataBits As Integer, ByVal nParity As Integer, ByVal nStopBits As Integer) As Integer
Declare Function FPV_ReadSerialBytes Lib "fp3d32.dll" (lpRcvBuf As Any, ByVal nRcvCnt As Integer, ByVal nTimeOutMS As
Integer) As Integer
Declare Function FPV_WriteSerialBytes Lib "fp3d32.dll" (lpTxBuf As Any, ByVal nTxCnt As Integer, ByVal nTimeOutMS As
Integer) As Integer
Declare Function FPV_GetSerialReg Lib "fp3d32.dll" (ByVal nReg As Integer) As Integer
Declare Function FPV_SetSerialReg Lib "fp3d32.dll" (ByVal nReg As Integer, ByVal nRegVal As Integer) As Integer

'//Mimic Windows private profile functions
```

```
Declare Function FPV_GetPrivateProfileString Lib "fp3d32.dll" (ByVal lpszSection As String, ByVal lpszEntry As String, ByVal lpszDefault As String, ByVal lpszReturnBuffer As String, ByVal cbReturnBytes As Integer, ByVal lpszFileName As String) As Integer
Declare Function FPV_WritePrivateProfileString Lib "fp3d32.dll" (ByVal lpszSection As String, ByVal lpszEntry As String, ByVal lpszString As String, ByVal lpszFileName As String) As Integer
Declare Function FPV_OpenPrivateProfileString Lib "fp3d32.dll" (ByVal lpszFileName As String, ByVal bWrite As Integer) As Integer
Declare Function FPV_ClosePrivateProfileString Lib "fp3d32.dll" (ByVal bWrite As Integer) As Integer

'// Load and save file will call seperate Dlls based on the file extension
'// This allows new file formats to be added without changing the main DLL
'// For DOS .TGA and .DIB are supported
'// A NULL hpPixBuf will cause the image to be loaded directly into VGA mem
Declare Function FPV_FileResInfo Lib "fp3d32.dll" (ByVal szPathName As String, lpFileRes As FILERESINFO) As Integer
Declare Function FPV_LoadFile Lib "fp3d32.dll" (ByVal szPathName As String, ByVal nX As Integer, ByVal nY As Integer, ByVal nWidth As Integer, ByVal nHeight As Integer, ByVal nFlags As Integer, hpPixBuf As Any, ByVal nPitch As Integer) As Integer
Declare Function FPV_SaveFile Lib "fp3d32.dll" (ByVal szPathName As String, ByVal nX As Integer, ByVal nY As Integer, ByVal nWidth As Integer, ByVal nHeight As Integer, ByVal nDepth As Integer, ByVal nFlags As Integer, hpPixBuf As Any, ByVal nPitch As Integer) As Integer

'// Available with programmable frequency boards
Declare Function FPV_SetVidFreq Lib "fp3d32.dll" (ByVal nNTSCBits As Integer, ByVal nPALBits As Integer) As Integer
'// Available with programmable iris boards
Declare Function FPV_SetIrisLevel Lib "fp3d32.dll" (ByVal nIrisBits As Integer) As Integer

'// Available under Windows with DCI
Declare Function FPV_AutoWindow Lib "fp3d32.dll" (ByVal hWnd As Long, ByVal nXOff As Integer, ByVal nYOff As Integer, ByVal nWidthAdj As Integer, ByVal nHeightAdj As Integer, ByVal lFlags As Long) As Integer
Declare Function FPV_AutoWindowInfo Lib "fp3d32.dll" (pAWInfo As Any) As Integer

'// Only available in a debug build
Declare Function FPV_DumpViperRegs Lib "fp3d32.dll" () As Integer

'//////////////////////////////////////////////////////////////////////////////
'//////////////////////////////////////////////////////////////////////////////

'// Get occlusion mode

Declare Function FPV_GetKeyMode Lib "fp3d32.dll" () As Integer

'// Set occlusion mode

Declare Function FPV_SetKeyMode Lib "fp3d32.dll" (ByVal nMode As Integer) As Integer

'// Set occlusion color key / chroma key RGB value

Declare Function FPV_SetKeyValue Lib "fp3d32.dll" (ByVal nMode As Integer, ByVal nRedLow As Integer, ByVal nRedHigh As Integer, ByVal nGreenLow As Integer, ByVal nGreenHigh As Integer, ByVal nBlueLow As Integer, ByVal nBlueHigh As Integer) As Integer

'// Get occlusion color key for a given mode

Declare Function FPV_GetKeyValue Lib "fp3d32.dll" (ByVal mode As Integer, pRedLow As Integer, pRedHigh As Integer, pGreenLow As Integer, pGreenHigh As Integer, pBlueLow As Integer, pBlueHigh As Integer) As Integer

'// Enable/disable video window
Declare Function FPV_OverlayEnable Lib "fp3d32.dll" (ByVal bEnable As Integer) As Integer

'// Get video window enabled status
Declare Function FPV_IsOverlayEnabled Lib "fp3d32.dll" () As Integer

'// Get frame buffer selector
Declare Function FPV_GetFrameBufferPointer Lib "fp3d32.dll" () As Long

'// Get capture buffer offset
Declare Function FPV_GetCaptureBufferOffset Lib "fp3d32.dll" () As Long
```

```
'// Set RS-170 Lookup Table value
Declare Function FPV_SetRS170LUT Lib "fp3d32.dll" (ByVal ix As Integer, ByVal value As Integer) As Integer

'// Read write to video, DOS only
Declare Function FPV_ReadVideoByte Lib "fp3d32.dll" (ByVal nAddress As Integer) As Integer
Declare Function FPV_WriteVideoByte Lib "fp3d32.dll" (ByVal nAddress As Integer, ByVal nByte As Integer) As Integer
Declare Function FPV_ReadVideoMemory Lib "fp3d32.dll" (ByVal nAddress As Integer, pDest As Any, ByVal nCount As Integer) As
Integer
Declare Function FPV_WriteVideoMemory Lib "fp3d32.dll" (ByVal nAddress As Integer, pSource As Any, ByVal nCount As Integer)
As Integer

'// Get library build number
Declare Function FPV_GetLibBuild Lib "fp3d32.dll" () As Integer

'// Read overlay data, converting to desired format
Declare Function FPV_GetOverlayRect Lib "fp3d32.dll" (ByVal x As Integer, ByVal y As Integer, ByVal nWidth As Integer, ByVal
nHeight As Integer, pDest As Any, ByVal nDestDepth As Integer, ByVal nPitch As Integer) As Integer

'// Quickest way to read overlay memory
Declare Function FPV_ReadOverlayMemory Lib "fp3d32.dll" (ByVal nOffset As Long, ByVal nSize As Long, pDest As Any) As
Integer

"// YUV conversion
Declare Function FPV_ConvertYuvToRgb Lib "fp3d32.dll" (pSource As Any, pDest As Any, ByVal nWidth As Integer, ByVal nHeight
As Integer, ByVal nSourcePitch As Integer, ByVal nDestPitch As Integer, ByVal nFlags As Integer) As Integer
Declare Function FPV_ConvertRgbToYuv Lib "fp3d32.dll" (pSource As Any, pDest As Any, ByVal nWidth As Integer, ByVal nHeight
As Integer, ByVal nSourcePitch As Integer, ByVal nDestPitch As Integer, ByVal nFlags As Integer) As Integer


'// Procs to pack/unpack YUV 444/422
Declare Function FPV_PackYuv Lib "fp3d32.dll" (sSource As Any, sDest As Any, ByVal nWidth As Integer) As Integer
Declare Function FPV_UnpackYuv Lib "fp3d32.dll" (sSource As Any, sDest As Any, ByVal nWidth As Integer) As Integer

Global Const BOARD_LITE = 0
Global Const BOARD_PLUS = 1
Global Const BOARD_PRO = 2
Global Const BOARD_UNKNOWN = &H10
```

7.9 fpg.bas

```
Option Explicit

Declare Function GetDeviceCaps Lib "GDI32" (ByVal hDC As Integer, ByVal nIndex As Integer) As Integer
Declare Function GetDesktopWindow Lib "User32" () As Long
Declare Function GetDC Lib "User32" (ByVal hWnd As Long) As Long
Declare Function ReleaseDC Lib "User32" (ByVal hWnd As Long, ByVal hDC As Long) As Long

Rem Arrays to hold Default values for the different standards and types
Public DefBrightness(0 To 1, 0 To 3) As Integer
Public DefContrast(0 To 1, 0 To 3) As Integer
Public DefHue(0 To 1, 0 To 3) As Integer
Public DefSaturation(0 To 1, 0 To 3) As Integer
Public SyncGreen As Boolean
```

Appendix B

Cost Breakdown of the Modular Mechatronic Internet Control System

This appendix presents a breakdown of the manufacturing costs that are incurred in the development of Internet control system. The breakdown of costs are presented as a spreadsheet in landscape format.

Cost Breakdown of the Modular Mechatronic Internet Control System

Network Peripherals			
DE-809TC Ethernet Hub		1000.00	
Server network card (3Com Etherlink 10BASE-T PCI card)		150.00	
Camera Client network card (3Com Etherlink 10BASE-T PCI card)		150.00	
Conveyer Client network card (3Com Etherlink 10BASE-T PCI card)		150.00	
Robot Client network card (3Com Etherlink 10BASE-T PCI card)		150.00	
LAN Cables (Twisted Pair and RJ-45 connectors)		300.00	
Cable Tray		125.00	
	Sub Total		**R 2,025.00**
Visual Feedback Module			
Tedelex Private Eye Camera	4x	250.00	
Coaxial Cables		350.00	
Camera Power Supply		230.00	
Camera Selector Module		65.00	
	Sub Total		**R 1,645.00**
Host Server Module			
PII-400 Computer		4500.00	
FlashPoint3D Frame Grabber		5500.00	
	Sub Total		**R 10,000.00**
Camera Client Module			
PI-166 Computer		3500.00	
PCL-836 I/O Card		4200.00	
PCLD-880 Terminal connector card		1200.00	
	Sub Total		**R 8,900.00**
Conveyer Client Module			
Power Cabinet power supply		4500.00	
Relay Power Modules	10x	35.00	
PI-120 Computer		3000.00	
PCL-836 I/O Card		4200.00	
PCLD-880 Terminal connector card		1200.00	
	Sub Total		**R 13,250.00**

Cost Breakdown of the Modular Mechatronic Internet Control System (Continue)

Robot Client Module		8500.00	
PIII-500 Computer	2x	5500.00	
PCL-832 Servo Card		3250.00	
PC-30GA Interface card		2500.00	
Feedback circuitry switch mode power supply	6x	250.00	
LM12 Driver circuit		200.00	
Signal conditioning circuitry		850.00	
Manufacturing and material cost of controller cabinet		600.00	
Cabling	**Sub Total**		**R 28,400.00**

Total Development Cost : **R 64,220.00**