dmu.ac.uk
DE MONTFORT UNIVERSITY
LEICESTER

Integrated Real-Virtuality System and Environments for Advanced Control System Developers and Machines Builders

MOHAMED HUSSEIN

Mechatronics Research Centre
Faculty of Computing Sciences and Engineering
De Montfort University, United Kingdom

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

April 2008

.... to my wife Nora, for her unconditional love and support.

Acknowledgements

I would like to express my sincere gratitude to my supervisory team: Professor Philip Moore, Dr. Jun Sheng Pu and Mr. Chi Biu Wong, at De Montfort University, United Kingdom, for their support, ideas, guidance and encouragement throughout my research study. Grateful acknowledgement is also made to my colleagues at the Mechatronics Research Centre, De Montfort University, for their ideas and support. Lastly, I would like to thank my sponsors Universiti Teknologi Malaysia (UTM) and the Government of Malaysia.

ABSTRACT

The pace of technological change is increasing and sophisticated customer driven markets are forcing rapid machine evolution, increasing complexity and quality, and faster response. To survive and thrive in these markets, machine builders/suppliers require absolute customer and market orientation, focusing on rapid provision of solutions rather than products. Their production systems will need to accommodate unpredictable changes while maintaining financial and operational efficiency with short lead and delivery times.

Real-Virtuality (R-V) systems are an innovative environment to address these requirements by facilitating enhanced support in machine system design utilising integrated real-virtual environments centred on concurrent machine system development and realization. This environment supports not only machine system design but also the development of the control system at the same time. Utilising the Real-Virtual Mapping Environment (RVME), 3-D simulation machine models can perform actual machine operations in real-time when coupled with the real machine controller. This provides a more understandable, reliable and transparent machine function and performance.

The research study explores different types of controller verification methods and proposes a new method which employs the use of a control signal emulator. The research study has fomulated a novel technique for emulating quadrature encoder signals to provide virtual closed loop control of servomotors. The deployment of a control signal emulator technique makes the system unique and removes its dependency on specific hardware. Enabling the real-time data from the signal emulation environment eases the task of realising a real-time machine simulator.

To evaluate the proposed architecture, three case studies were performed. The results have shown that it is possible to create verified and validated machine control programs with no modification needed when applied to the real machine. The migration from the virtual to the real world is totally seamless. The result from the study show that the virtual machine is able to operate and respond as a real machine in real-time. This opens up the unexplored potential of integrated 3-D virtual technology. The real-time 3-D simulation virtual machine will enable commissioning and training to be conducted at an earlier stage in the design process (without having to wait for the real machine to be built). Furthermore, various test scenarios can also be developed and tested on the system which helps to provide a better understanding of the machine behaviours and responses.

This research study has made an original contribution in the field of machine system development. It has contributed a novel approach of using emulated control signals to provide machine control programmers with a platform to test their application programs at machine level which involves both discrete digital signals and continuous signals. The real-time virtual environment extends the application domain for the use of simulation. The architecture proposed is generic; to be exact it is not constrained to a specific industrial control system or to a specific simulation vendor.

Table of Contents

List of Figures

List of Tables

List of Acronyms and Abbreviations

2-D	Two Dimensional
3-D	Three Dimensional
API	Application Programming Interface
AS-I	Automated System Interconnect
CAN	Control Area Network
CASE	Computer Aided Software Engineering
CCE	Coca-Cola Enterprise
CD	Compact Disc
CEC	Commission for the European Community
CIM	Computer Integrated Manufacturing
CMS	Cellular Manufacturing System
CNC	Computer Numerical Control
DAC	Digital Analogue Converter
DCCS	Data Control and Communication Support
DCS	Distributed Control System
DCS	Distributed Control System
DDC	Direct Digital Control
DDE	Dynamic Data Exchange
DES	Discrete Event Simulation
DLL	Dynamic Link Library
DMU	De Montfort University
DOF	Degree of Freedom
DSO	Dynamic Shared Library
FAS	Flexible Assembly System
FBD	Function Block Diagram
FMS	Flexible Manufacturing System
GM	General Motors

GSL	Graphical Simulation Language
GUI	Graphical User Interface
HMI	Human Machine Interface
I/O	Input/Output
IEC	International Electrotechnical Commission
IL	Instruction List
IMAS	Intelligent Machines and Automation Systems
IMS	Intelligent Manufacturing System
ISO	International Standard Organisation/ International Organisation for Standardisation
JOP	Japanese Open Systems Promotion/Japanese OMAC (Open Modular Architecture Control)
LD	Ladder Diagram
LLTI	Low Level Telerobotics Interface
LVDT	Linear Variable Displacement Transducer
MCM	Model Communication Module
MRC	Mechatronics Research Centre
NASA	National Aviation Space Association
OAC	Open Architecture Control
OLE	Object Linking and Embedding
OMAC	Open Modular Architecture Control
OOP	Object Oriented Programming
OPC	OLE (Object Linking and Embedding) for Process Control
OSACA	Open System Architecture for Controls
OSI	Open Systems Interconnection
PC	Personal Computer
PID	Proportional-Integral-Derivative
PLC	Programmable Logic Controller
RRS	Realistic Robot Simulation
RTEE	Real-Time Emulation Environment

R-V	Real-Virtuality
RVME	Real-Virtual Mapping Environment
SCADA	Supervisory Control and Data Acquisition
SCM	Signal Control and Management
SFC	Sequential Flow Chart
ST	Structured Text
TCP/IP	Transmission Control Protocol/Internet Protocol
UDP	User Datagram Protocol
VIM	Visual Interactive Modelling
VIR-ENG	Integrated Design, Simulation and Distributed Control for Agile Modular Manufacturing Machinery
VM	Virtual Manufacturing
VME	Virtual Machine Environment
VR	Virtual Reality
ZEM	Zanussi Electro-Mechnica

List of Symbols

θ	angular displacement
ω	angular velocity
ω_{max}	maximum angular velocity
α	angular acceleration
t	time
t_{max}	maximum time
t_{acc}	time taken to accelerates
t_{dec}	time taken to decelerates
t_{total}	total time
Cd	continuous data
Co	continuous operation
Cp	continuous program
Cs	continuous signal
Dd	discrete data
Do	discrete operation
Dp	discrete program
Ds	discrete signal
Kp	proportional gain
Ki	integral gain
Kd	derivative gain

CHAPTER 1

INTRODUCTION

Design, simulation and control of manufacturing machinery and related systems can be seen as a collection of technologies, which if integrated properly can provide highly convenient interfaces between humans, computers and machines. Attempts have been made such that the activities involved were more intuitive to use (Stone 1998). The term intuitive reflects an ambition of Real-Virtuality. The term Real-Virtuality (R-V) at first sight, might appear a contradiction of terms and hence needs some explanation. Traditionally, "real" suggests an artefact or physical presence of an item e.g. machinery, components, etc while "virtual" usually refers to computer software model or synthetic environment which is capable of mimicking the behaviour of a chosen system. However, certain "virtual" models can provide almost perfect behaviour patterns of chosen machinery and hence provide users with what is referred to as a transparent interface, making navigation through, and interaction with virtual environment as real as possible. Here, "real" will refer to actual artefacts, while "R-V" will be used for a combination of real machinery and software models which enable complex procedures to be done with high levels of confidence. Thus, progress in the fields of information technology such as computer networking and 3-D graphics is now enabling industrial companies to change their approach from mainly hardware based solutions to information-oriented ones (Iwata et al. 1997).

This research involves the investigation on how 3-D virtual environments can be used for the integration of design, simulation and controller program development of a typical manufacturing system. The research program has evolved around the adoption of 3-D simulation models being used as virtual machines and being controlled by a real controller. The controller can be programmed, tested, verified and validated in real time and the action of the machine is realised in the 3-D model. The R-V system opens up a

new horizon for system installation and commissioning, customer support, user training and system re-configuration. The following sections provide the research background, the hypothesis being tested, followed by the scope and the objectives of the project.

1.1 RESEARCH BACKGROUND

Development utilising affordable computer technology with increase of processing power, provision of high resolution graphics, the simplicity of object-oriented-programming (OOP) languages, and easy access to software tools now permits the realisation of new interactive graphical visualisation environments which could revolutionise research, design & analysis, and education in the field of control systems. Advancement in 3-D modelling allows designed machines or systems being viewed in highly realistic ways. Thus, detailed 3-D models coupled with matching machine system realism, helps in presenting designs to potential customers even before prototypes are available. Simulation on the other hand has been identified as a tool in imitating the operation of a machine or system over time. Simulation techniques together with some 3-D visualisation can provide tools for the verification of specific designs (Min et al. 2002). The designed machine or system behaviour is observed and studied as it evolves over time. Simulated models help to get the feel of the real system early in the design stage before the system is built.

There are considerable changes in manufacturing businesses of all sizes as they adapt to operating in a dynamic global market, with customer driven manufacture and service provision (Weston 1998). The problems faced by the different machinery sectors are almost identical. This is not surprising as the dominant influences are globalisation of markets, effects of legislation, rapid pace of technological change, reduced product life-spans, and the need for rapid response to customer demands.

The trend in advanced manufacturing and production equipment design is towards modularity. Different clients within the same sector of manufacturing activity frequently

have varying initial needs. In order to be cost effective these can only be met by designing modular machines which can be individually configured to meet the specific needs of each client. The rapid evolution in markets with corresponding changes in the products to be manufactured, require that the manufacturing equipment can be rapidly and reliably re-configured within the manufacturing plant in an efficient manner. Hence, machines should be able to easily adapt to new functions or upgrade existing functions using new parameters. Companies are adopting agile manufacturing methods to meet with these challenges (Maskell 2001). This requires that in general, manufacturing machines should have an open and a modular structure.

Although the constraints of modularity can be taken into account by the machine designer/builder in the initial mechanical and electrical design of his equipment, this modularity can cause considerable problems for manufacturer in the continuing development or the re-configuration of the advanced control systems for their machines. It is frequently difficult to guarantee in advance the performance of a machine in a new configuration without first building the machine in the new version and determining the constraints that the control system might impose.

Although the performance of individual mechanical sub-assemblies may be pre-determined with accuracy, this may not be possible for a complex control system where the management of many different but associated functions could limit the overall throughput of the machine with a new configuration. In general, each time a new configuration of a machine is built for the first time; considerable time is spent in modifying and optimising the control system. Furthermore, this work can only be validated on the machine itself so adding to the critical path either for initial machine delivery/commissioning or for in-field modifications.

There is also ever increasing pressure on machine builders to design equipment which will meet the specific requirements of each client, while being prepared for the possibility of adding new features/functions to the machine during its lifetime to meet the changing production needs and facilitating immediate service support to customers to minimise down time on production lines. This leads to the need for design of flexible

modular equipment which can be built in a wide range of different configurations and which allows rapid in-field re-configuration to meet changing production requirements and frequent product changeover. The problems are similar for most high tech equipment builders across many industrial sectors – designing a new generation of machines with enhanced modularity and flexibility whilst allowing faster, more sophisticated and more cost-effective service support.

A key challenge is to develop advanced control systems with proven operation in the most timely and cost effective manner, which can then be commissioned on the customer site. The difficulty is that in general, the target machine systems to be controlled are not available early enough to enable a full verification test of the control system being developed as the manufacturer can take several months for the installation and smooth operation a new production system. The control system plays a very significant role in minimising the down time in the use phase of the production line or the machine. Another important business factor may be the ability of the supplier to offer maintenance and service support including controller up-grades. As global markets continue to develop, machine builders need to develop strategies to support their equipment and processes at all their clients work sites throughout the world in a cost effective and technically efficient manner. Sending out specialist staff on emergency breakdown services is not a preferred solution due to the high travel and labour costs, disruption of work and customer dissatisfaction with the unavoidable delays.

1.2 SCOPE OF RESEARCH

The overall aim of the proposed research is to enable machine builders and control system developers to gain the ability to support and develop their advanced control systems from their home base during different life cycle phases of the system in a highly effective way. As such the research hypothesis to be tested is "*Real-Virtuality systems can significantly enhance the development and support processes of manufacturing machine systems and their associated control systems*". The scope of

this research is to formulate an integrated Real-Virtuality (R-V) system for system developers and manufacturers, which can offer them the facility to:

i. integrate the real-time control system with simulation models of the target machine for application code development

ii. simulate in real-time the behaviour of the target machine including the operation of sensors, actuators and other I/O actions

iii. address stringent and complex timing requirements and interactions in the area of machine control.

1.3 OBJECTIVES

The following objectives have been identified to support the research program outlined.

1.3.1 To Establish New Concept For Bridging Virtual And Real World

This objective is to devise a new concept in minimising the gap between the virtual world with the real world in machine building and control system development, leading to a combination of real and virtual systems where the transition between the two is seamless.

1.3.2 To Improve Advanced Control Systems Development, Validation and Deployment.

This objective is to improve competitiveness by providing a faster proven design and lower cost developments, enhanced modularity, easier re-configuration and easier

remote service/maintenance

1.3.3 To Enhance Digital Factory Software Packages Facilitating 'Real-Virtuality' Modelling /Simulation

This objective is to provide an enhanced scope and wider application base for digital factory tools (extended to control engineering). This is facilitated by the use of the virtual machines which can be executed in the virtual time environment, with open interfaces, which would allow its integration with the real-time emulation environment. Simulation will then truly represent the reality.

1.3.4 To Develop A Readily Verified And Tested System

This objective is to develop a real-time emulation environment which will enable the control system application tasks to be developed, tested, and verified, without access to the target machine system or having it built physically. Once the control system is fully tested it can easily be detached from the virtual world and ready for use in the real system.

1.3.5 To Establish A 'Real Virtuality' Simulator

The objective is to develop a prototype 'real-virtuality' system simulator, which can be used to facilitate the life cycle support of advanced control system development and use-phase–operation, commissioning and training services of the complete machine system.

1.4 THESIS ORGANISATION

The thesis is organised within seven further chapters as follows:

Chapter 2 – Technology Related To Manufacturing System, Controller And Simulation: outlines the key elements of manufacturing machinery and simulation. This includes overview and historical background of technologies within industrial controllers and simulation, and some current manufacturing simulation applications that motivate this research.

Chapter 3 – Related Work And Motivation: provides in depth review on related research work in the area of industrial controller application within the virtual environment. The chapter gives a frame of reference by reviewing relevant literature.

Chapter 4 – A Reference Architecture For A Real-Virtuality System: defines an architecture for the proposed Real-Virtuality (R-V) system.

Chapter 5 – A Real-Virtuality Prototype: describes the development of a prototype system based on the proposed architecture. This chapter discusses in detail the implementation of each and every elements of the R-V system.

Chapter 6 – The Real-Virtuality (R-V) System Configuration And Implementation: explains how to setup and use the developed R-V system for machine design and development.

Chapter 7 – Real Virtuality Case Studies, Results And Discussions: presents test cases on the R-V system. This is followed by evaluation and discussion of the results from the studies.

Chapter 8 – Conclusion: presents the conclusion from the research, contributions that have been made and suggests directions for the future work.

CHAPTER 2

TECHNOLOGY RELATED TO MANUFACTURING SYSTEMS, CONTROLLERS AND SIMULATION

2.1 INTRODUCTION

A manufacturing system often involves significant amount of investment and its social and economic impacts are enormous. At a basic level, manufacturing is about a group machines working together to perform a certain task. Therefore, the overall machine system design is bound to play a critical role in the business perspective and thus draw increasing attention from research community and business enterprise.

Design is a process of interplay between the goal and the means (Suh 1990). The goal is what to be achieved and the means is how the goal is to be achieved. Throughout the entire design process, one of the most difficult tasks is to obtain a concrete design, which clearly maps the way between what and how. It is very important to have a correct design formulated at an early stage because 60% of the design including the cost has been commited at this stage (Nevins and Whitney 1989).

A true manufacturing machine is a system which involves not only hardware facilities and working procedures but also people with various skills. Thus in practice a manufacturing system is likely to be fairly complex. A typical manufacturing system is a dynamic organization where business and manufacturing process are changing to meet market demand and people in the system provide a flexible response to the needs. However, it is often found that the design of a manufacturing machine is actually more costly or more complex than required.

Simulation models in machine design are increasingly being used in decision making

and solving problems. Simulation model helps the developer in making decisions using information derived from the results of the model simulation. In general, simulation is regarded as the imitation of real world process or system operation. It is used to describe and analyze the behaviour of a system, ask what-if questions about the real system, and aid in the design of real systems [Banks 1998, 1999]. The use of simulation applications include the modelling and verification of discrete and continuous manufacturing processes, robots and other machinery offline programming, layout planning, process and system visualisation, ergonomic analysis of manual tasks and work area layout, evaluation of scheduling algorithms and dispatching rules, and business process engineering [McLean and Leong 2001].

This chapter will give a review on the developments of manufacturing system, controller and simulation and their entities.

2.2 ELEMENTS OF PHYSICAL MANUFACTURING SYSTEM

Manufacturing is defined as a series of interrelated activities and operations involving the design, materials selection, planning, manufacturing production, quality assurance, management and marketing of the products of the manufacturing industries (Hitomi 1996). It is also common to characterize manufacturing as an input-output system which produces outputs (economic goods) through activities of transformation of inputs (factors of production) (Wu 1994).

Banks et al. (2000) define a *system* as a group of objects that are joined together in some regular interaction or interdependence toward the accomplishment of some purpose. The American Heritage (2000) defines a system as a group of interacting, interrelated, or interdependent elements forming a complex whole. Prasad (1996) suggests that for a system to be functional, the followings should be true:

i. *Parts of a Whole Function*: A system consists of parts which each must provide or contain a set of partial but distinct function by itself i.e. functions

that can be distinctly identified as associated each part of the whole.

ii. *Whole of the Parts*: In order to accomplish the function (aim or purpose) of the whole system, a part or a combination of parts should work together. When every part is in place, each part should capable of contributing a functional build-up leading to the system's functionality.

iii. *Structure or Constancy of Purpose*: Parts need to be arranged to work as a unit in a definite order (in some instance priority). The system requires a structure or a "constancy-of-purpose" to provide the system functionality.

After a lengthy discussion on the term *system* and *manufacturing* in his thesis, Holts (2001) comes out with a definition of *manufacturing system* which he describes as:

> *"A complex whole formed by a group of interacting, interrelated, and interdependent elements with the purpose of executing all the activities and operations needed to put a product on the market*".

Physical manufacturing system is a collection of integrated equipment designed for some special mission. In describing a build up of a physical manufacturing system, VIR-ENG[1] (Pu and Moore 1998, and VIR-ENG 2001) comes out with a manufacturing machine system model which divides physical manufacturing system into four level of abstraction as shown in Figure 2.1. VIR-ENG highlights that a manufacturing system consists of various elements (parts) that are put together and arranged to work as a unit. Individual elements have their own functions but also compliment one another to establish the system operations (functionality).

[1] VIR-ENG is an EC Framework IV project to address the need of a suite of tools within an integrated environment for the design and development of machines systems.

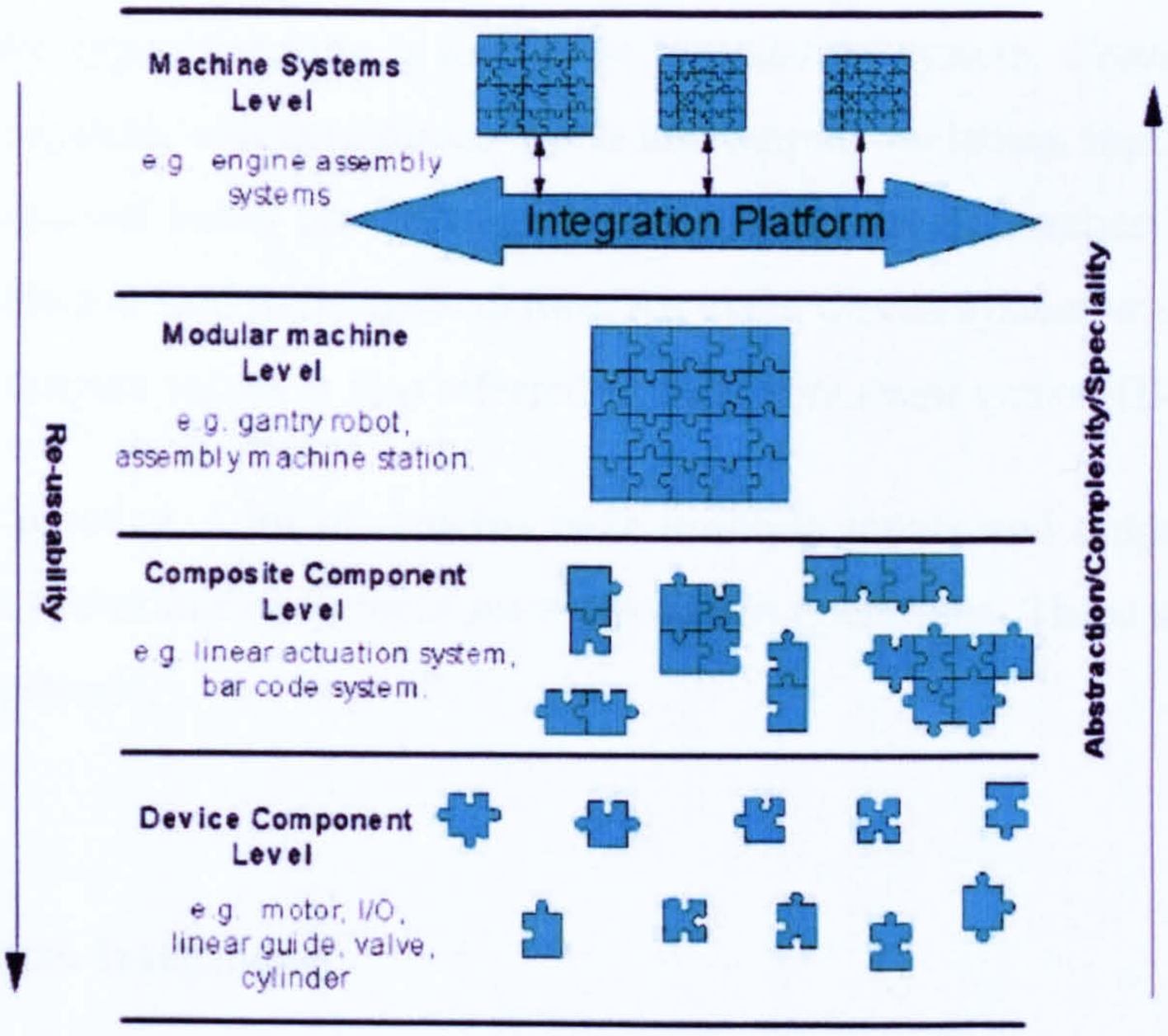


Figure 2.1 The VIR-ENG manufacturing machine system model

2.2.1 System Classifications

Physical manufacturing system which is made of machine(s) has operations which are always described of either *discrete* or *continuous*[2]. The classification is based on the state variables involved in the operation. In general, the discrete operation is described as an operation where the input and output variables may only have a finite number of different values. On the other hand, a continuous operation is described as an operation where the variables may take continuous values over some range.

[2] There are many ways a system is classified. Depends which level of solution tried to be made on the system, it is normally classified either to be statics - dynamics, deterministic - stochastic, linear - non-linear or continuous - discrete. *See* (Banks, Carson, Nelson & Nicol 2000, Law & Kelton 1991). In context of machine builder and system design applications a system is normally classified as continuous – discrete based on the system operations.

Input and output variables of a system operation are sometimes monitored over periods of time. This type of system is known as *time driven* system. *Continuous system* is a time driven system with continuous inputs and outputs variables. Inputs and outputs can also be monitored based on the value change due to the occurrence of events. In such case the system is said to be *event driven*. An event driven system with finite number of inputs and outputs values is also referred as a *discrete event system* (Banks et al 2000).

In modern practice, a lot of systems have multiple inputs and outputs with some are time driven operations and others are event driven operations. These systems are known as *hybrid systems*[3].

2.2.2 System Integration

Increasing market demands and complex structure of manufacturing systems has necessitated computer assisted systems to be the main tools of manufacturing system automation (Gelgele 2002). At present there are many different types of automated manufacturing systems. Some of the more common systems are the flexible manufacturing system (FMS), flexible assembly system (FAS), cellular manufacturing system (CMS), and computer integrated manufacturing system (CIM), which all have different scopes (Yien and Tseng 1997). Although some of the systems have different names they are similar in function. An FMS, for example, covers both fabrication and assembly, whereas FAS only covers assembly. In other cases, different names refer to similar types of systems, such as the CMS and CIM, although they emphasize different aspects. Some of these systems cannot in themselves be considered as manufacturing systems if the term is understood in the broader sense. Such systems, including the FMS, FAS, and CMS, only focus on the material flow, i.e. production and logistics aspects (Yien and Tseng 1997). On the other hand, a computer integrated manufacturing (CIM) system integrates all the activities and system components necessary to comply

[3] Law and Kelton (1991) indicate that in practice, very few systems are wholly discrete or continuous. A system is normally classified as discrete or continuous since one type of change predominates for most systems.

with the manufacturing system definition.

Integrated manufacturing uses computers to connect physically separated machine processes. When integrated, the processes can share information and initiate actions. This allows decisions to be made faster and with fewer errors. Design of integrated manufacturing system equipped with automated machines has been deeply influenced by the evolution of microelectronics, control and computer science. An automated system is indeed a mechatronics system which is the integration between mechanical systems, electronic systems, control systems and computing. Current trends in electronics, computer science and computer control are providing the technical capability technology to greatly facilitate the development of integrated industrial control system. These trends include distributed, digital, microcomputer based control system; real-time programming languages; and high speed data links.

An integrated system requires that there be two or more controllers connected to pass information. A simple example is a robot controller and a programmable logic controller working together in a single machine. A complex example is an entire manufacturing plant with hundreds of workstations connected to a central database. Integration means putting together heterogeneous components to form a synergistic whole (Vernadat 1996). It is apparent that integration becomes essential and important element of a system at higher manufacturing machine system level (refer VIR-ENG model, Figure 2.1)

However, in the view of modern machine builders and automated system designer, an automated system consists of two main elements, the system and the system controller. The system is the machine(s) that consist of the entire mechanical and electronics component whilst the controller is the system's "brain" which does all the control, logics and communication. Levels of integration decide the complexity and as such, integration may involve only a single machine or several machines and the control may be affected by a single controller or multiple controllers.

System operations consist of either discrete operations (Do), continuous operations (Co)

or blend of both (hybrid). Meanwhile the controller which controls the associate system operations with will have the necessary discrete program (Dp) and/or continuous program (Cp). Interaction between the system and the controller is achieved through electrically generated discrete signals (Ds) and/or continuous signal (Cs) transmitted via physical wires. Figure 2.2 illustrates the common arrangement of the physical manufacturing system elements.

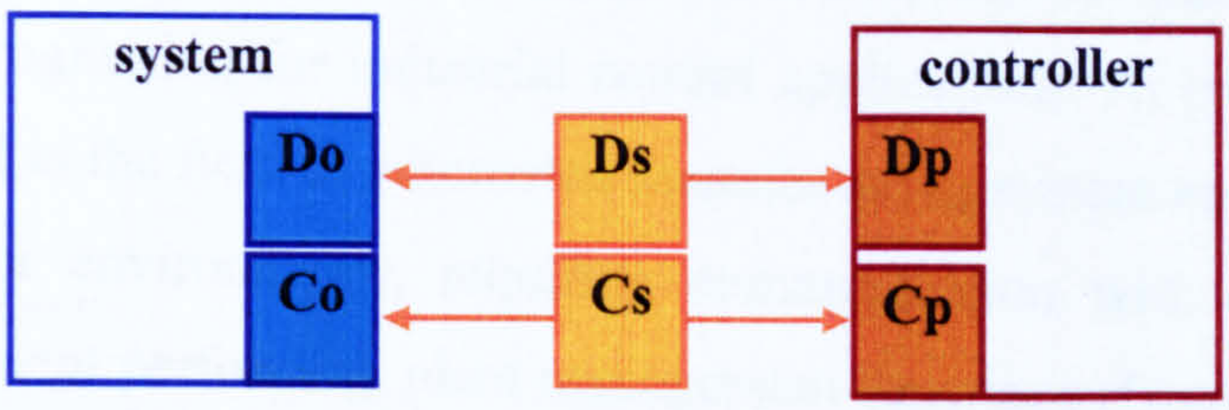


Figure 2.2 Elements of physical manufacturing system

2.3 INDUSTRIAL CONTROL SYSTEM

Control systems for highly complex systems (such as processing plants, manufacturing processes, aerospace vehicles, power plants, etc.) are themselves very complex (Wills et al. 2000). Notions of "control" are expanding from the traditional loop-control concept to include such other functionalities as supervision, coordination and planning, situation awareness, diagnostics and optimization (Bonissone et. al. 1995 and Vachtsevanos 1998).

History shows that computer has been in used for control in industries since late 1950s or early 1960s. Among early examples of computer system in industries are at Texaco refinery in Port Arthur (1956), Texas and Louisiana Power and Light Company, a power station in Sterling (1958), Louisiana (Bennett 1994 and Groover 2001). Early use

of computer is more on control of industrial processes which demand less speed in signal processing. The use of computer in product manufacturing which involves discrete operation only became popular in early 1970s with the development of microcomputer. Microprocessor based *programmable logic controller* (PLC) is used to replace relay banks to satisfy the early requirement of manufacturing automation (Groover 2001). In the late 1980's, General Motors and other manufacturers began using standard personal computers (PCs) for machine control. These new systems not only ran on off-the-shelf PC hardware, but also introduced advances such as flow chart programming programing for industrial control applications. At present, PLCs are still extensively used in the field of automated manufacturing system and they are integrated into much larger environments, requiring communication with other controllers or computer equipment performing plant management functions (Lewis 1998 and Bonfatti et al. 1999).

2.3.1 Control Architecture

The application of controls in manufacturing is expanding at a very rapid pace due to advances in computer technology, computer science, data communication and software engineering. Manufacturing system control is concerned with managing and controlling the physical operations in the machines(s) to implement orderliness. Industrial control system is an important component in realising industrial automation. In industry or manufacturing, control is used on each and several different level. Kamen (1999) divides industrial control application into four different levels (Figure 2.3):

i. Enterprise Level: level where high–level corporate information are carried out involving finances, administrative matters, master production scheduling, customer relations, etc..

ii. Factory Floor Level: level which contains the complete production facility which includes scheduling, routing, inventory control, material

requirement planning, quality control etc.

iii. Cell/Line Level: level that contains groups of machines or workstation connected and supported by material handling system, computer and other equipment necessary to the manufacturing process.

iv. Machine/Process Level: level which contains the individual machine or equipment items needed to manufacture products such as industrial robots, automated machines, powered conveyors, automated guided vehicles etc.

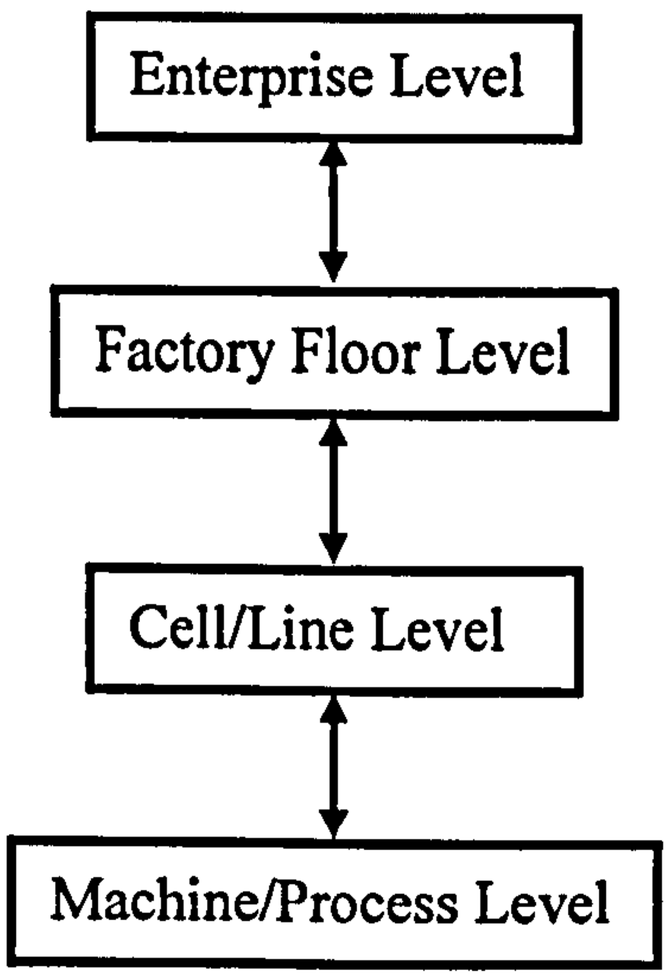


Figure 2.3 Different level of industrial control (Kamen 1999)

Groover (2001) however extends the above industrial control application level with one additional level named "Device Level". This level consists of devices such as actuators, sensors, and other hardware which comprise the machine level. Example of control application at this level is control of one joint of an industrial robot which is built from combination of devices. The Groover five levels of control in manufacturing are shown

in Figure 2.4 below.

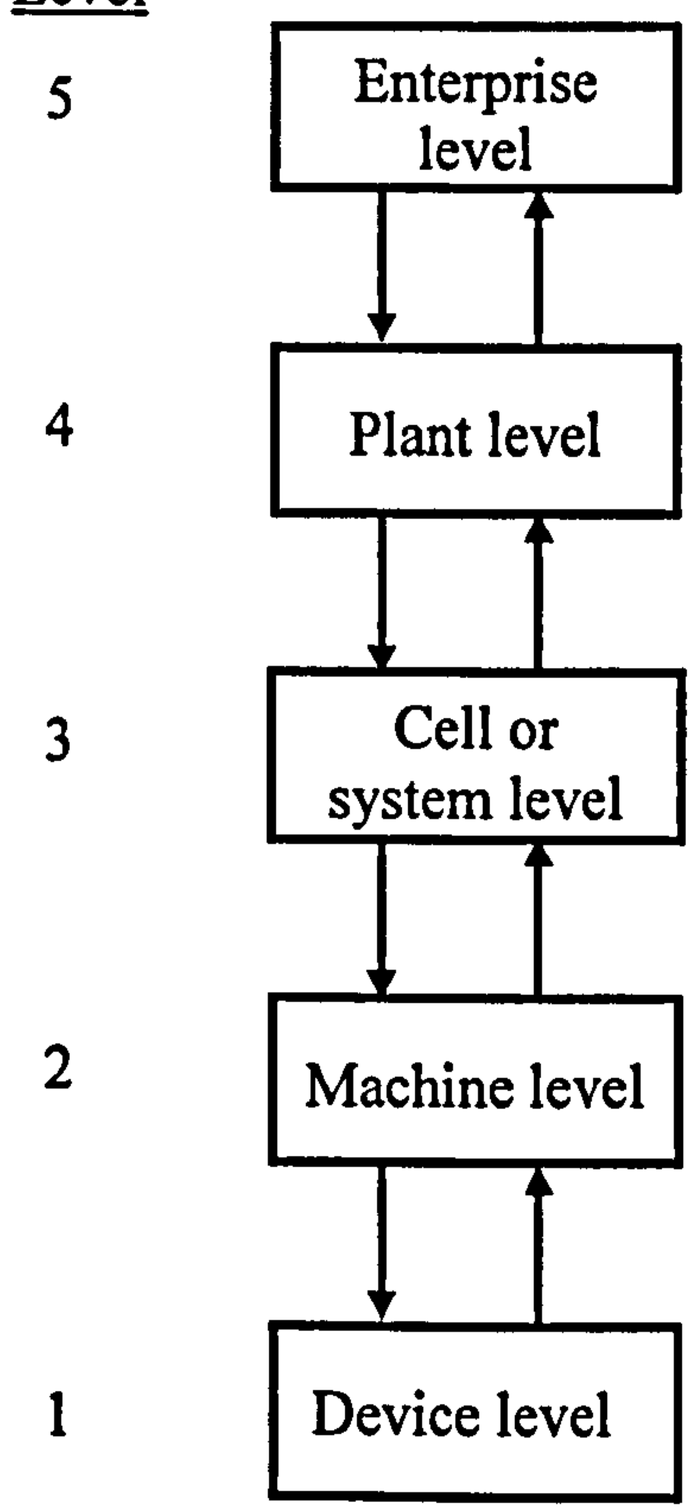


Figure 2.4 Groover (2001) five levels of control in manufacturing

The control application levels described by Kamen and Groover are the most common industrial control architecture, also known as *hierarchical* control architecture. The work in this research is focussed on the middle levels of Groover hierarchical architecture where the control system involves is the control of physical manufacturing system. The Level 2 and Level 3 of Groover five level of control in manufacturing are concerned with controlling individual machine or group(s) of machines. Although Level 1 also is part of physical element of manufacturing system, this research does not directly deal with control application at this level. Control at this level normally is

known as *low level control*[4].

Although hierarchical architecture presented by Kamen and Groover is quite common in illustrating industrial or manufacturing control architecture, a lot of research has been carried out in defining industrial control architecture. Some studies classify the architecture into four different categories as shown in Figure 2.5 (Dilts et al. 1991, Klingstam 1998, Chong 2002a and Danielsson 2002):

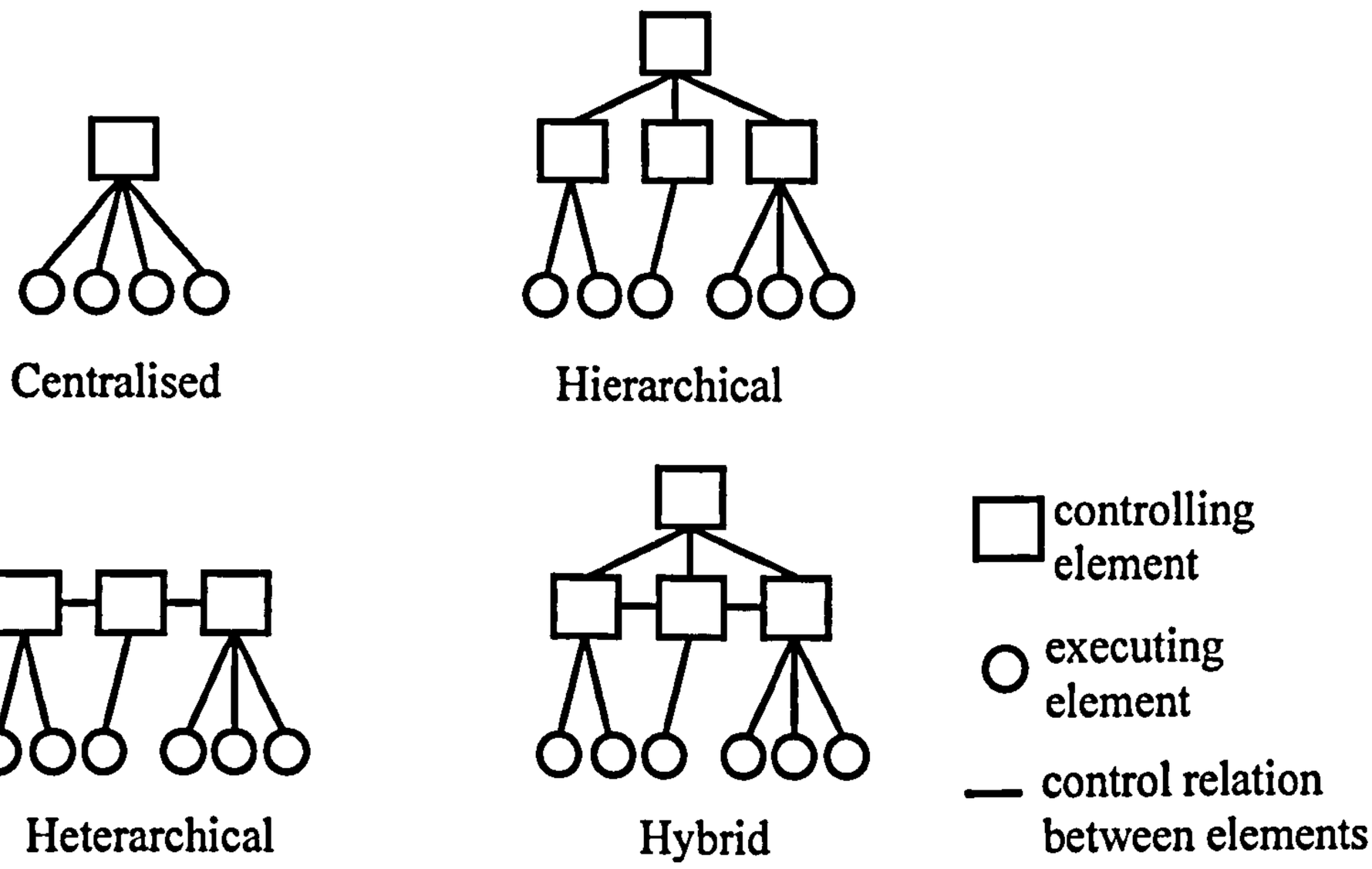


Figure 2.5 Control architecture for automated manufacturing (Dilts et al. 1991)

i. *Centralised*: built around a central control module, which communicates directly with the surrounding equipment.

ii. *Hierarchical*: divided into several levels, where different levels serve

[4] *Low-level* control is also sometime known as d*evice level* control. Research in low level control normally involves understanding the dynamic behaviour of a system and formulating control strategies to improve system performance which is beyond the scope of this thesis.

different functions. Higher levels are used for planning and organisation and lower levels for the control and execution of functions.

iii. *Heterarchical*: functionality is distributed among industrial control systems working on the same level.

iv. *Hybrid*: a mix between hierarchical and heterarchical.

2.3.2 Elements of Machine Controller

Machines are controlled by devices called controllers. Modern industrial controllers which are used in industries basically are microprocessor based equipment which requires program instructions, make the control calculation and execute the instructions by transmitting the proper commands to actuating devices.

Parker Haniffin Corporation in their Engineering Reference print (Parker Motion & Control 1995) has identified that a complete machine control has the primary elements as shown in Figure 2.6which comprises of:

i. Motion Control: For precise programmable load movement using a servo motor, stepper motor, or hydraulic actuators. Feedback elements are often employed.

ii. Analogue and Digital I/O: For actuation of an external process, devices such as solenoids, cutters, heaters, valves, etc.

iii. Operator Interface: For flexible interaction with the machine process for both setup and on-line variations. Touch screens, data pads, and thumbwheels are examples.

iv. Communications Support: For process monitoring, diagnostics and data transfer with peripheral systems.

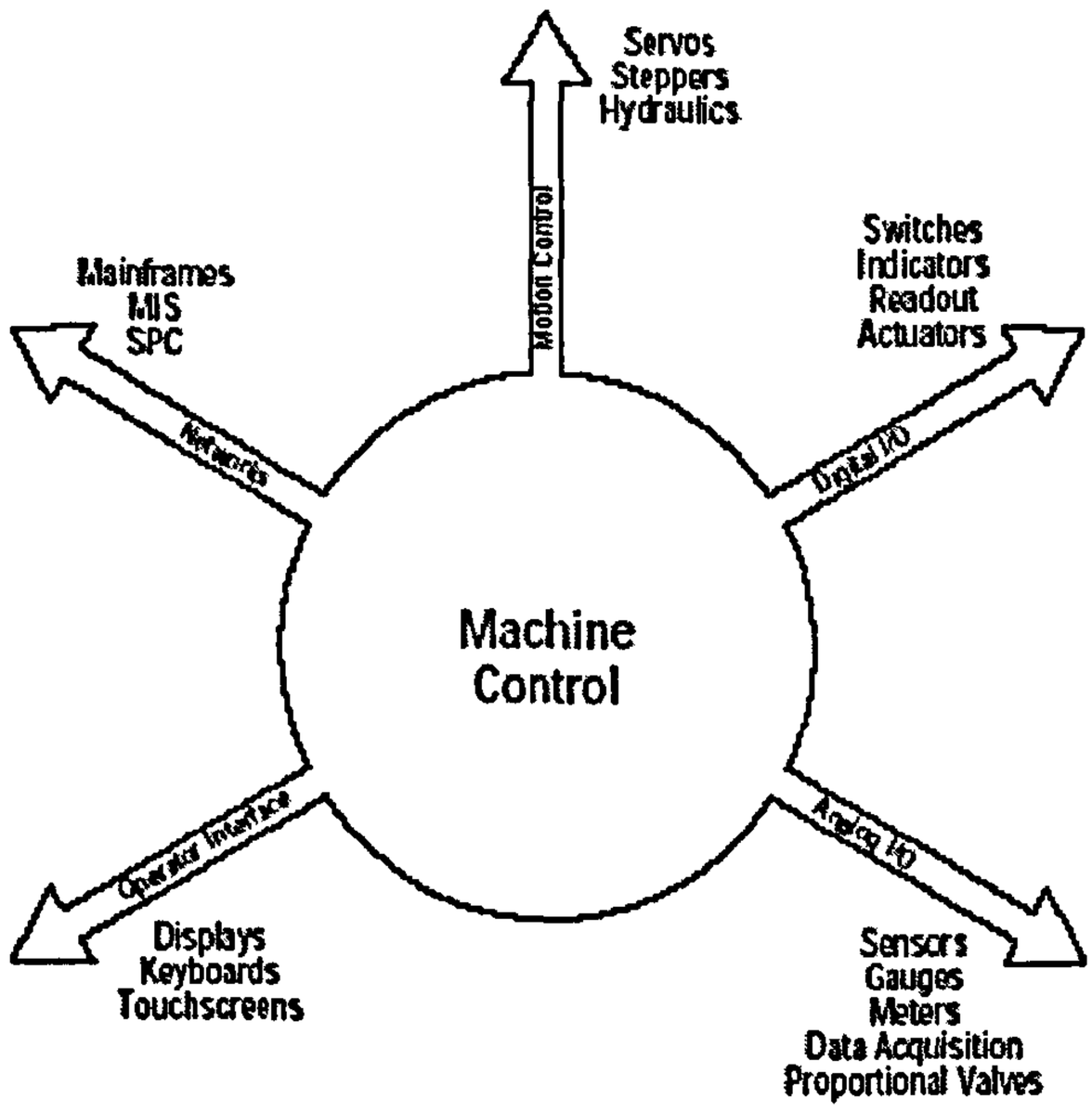


Figure 2.6 Primary machine control elements (Parker Motion & Control 1995)

2.3.2.1 Discrete Control

Discrete control[5] is a most prominent type of control in manufacturing system. In discrete control, the operational parameters and variables in the system are changed at discrete moments in time. The variables and parameters involved in the change are also discrete, typically of binary type (ON or OFF). The changes are defined in advance by means of an instructional program. The changes are either *event driven changes* or *time driven changes* (Bateson 1999).The two cases can be distinguished by detecting either the state of the system change or the amount of time elapsed.

[5] *Discrete control* is also sometime known as *discrete logical control* (Kamen 1999). This is to distinguish between the use of discrete control in logical operation and in digital continuous control (which data are sampled periodically or discretely). However, the earlier denotation will be adopted throughout the thesis.

2.3.2.2 Continuous Analogue and Digital Control System

Continuous processes require continuous sensors and/or actuators. In continuous control, the usual objective is to maintain the value of an output variable at a desired level (setpoint). Process control and servo control are two types of continuous control. However, in manufacturing system, servo control is more prominent and vital because a lot of servomechanisms are being used and the sampling rate is very fast (milliseconds). Early servo control makes use of analogue sensory devices such as potentiometers and tachometers. Signals are converted to digital form via analogue-to-digital converter to perform what is known as *analogue servo control*, which is quite similar to process control. Since the introduction of digital encoders in 1960s a mixture of analogue and digital servo control and completely digital servo loop are being used in industries. An example of mixed analogue and digital servo control loops is shown in Figure 2.7 below (Olsson and Piani 1992).

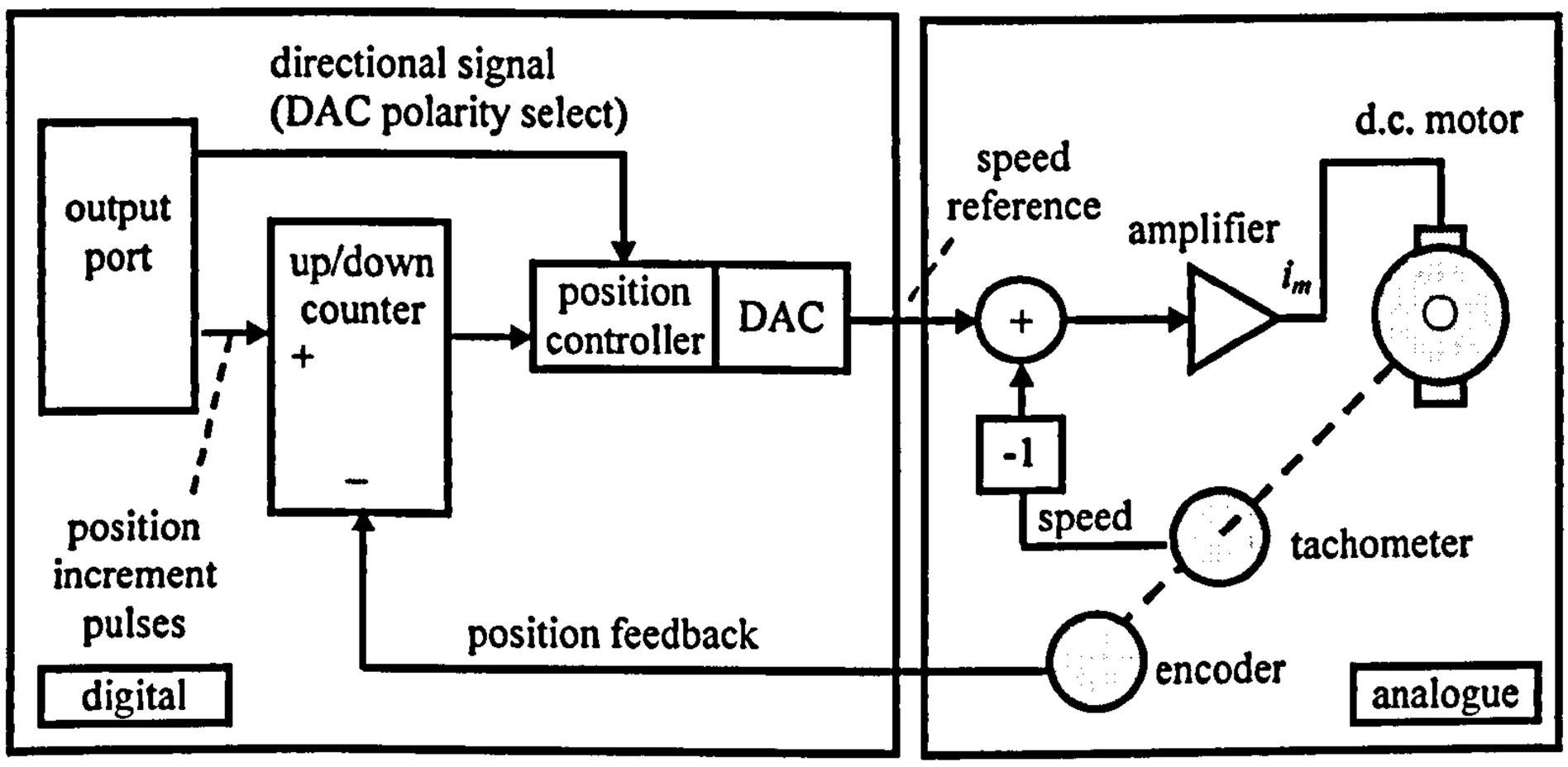


Figure 2.7 Mixed analogue and digital servo control loops (Olsson and Piani 1992)

For servo axis movement in modern automated machine applications, the setpoint needs to be generated automatically. Modern industrial controller is equipped with *motion*

control system which is used to generate setpoints over time. An example of a motion control system is shown in Figure 2.8. The motion controller accepts commands or other inputs to generate a motion profile using parameters such as distance to move, maximum acceleration and maximum velocity. The motion profile is then used to generate a set of setpoints, and times they should be output. The setpoint scheduler will then use a real time clock to output these setpoints to the motor drive (Jack 2004).

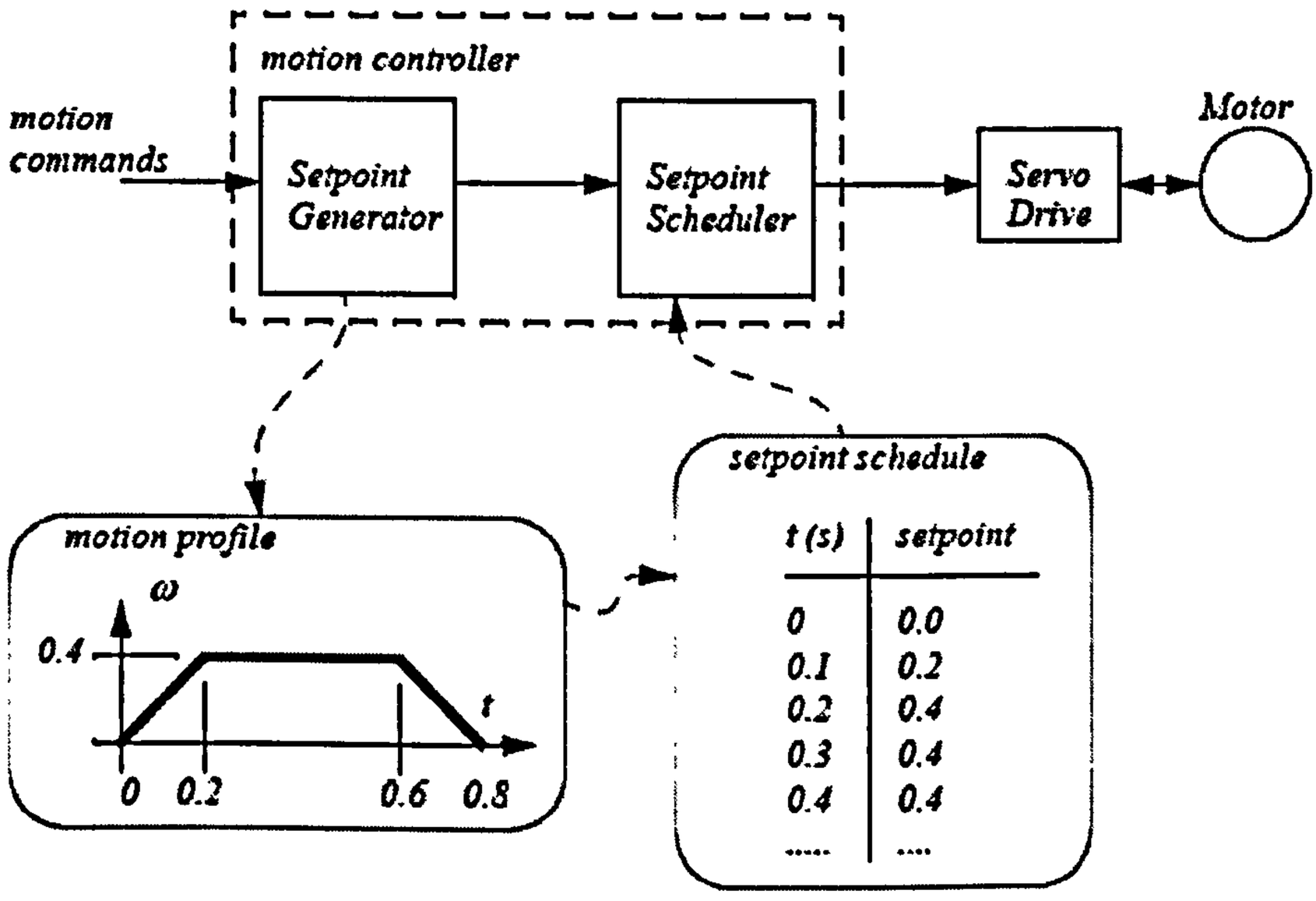


Figure 2.8 Motion controller (Jack 2004)

2.3.2.3 Industrial Controller Technology Review

PLC and PC-based Controller

When microprocessors found their way into industrial controllers, the term Programmable Logic Controller (PLC) was born. First developed during the early 1970s, PLCs started as simple devices to replace electro-mechanical relays. Utilising the integrated circuit technology, PLCs performed simple sequential control tasks, in isolation from other control and monitoring equipment. PLCs have grown from simple

discrete control devices into complex systems capable of almost any type of control application, including motion control, data manipulation, and advanced computing functions. Like many technologies in automation, it continues to shrink in size, increase in function, communicate more, and integrate well with other forms of industrial computers. In order to survive, PLCs has to adapt themselves with the battle with PC-based control (Mintchell 1998).

At present, PLCs are extensively used in the field of automated system and they are integrated into much larger environments, requiring communication with other controllers or computer equipment performing plant management functions (Lewis, 1998 and Bonfatti et al., 1999). PLCs are still the logical choice for many control applications. The main reason for the success of PLC applications were its ability to reduce cost, ease of configuration and low maintenance (Chouinard 2000).

PC-based control, sometimes known as *soft*control or *soft*logic has been surrounding the industry's circle for more than 15 years has now becomes more popular. PC enthusiast claim PC-based systems are easier to install and use, provide superior diagnostics, and offer more flexible choices for systems integrators. With massive memories, open networking connections, and blazing processing speeds, PCs are proving a good alternative to traditional programmable controllers. The most successful story for PC-based control is at General Motors' Powertrain plants in the Detroit. There, over three thousand PCs in a dozen factories go to work every day machining and assembling engines and transmissions.

PC-based control is now increasingly recognised as an open and powerful hardware platform, which can provide effective and reliable control, with no requirement for additional processors or complex hardware additions (Hayes 2000). PC enthusiast claim PC based systems are easier to install and use, provide superior diagnostics, and offer more flexible choices for systems integrators. With massive memories, open networking connections, and blazing processing speeds, PCs are proving a good alternative to traditional programmable controllers. There are significant advantages in the use of PC-based over traditional PLC systems. Not only in term of performance, but also the

availability of open systems with industry standards. PC-based control easily integrates with other tasks, including data logging and manufacturing execution systems. It also runs on non-proprietary hardware (McKay 2002). Among the advantages from adopting PC-based system in comparison to PLC is the ability to run human machine interface (HMI), control, supervisory control and data acquisition (SCADA) and other applications on a single box. This results in savings not only in buying fewer pieces of equipment, but also in reduced panel space and the reduced integration effort needed to make the system work (Nemarton 2004).

Machine Control Languages

In the beginning, most programs for PLCs are written in *Ladder Logic*, a software representation of physical relays. However, each PLC company's software was so uniquely its own, that it was virtually unintelligible to anyone else. Machine users felt increasingly trapped into always having to use the same control equipment vendor. The different makes of controller could not interface with one another; or in industrial term they are said as not *interoperable* (Dobson 2004). Many problems have also accumulated through incorporating PLCs into control systems, especially in the software growth and testing area (Hassapis 2000). In December 1993, International Electrotechnical Commission (IEC) started an initiative aimed at improving the standardisation and efficiency of industrial controllers, including hardware, software and usage (IEC 1993). The standard which is called IEC 61131, consist of multiple sections: Part 1 General Overview, Part 2 Hardware, Part 3 Programming Languages, Part 4 User Guidelines, Part 5 Communications, Part 6 Fieldbus Interconnectivity, Part 7 Fuzzy Control Programming and IEC Part 8 Application and Implementation Guidelines.

Part 3 of the standard which is popularly known as IEC 61131-3 standard, covers the global standard for industrial control programming. A standard programming interface allows people with different backgrounds and skills to create different elements of a

program during different stages of the software lifecycle: specification, design, implementation, testing, installation and maintenance.

Within the standard, five programming languages are defined (Mintchell 1999, Dobson 2004). The languages are as follows:

i. *Ladder Diagram (LD)*: A graphical depiction of the switches in a control system which allows easy visualisation of the system's operation. It is popular with shop floor engineers and technicians and also with traditional engineers who were brought up with relay-based control technologies. IEC Ladder also encompasses user-defined function blocks, so can be used in a hierarchical control system.

ii. *Complex Algorithm or Structured Text (ST)*: Very powerful language with its roots in Ada, Pascal and "C". It can be used excellently for the definition of complex function blocks, which can be used within any of the other languages.

iii. *Instruction Set or Instruction List (IL)*: This relatively low level language is used as a program assembler, connecting different parts of the program together.

iv. *Sequential Function Chart (SFC)*: Specified as means to describe the sequential behaviour of a control program, as an aid to structure the internal organisation of a program.

v. *Function Block Diagram (FBD)*: Another graphical language used for charting signal and data flows. The blocks are reusable mini-programs

There is one intended benefit expected of the whole acceptance of IEC 61131; the portability of code from one application to another. In theory, once an application has been developed using a particular manufacturers system, the code should be able to be

ported to another manufacturers system with ease. The standards have been written with enough definition to allow this to be completely possible (Powersoft 2003).

Although IEC 61131 is focused on PLCs (programmable logic controllers) it took great care to embrace other technological solutions too, such as PC-based SCADA and DCS (distributed control systems).

Controller Communication

When computer first introduced for control application it was used to control system devices directly, this is also known as *direct digital control* (DDC). The use of single computer in controlling large manufacturing system is not realistic because of low reliability and high cost of wiring and installation. A few years after the birth of PLCs, Honeywell placed a bunch of microprocessors inside several industrial controllers and operator terminals, connected everything with a network, creating what is known as Distributed Control System (DCS) (Harrold 2002). In 1980, the introduction of local area network offered a high capacity low cost communication which made distributed computing a reality in many automation services.

As the industrial automation systems become larger and the number of automation devices increases, it has become very important for industrial automation to provide standards which make it possible to interconnect many different devices in a standard way. Open architecture controller allows controller software/firmware to be run on and with a wide variety of different vendor products. However, a common communication standard is essential. The use of common standards also enable suppliers to provide service and support open automation systems worldwide (Kinsella 1998). There are various standard have been developed over the years such as Ethernet, Profibus, CANOpen (European CAN bus) and AS-I (automated systems interconnect).

Djiev (2004) divides industrial communication networks into three hierarachical levels. The levels are shown in Figure 2.9 and their examples are as follows:

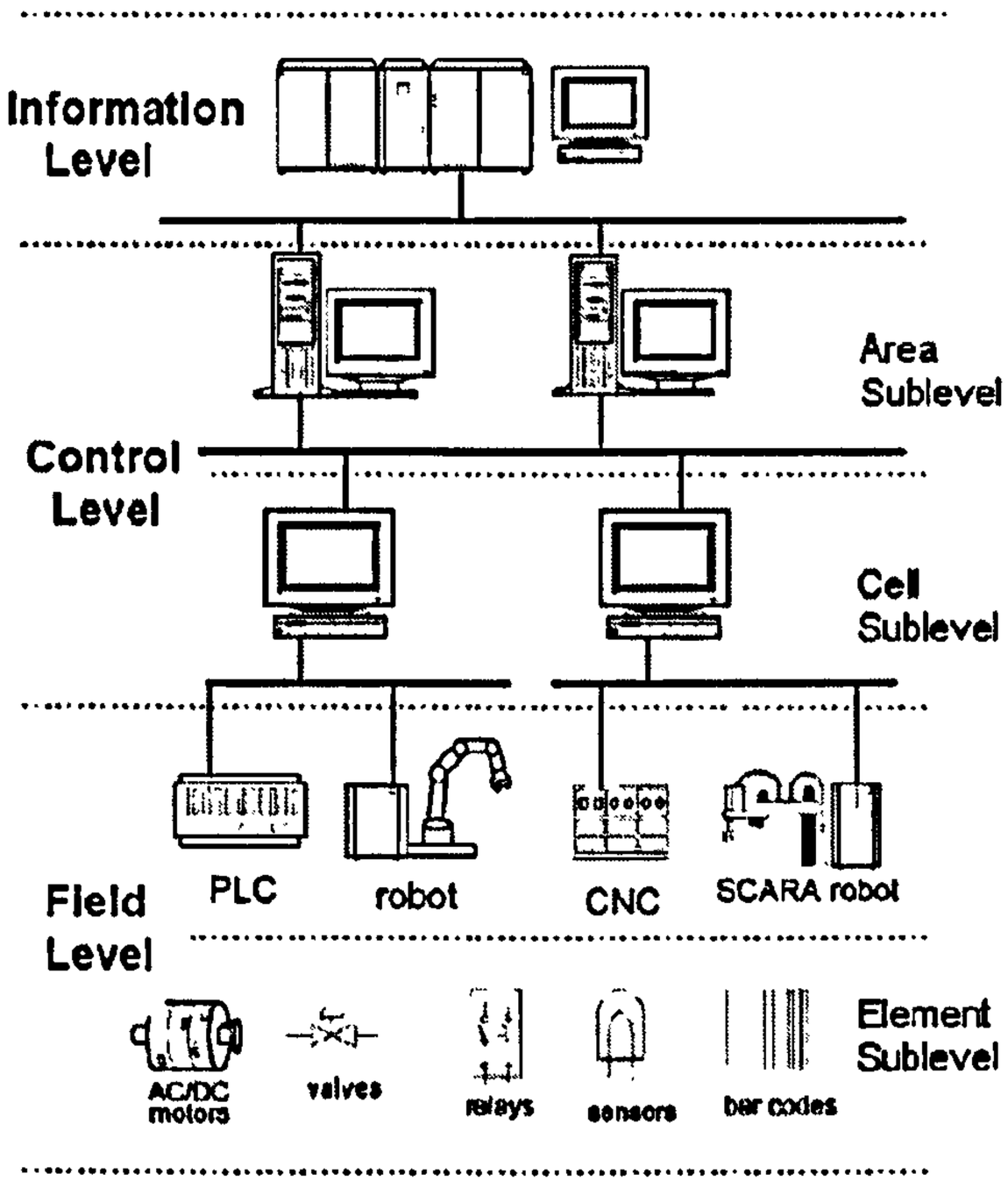


Figure 2.9 Hierarchical levels in industrial communication networks (Djiev 2005)

i. Field Level – CAN, DeviceNet, Foundation Fieldbus, Profibus-PA, LonWorks

ii. Control Level – ControlNet, Profibus-DP

iii. Information Level – Ethernet/IP, Modbus/TCP, FF-HSE, PROFInet, EtherCAT, Ethernet Powerlink.

In order to overcome the difficulties of having various communication standards, International Organisation for Standardisation (ISO) has come out with Open Systems Interconnection (OSI) (Danielsson 2002). The OSI standards permit any pair of

automation devices to communicate reliably regardless of the manufacturer. The OSI produces a model which describes seven layers of interaction for an information system communicating over a network, presenting a stack of layers representing major function areas that are generally required or useful for data communication between nodes in a distributed environment (Rees 2003). Each layer is isolated hence allows abstraction such that lower layers are not dependant on upper layers beyond what is needed to exchange data between the layers. This is especially important at the lower levels where the same data may have to travel across different media or link-layer protocols to get where it is going.

2.3.3 Open Architecture Control

Open architecture control (OAC) is a new term in the field of machine control. The idea of open architecture control is to separate between hardware and software for machine tools in order to enable design flexibility and to reduce the huge maintenance and upgrade costs. Traditionally, in the field of machine control and in other segments of machinery, the leading control companies (i.e. control hardware developers) keep developing systems where the software is embedded in their hardware. In order for user to run the application, they need to use proprietary hardware. The control companies will always try to maintain their market share and to avoid openness as long as possible. OAC is about the potential to separate between hardware and software for machine tools in order to enable design flexibility and to reduce the huge maintenance and upgrade costs. With the adoption of an open system, user will has easier reconfiguration and adaptation, wider choices in contribution of better performance and lower costs.

From the user's point of view openness is focused on capabilities to integrate, extend and reuse software modules in control systems. Perhaps the greatest advantage of using open technology is that users are free to choose the best product (hardware or software) to solve a given problem without regard to supplier. Sperling and Lutz (1997) describes there are five capabilities of modules in open control systems which are:

i. portable: a module can run in different control systems

ii. extendable: the functionality of modules can be extended

iii. exchangeable: a module can be replaced by one with comparable functionality

iv. scalable: multiple instances of modules are possible to increase performance

v. interoperable: modules cooperate (exchange data)

There are a lot of research materials on open architecture control. Some of them which are quite well known are OSACA (Open System Architecture for Controls within Automation systems), OMAC (Open Modular Architecture Control) and JOP (Japanese OMAC). Even OMAC, OSACA and JOP have initiated an effort to develop a Global HMI Standard to share a unified API for HMI design for OAC (Mathias and Hellmann 1999). The goal of the effort is to define a HMI API for most of the control products in the world. Major machine control manufacturers such as Fanuc, Siemens, Mitsubishi, Rockwell Automation, and end- users, such as GM, Boeing are participating in this endeavour (Katz et al. 2000).

Several standards have also been introduced for open architecture control. Unfortunately, in reality, not many products follow the standard strictly. For example, it is difficult to find a PLC product that supports all languages or easily import and export its program to and from another system. This is because there are already many proprietary standards (for example bus standards for PLC) for each company and each standard has its own advantage (Katz et al 2000).

2.4 SYSTEM SIMULATION

When designing manufacturing systems, simulation tools are used to replicate an actual or proposed system. Simulation is an analytical tool for designing or experimenting with complex systems. Simulation has been defined as the process of designing a model of a real system and conducting experiments with the model either to understand the system's behaviour or to evaluate various strategies for operating it (Pegden et al. 1995).

When simulation is first introduced in system modelling the aim is exploratory by nature. It enables evaluation and checking of different possibilities on the developed system in a faster execution time. Today's manufacturing industry is facing problems that have been growing in size and complexity over the last several years (Farahmand 2000). As a result, there is an immediate need for procedures or techniques in solving various problems encountered in today's manufacturing arena without extended shutdowns or expensive modifications (Clark 1996). Simulation has been proven as a great tool in evaluating complex and dynamic systems without the constraint of cost and time compared studying with real systems. Experimentation with the simulation helps the system characteristics to be learnt and performance data to be generated, hence enabling modification to improve performance. As the software development progressed concurrently with the construction of the hardware system, it became evident that a simulation of the expected hardware system would be extremely useful.

2.4.1 Development in Simulation Software

In early use of computer as simulation tools, simulation is understood as running programs written in languages such as FORTRAN, GPSS, SIMSCRIPT and SIMULA (Banks et al. 2000). It was not until the late 1970s a shift occurred from a program-centric view of the simulation process to a model-centric view started taking place (Page 1994). Software developments during this period were driven by the emergence

of desktop computers and microcomputers (Banks et al., 2000). In addition to that, following to the reduction of cost in graphical displays, simulation software packages featuring simple 2-D graphical user interfaces were developed. During the 1980s and 1990s further reduction in microcomputer and computer memory price, and the increase in their speed, saw the number of such packages increase significantly. Some of the more leading simulation software is Extend and Witness; and low level software like LabView and Simulink.

The 1990s saw 2-D based software evolve further into packages offering advanced 3-D visualisation capabilities, such as Flexsim, Arena, Simul8, and MicroSaint. With these packages it was no longer necessary to write a computer program in order to build a simulation model; instead the model could be built using graphical user interfaces (GUIs) that provided icons and menus for creating elements, connections, etc. This new user friendly software is called visual interactive modelling (VIM) system (Pidd 2004a) or just simulation environments (Banks et al. 2000), the term which is more popularly used. They become popular because they offer the prospect of rapid application development by people who are not computer professionals (Pidd 2004b). Today, there are hundreds of commercially available simulation software packages; some are 2-D, others come in 3-D, and a few offer both.

Further development sees new progression in simulation software which is not only capable of producing good 3-D simulation environment but also provide capability for user to manipulate the environment externally. The *openness* of simulation packages such as Delmia (Quest and IGRIP), AutoMod and Witness (Mecklenburg, 2001) make possible for interaction and communication between the simulation packages with other engineering elements. Hence, they provide a lot of simulation usage in variety areas of research.

2.4.2 Simulation Technologies in Manufacturing Applications

With enterprises facing tremendous time-to-market pressures, manufacturing systems must be implemented quickly and modified easily. Rapid process realisation and agility have been identified among the major imperatives for enabling the next generation manufacturing paradigm. This initiates the use of an integrated and realistic simulation environment that represents the physical and logical schema, and behaviour of a real manufacturing system including the products and manufacturing which serve as a platform for enhancing all levels of decision and control. Popularly known as *virtual manufacturing* (VM), the *digital plant* or *digital factory* is defined as an integrated simulation model of major subsystems in a factory that considers the factory as a whole and provides an advanced decision support capability (Jain et al. 2001).

One of the ideas of VM is to minimise any delays in manufacturing system development. The Next Generation Manufacturing report (NGM 1997) recommends pervasive modelling and simulation can speed up the development and modification of manufacturing capability by ensuring that the proposed system will work as planned. Integration among different sub-systems of the manufacturing system can also be validated using simulation to minimize the production ramp-up period.

For this reason, wide varieties of virtual manufacturing research directions have been proposed. Some efforts focus on advanced visualization of new system designs using virtual reality, while others aim on modelling the system functionality. Joines et al. (2000) and Chick et al. (2003) have made detail discussion on the direction and the future of simulation. These following sections will look into some recent developments that make use of simulation technologies that benefit machine system design.

2.4.2.1 Controller Simulation and Verification

Simulation has long being used in verifying machine systems before further resource is commited in building real machine. The concept known as *virtual engineering* is widely

used in building CNC and robotics machineries, and PLC control applications. However most of the time transferring the knowledge gain from simulation into working version is not straight forward. Loss of time and money in systems integration, testing, and installation is profound. There are increased pressures on finding better techniques to design, integrate, test, evaluate, and later reconfigure the control systems (Kolla et al. 2002).

Current trend sees a new concept of virtual engineering trying to reduce the gap between simulation models and real machine system. Among the objective is to allow verification and testing of the control software with the simulation environment. There are two approaches taken to reach this objective. First, is to have a simulation environment embedded with controller simulator which is then capable of transferring or generating codes for the controller applications. Second, is the use of the simulation environment to test a controller program (Chuen 2003). The earlier is known as controller simulation and the later as controller verification.

Project such as Realistic Robot Simulation (RRS and RRS-II) (Bernhardt et al. 1995, 2001) and VIR-ENG (Adolfsson et. al. 1998, Olofsgård et al. 2002 and Chong et al. 2002b) have proved that such applications can contribute in closing the gap between simulation and real system. RRS and RRS-II are aimed to achieve a more realistic simulation of robot controller by improving robot simulation accuracy and robotic offline programming. VIR-ENG making benefit the IEC 61131-3 programming environment to produces a seamless integration between simulation model and real manufacturing machines.

2.4.2.2 Simulation-based Commissioning

Traditionally, for control engineers, control designs and programs can only be tested and validated only when real equipment is placed on the shop floor. Building and commissioning phase is usually on the critical path of bringing a new product to market.

Direct costs, production start delays, lost revenue and, on occasion, lost market share may result from design change or rework. During the ramp-up of production lines, as well as in the machine building and digital special machine building business, time to ramp-up and risk of error have become crucial factors. The shifting of the testing and commissioning process from the shop floor onto the control engineer's desktop, accelerates the planning and engineering process, the commissioning phase, and production ramp-up. Among the issues which the simulation-based commissioning tries to resolve are (Tecnomatrix 2003):

i. allows control departments to work *concurrently* by sharing manufacturing information

ii. catch logic errors well before ramp-up

iii. evaluate controller program changes on the virtual model (instead of taking risks on the real equipment)

iv. prove the feasibility of the production cell and its time cycle

v. cut time and cost by creating shop-floor documentation off line

Simulation-based commissioning is also called Virtual Commissioning (Tecnomatrix 2003) or Soft-Commissioning (Schludermann et al. 2000).

2.4.2.3 Virtual Training

Simulation is the key for virtual training. Using printed documentation and slide shows for training purposes are often not sufficient enough to convey the entire complexity of a machines design and functions. For that reason, it is still common for operators and maintenance staffs to be trained by utilizing a piece of real equipment. Some

manufacturing companies even maintain specialized training centres that are equipped with a variety of training equipment. Bluemel et al. (2003) indicates, although this traditional approach provides adequate training, there are some disadvantages that have become more apparent in recent years which are:

i. *Increasing variety of products*: Within manufacturing, there is a trend toward customized products. It becomes increasingly difficult to have many kinds of products available for training purposes.

ii. *Shortening of time interval for applying modifications*: Often, products are constantly improved and modified. Thus, the equipment found in the training centre becomes quickly outdated.

iii. *Training centre is bound to a specific location*: With the globalization of the market, companies extend their business to the international marketplace. Training centres, however, are still mostly located near the manufacturing location, which in turn increases travel expenses.

iv. *Experts are only available locally*: Although training centres provide excellent training facilities, not every possible failure can be foreseen and trained. Due to the fact that there is currently no adequate approach to convey this expert knowledge to the customer's site, the expert often has to physically travel to the customer's location.

v. *Increasing cost for equipment*: Products not only become more complex, but also more expensive and damageable. Frequently taking apart and putting back together a machine also accelerates wear and increases costs due to depreciation. Facing an increased competition in the market, it becomes more important to limit these costs.

Using cutting-edge technology, virtual training may take place in a realistic, simulated

version of the actual facility, complete with the actions, sights, and probably sounds of the system. Virtual training is the one of the most advanced methods of teaching skills and processes to employees. Properly designed computer based training applications have demonstrated to be successful as learning tools (Fletcher 1996 and Barnes 1996). Virtual reality technology contributes in improving realism and effectiveness of virtual training. NASA, for example has been using virtual reality simulation to train astronauts on shuttle functionality for space missions since 1992 (Sherman 1997). Automotive industry also has shown great interest in the use of simulation for the training of design, production and maintenance [Gomes and Zachmann 1999].

In manufacturing, virtual training has been used widely in training CNC machine user. VERTICUT for example has computer-based machine training centre at the Foresight Centre at the University of Liverpool. A good demonstration of virtual training in manufacturing is the Zanussi Electro-Mechanica (ZEM) industrial training module developed under IRMA[6] project. The industrial virtual environment training application module is set in the context of monitoring and fault rectification of a refrigerator compressor motor assembly and test line at ZEM. The virtual environment comprises of a refrigerator motor and stator line manufacturing plant modelled in Delmia (IGRIP) software. The manufacturing line is modelled in varying levels of detail using the simulation tool and virtual environment created for operator or supervisor training utilising head-mounted displays and data-glove interaction (Modern et al. 2003).

Advancement in internet technology also witnesses the possibility of having internet-based virtual training. This provides not only remote training for the industries but also benefits the education sector (Rafe et al. 2001). A common problem faced by institutions nowadays is the limited availability of expensive machines and control

[6] IRMA is an Intelligent Manufacturing Systems (IMS) approved inter-regional Virtual Reality (VR) project with partners from the European Union, Newly Associated States of Eastern Europe, Switzerland and Japan. The key aims of the project are to build, integrate, demonstrate and evaluate VR 'application demonstrators' which reflect different but complementary aspects of industrial manufacturing, and integrate different commercially available software simulation tools with Virtual Environment software (VEs) that support real-time, two-way, interaction (IRMA 2004).

equipment, with which students in the educational program can work, in order to acquire valuable *hands on* experience. A virtual laboratory approach which is based on the concept that it provides a working facility for hands-on training whilst reducing the need for multiple high cost actual devices helps to provide an alternative solution (Safaric et al. 2001).

2.4.2.4 Remote Application

Simulation has also been used as a tool in remote applications such as teleoperation and remote monitoring. Teleoperation allows a human operator to use a visual display and a "master" manipulator (e.g. a joystick) to manually control a remote "slave" device such as a vehicle, machine or robotic arm. Force feedback is sometimes used to backdrive the input device and provides tactile clues to the operator. Presenting realistic visual information and contact forces to the operator improves task performance and increases the sense of *telepresence* (Sheridan 1993). Teleoperation allows user to execute or to control remote tasks, i.e. without being where the action takes place. Such application is useful in hazardous and unreachable situation.

Remote monitoring on the other hand is about collecting and analysing data of a system, providing the ability to monitor and/or providing simulation-based service and maintenance even from a remote, off-site location. Making use of internet, new intelligent monitoring tools and high-speed plant floor and telecommunications networks, manufacturers can monitor or replay their system in home based simulation environment. Together with historical data, this allows the simulation model to be used for analysis in particular for exploring virtual service & maintenance and/or scheduling scenarios (De Vin et al. 2004, Sundberg et al. 2006 and Pettersson et al. 2007).

To accommodate the ability of these remote applications, simulation software such as IGRIP and QUEST has included what they call *telerobotics feature* which enables external manipulation of developed model. AutoMod also allows external

communication via its Model Communication Module (MCM). These features indirectly contribute in the use of simulation with real hardware which is discussed in later sections.

2.4.3 Simulation in Manufacturing System Design

In manufacturing, simulation is used at various area of manufacturing system design. The simulation industry has now matured to the point that almost all manufacturing processes and flow of materials through the enterprise can be modelled and validated (Freedman 1999). In term of its use, the author perceives simulation application in manufacturing can be categorised into three different levels:

i. Low-level Simulation – simulation of machine specific application i.e. specific continuous operation such as servo control and process control. At this level mathematic model generations for the simulated system are essential and important. Evaluation focuses on analyses system dynamic responses such as stability, steady-state error etc. 2-D or graphical simulation representation is usually sufficient at this level. Examples of software used at this level are Simulink (from MathWorks) and VisSim (from Visual Solution).

ii. Mid-Level Simulation – simulation of manufacturing machine(s) such as robots, assembly stations etc. Simulation at this level can be in mixed operations which involve evaluation of both continuous and discrete operations; but usually continuous operations are of concern. Focuses at this level are on system kinematics analysis, path planning, collisions detection, off-line programming, machines coordination etc. Simulation representation is normally in 3-D which gives in depth understanding and evaluation of the system studied. Examples of software at this level are IGRIP (from Delmia), eM-Workplace (formerly RobCAD – from

Tecnomatix) and Dymola (from Dynasim)

iii. High-Level Simulation – high-level simulation is known as *discrete event simulation* (DES). Simulation at this level typically used for facilities material flow simulation, work in progress, queues and transportation time, facility layout etc. (Nwoke and Nelson 1993). Simulation representation can be in 2-D or 3-D however some software provide both 2-D and 3-D view. Examples of software use for simulation at this level are Witness (from Lanner Group), Arena (from Rockwell Automation), Quest (from Delmia) and AutoMod (from Brooks Automation).

Between the three levels of simulation application used as tools in manufacturing, the Mid-Level Simulation is the most recent and at present attracts a lot of research.

2.5 SUMMARY

A literature review focusing on machine system design has been presented in this chapter. A brief description in technology related to manufacturing systems, industrial control systems and simulation technology application has been provided. The intension is to give readers an introduction to and some basic information about the field and closely allied research.

The following chapter will look in more detail into the related research.

CHAPTER 3

RELATED WORK AND MOTIVATION

3.1 INTRODUCTION

When simulation is first introduced on manufacturing system design the focus is more on visualising the operation of an unbuilt system rather than on the system controller. Current research, however, has shown a great interest in using simulation for both system and controller design. Simulation models are increasingly being used in problem solving and in decision making. Clearly, testing a control system becomes much easier if the system is first designed in a computer-based virtual world. Simulation has long being used as a tool for design, analysing and evaluating systems to be built. Computer simulations allow all manner of extreme conditions and "what-if" scenarios to be tested safely. If a test uncovers a flaw in how the control system handles a certain situation, the necessary modifications can be implemented and retested merely by updating the control system's program in the simulator. Once all the what-if questions have been answered satisfactorily, the fully tested control program can be lifted right out of the simulator and run as-is in the real control system.

Having a fully tested control system is vital (Auinger et al. 1999). The only certain way to test how a control system handles every possible situation is to write the code, install it in the controller, and try it out on the system. Normally, testing takes place during the start-up phase of the system to be controlled which is an expensive, risky and error-prone way of developing control systems (Pfeiffer et al. 2003). This is also a very costly means of uncovering design flaws. A control system failure can shut down the plant and require expensive modifications. In the worst case, the entire control system may have to be scrapped, redesigned, and rebuilt (Van Doren 2003).

At present, a simulated control system can be installed in a simulated plant without the expense of real equipment and without the risk of disrupting the real plant's production. In this virtual world, both the plant and the control system exist only as customized code in some kind of a simulation computer or simulator. Simulations can be equally useful after the control system has been completely debugged and installed. As experienced operators run the real plant, new operators can be trained to do the same using a simulated plant and a simulated control system. Simulation training is often conducted in a mock-up of the control room equipped with all the operator displays used in the real thing. The existence of 3-D simulation packages helps to provide some reality into the visualisation.

One of the most important questions that have arisen since is how to verify control systems and control logic. Methods for off-line programming and verification of industrial control systems have been developed over recent years (see, Hanisch 1997, Adolfsson et al. 1998 and Hassapis 2000). Although off-line simulation can be used to help address the control system verification requirements, additional desired features for system verification (such as real-time capabilities, auto code generation etc.) still have not been fully accomplished. Several researchers (Kanai 1999, VIR-ENG 2001 and Min et al. 2002) have explored improved methods for industrial control verification but almost all the projects reported are solutions for specific application or for particular development frameworks. Several successful implementations have been realised of industrial control system emulators (see Auinger et al.1999, Schludermann et al. 2000 and Danielsson 2002), however these systems only partially address the following important areas:

- ability to provide real-time off-line programming, verification and optimisation
- simulation/emulation of the system with discrete, analogue and hybrid control signals
- software and hardware independent architecture, i.e. the architecture is not dependent on any specific simulation software or industrial control system

manufacturer. In other words, the ability to mix and match simulation software and control systems from different vendors in one application.

- full synchronisation between the simulated environment and the real system. A system is said to be real-time only if it is correct in operation dependent upon not only its logical, but also its temporal behaviour.

Thus, a generalised architecture, which can address these specific features, would represent a significant advance.

The main strand of this research is about manipulation of virtual environments in realising interaction between machine(s) and controllers and other related elements in manufacturing systems. In formulating the concept of the R-V system and its architecture, the following requirements have been identified:

- techniques to realise real-time virtual environments

- schemes for control system design, verification and validation

- methods for integrating seamlessly the simulation environment and the real environment of the machine system

- means that can help to enhance and/or accelerate the machine system design process

This chapter will look at different techniques being used in various area of research and their differences, before presenting the technique used in the proposed real virtuality system in the chapters to follow.

3.2 BRIDGING THE REAL AND VIRTUAL ENVIRONMENT

In simulating the controlled system behaviour, the most common way is by transforming the physical system directly into a virtual environment representation. This is shown in Figure 3.1. The virtual system and virtual controller will have similar characteristics as the physical system. The system will have virtual operations (discrete and continuous) and the virtual controller runs a simulated program. Interaction between virtual system and virtual controller is in the form of discrete data (Dd) and continuous data (Cd) comparable to discrete signal (Ds) and continuous signal (Cs) in the real system. Such arrangement is known as *offline simulation*[1].

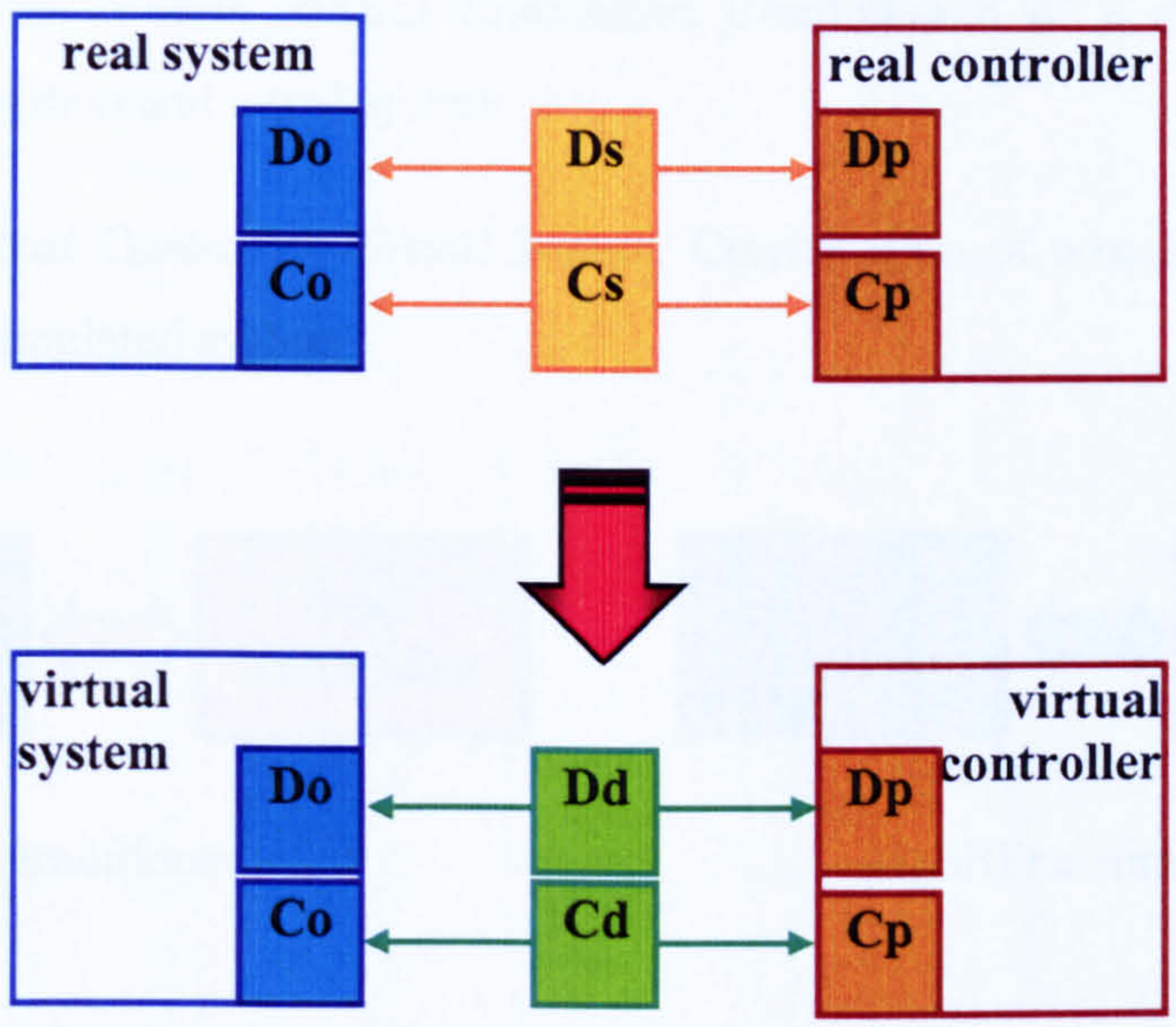


Figure 3.1 Direct physical system representations in simulation environment

As the existence of 3-D simulation has been proved to be useful in providing realism in visualisation, researchers move towards using a real time virtual environment. Real-time systems are those in which the correctness of the system depends not only on the logical results of computation but also on the time at which the results are produced.

[1] It should be noted that the term offline simulation used here is not offline programming (OLP) as the later is subset of the earlier. Offline simulation refers to any techniques which make use of a simulation environment to evaluate both machine and controller in the simulation environment.

Since the advancement of computer technology (software and hardware) efforts have been made to close the gap between the virtual and physical environment. In general, there are four possible approaches to test control systems which can be distinguished based on the possible combinations between reality and simulation as shown in Figure 3.2 and can be described as follows (Auinger et al. 1999, Verstegt and Verbraeck 2002, Pfeiffer et al. 2003):

i. The *traditional way*: Both the controller and system already exist. The control system is tested after installation.

ii. *Offline simulation*: Both the controller and the system are simulated.

iii. *Real System Virtual Controller*: Combination of a simulated control system and a real system.

iv. *Real Controller Virtual System*: Combination of a real controller and a simulated system.

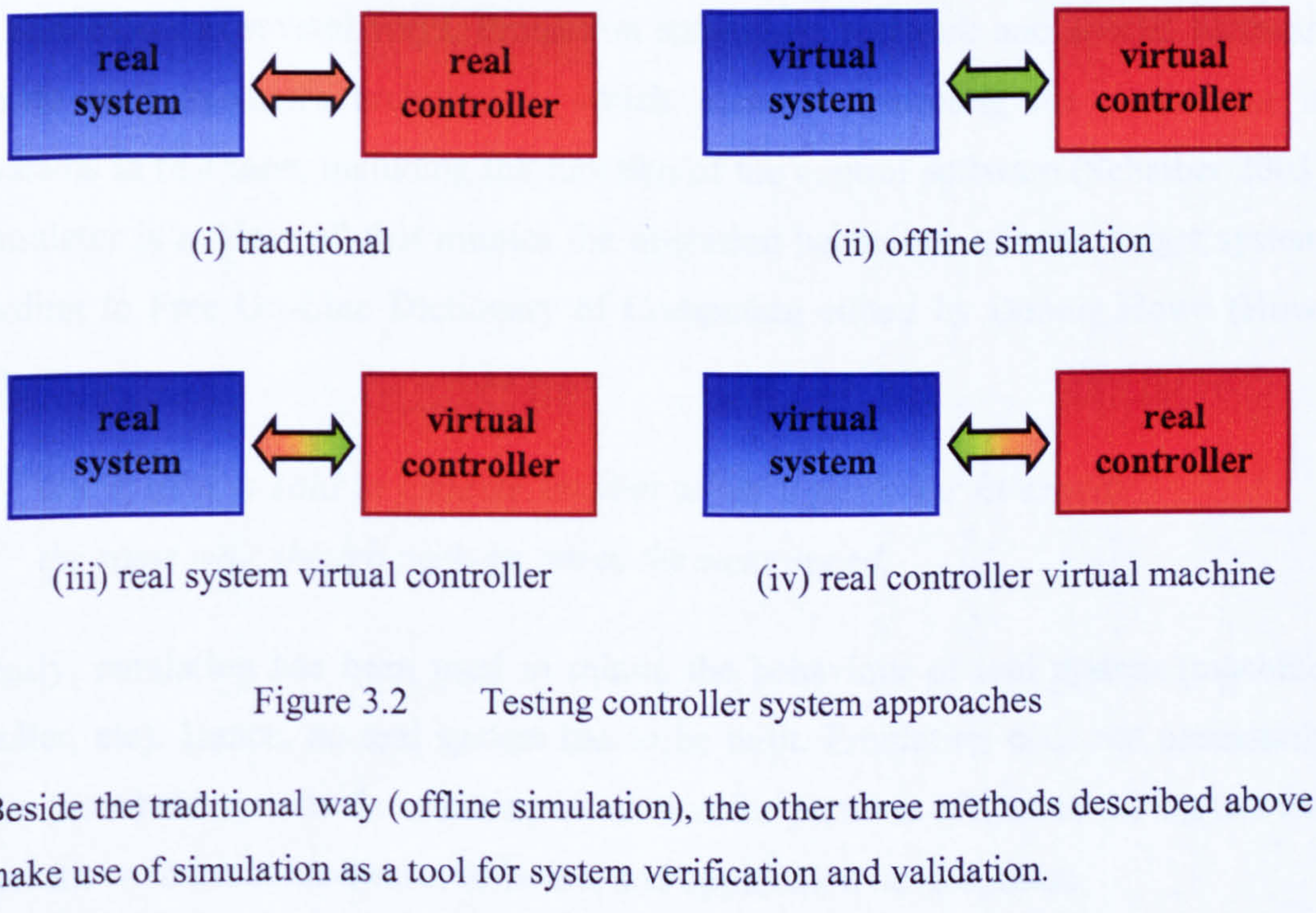


Figure 3.2 Testing controller system approaches

Beside the traditional way (offline simulation), the other three methods described above make use of simulation as a tool for system verification and validation.

For long, offline simulation has been accepted as a good tool to verify and validate control program. However most of the time the program generated in virtual environment is not directly applicable to the real controller. Time and again the program has to be modified and refine before it can be used. Moreover, there are thousands of controllers using a variety of in-house developed software to adapt to the market conditions. Current trends suggest the use of combination between reality and simulation. The adoption of reality into simulation has been proven to be very useful and can also be used beyond program verification and validation.

3.2.1 Emulation

Bringing reality into simulation is not an easy task. The term *emulation* is normally used to describe a simulated environment with real time capabilities. Coupling real hardware with the simulation is one of the techniques to provide an emulated environment. However emulation may not necessarily involve the use of hardware. Emulation differs from simulation in several ways. Emulation substitutes software and special hardware for parts of a product's architecture, which permits examining the product or its subsystems in real time, including the function of the control software (Scheiber 2003). An emulator is a "device" that mimics the operation behaviour of some target system. According to Free On-Line Dictionary of Computing edited by Dennis Howe (Howe 2004),

> *one system is said to emulate another when it performs in exactly the same way, though perhaps not at the same speed.*

Currently, emulation has been used to mimic the behaviour of real system (machine, controller, etc). Hence, no real system has to be built. Emulation does not necessarily involve the whole system. Substituting emulation for portions of system during test can also permit concurrent hardware, software, and application development.

Emulation has been proven to make training much easier, less expensive, safer and

more efficient. Flight simulator is a good example where the emulation technology is being used for a safe inexpensive training. The advantage of such training system has been acknowledged by many. Another advantage is that training on an emulator allows one to gain experience with rare or dangerous situations that one would not want to create on real systems for training purposes.

In manufacturing system applications, Mc Gregor (2000) has highlighted that emulation is useful and economically justified under the following circumstances:

i. when the testing is otherwise due to be carried out on the critical path of a project.

ii. when the time available does not permit full testing before start up.

iii. when actual system cannot be sufficiently loaded to fully test the control system.

v. when the cost of real testing is greater than that of emulated testing.

Research has also shown the benefit of using emulator in system control related application. According to LeBaron and Thompson (1988), the difference between a simulator and an emulator in system control application is that the emulator is an exact representation of the real controller. Since the emulator is an exact representation of the real controller is it easy to exchange logic between the real controller and the emulator (Danielsson and Moore, 2003). The use of emulation has allowed testing of control systems faster than in real-time and under safe conditions. Rohrer (2002) has suggested that the use of controller emulation at the development stage of system design will speed up the time for the system to become operational. This is illustrated in Figure 3.3 below.

The benefits of doing an emulation of a system are numerous. Schess (2001) describes the advantages of emulation as follows:

i. Provides integration testing/debugging in a lab environment.

ii. Facilitates logic check-out prior to the testing on the plant floor

iii. Reduce the amount of time people spend in the field. This reduces employee burn out and reduces start up cost.

iv. Cost savings from a reduction in debug time spent on the plant floor.

v. Reduce start up time. Time to market can be substantially decreased.

vi. Ability to test scenarios that won't actually exist in the field until a future date. This allows the end user to hold the contractor to all of their obligations and not be surprised by system malfunction when full production requirements of product variability and throughput are experienced.

vii. Reduces debug time. Test scenarios may be setup and tested far quicker through emulation than on the plant floor.

viii. Allows great visibility of the control system prior to installation

ix. Allows testing of complex product blends and complex scenarios

x. Assure schedule reliability

xi. Provides an effective tool for training

A lot of research has already been done or is still being pursued which shows how real time simulation or emulation can be a useful tool in system design and controller development of manufacturing system. Real time simulation is about bridging the two worlds, the simulation world and real world. The following section will look into several developed projects and relevant research which is able to bridge real and virtual environment. Each of them may differ in their emphasis but have provided a way to

bridge both worlds. The approach taken and level of work that has been conducted also differs from one another.

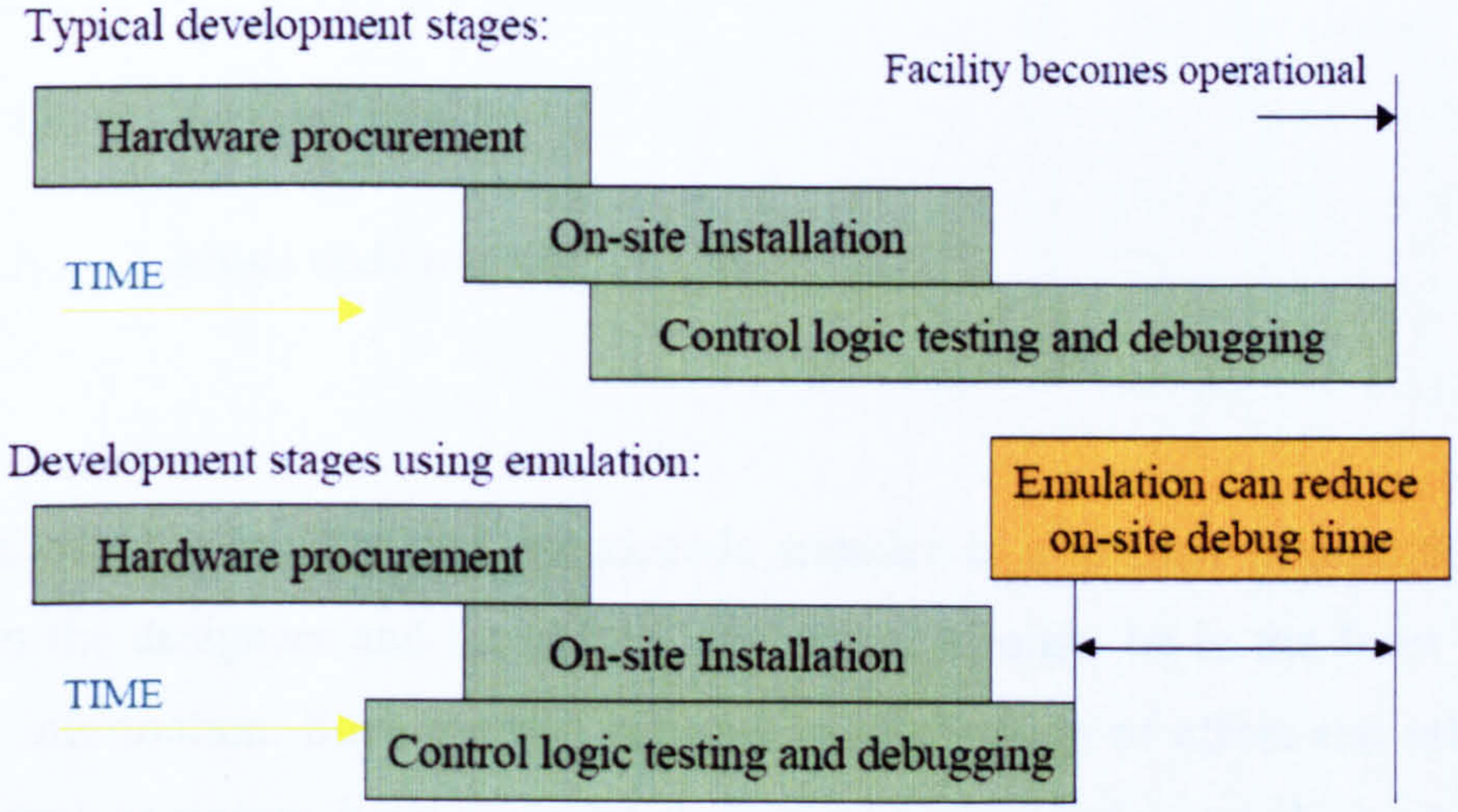


Figure 3.3 Benefit of using controller emulation at the development stage of system design (Rohrer 2002)

3.3 TECHNIQUES IN BRIDGING THE REAL AND VIRTUAL ENVIRONMENT

Initial technique into bridging the real and virtual environment is by using *offline simulation*. The idea of having an offline simulation came from the realisation that system control logic has to be developed again for the real control system when quite a similar logic has already been created while building the system's simulation model. Capability of using the logic developed in simulation model onto the real controller contributes to the shortening of machine delivery time. The technique of offline simulation is largely applied in evaluating system's discrete operations. Early work from McHaney (1989) recommends four methods of transferring discrete logics from

discrete model to the real system, which are as follows:

i. philosophic transfer

ii. pseudocode transfer

iii. database transfer

iv. actual code transfer

The philosophic transfer and pseudocode transfer is a manual information exchange between the designers and the control engineers. It might be in the form of verbal or written information. Such method is prone to duplication of effort and other errors as the control engineers have to make their own interpretation on the code. Database transfers may be possible by making use of a common data format which is agreeable and understandable between the designers and the control engineers. This is achieved by using a general purpose language such as FORTRAN or C. Thus, the simulation language should have the ability to use FORTRAN or C subroutines. The actual code transfer removes the need of duplication. However, McHaney indicates more cooperation is needed between the designers and control engineers as both parties may not be familiar with each others field of work.

The work of McHaney provides a reference for further development in bridging simulation and real environment. Further work sees the manipulation of simulation data within the simulation environment to produce a useable real controller logic which is transferable to the real controller rather then generating raw type of data. Such technique has been proven successfully in VIR-ENG project. By adopting the newly introduced IEC 61331, within the VIR-ENG simulation environment, controller codes are produced for the real controller based on the simulation model. Although the technique helps in preparing the control code for the real system, there are some modifications needed to been done on the transferred code at the real controller side

(Chong 2002a). VIR-ENG also makes use advanced 3-D simulation model which helps better system realisation in the simulation environment

Offline simulation technique is proven easy to apply for discrete operation especially for simulation at high-level. For applications which involve continuous operations, offline simulation tends to be more difficult unless the dynamic model of the system is already available such as in robotics application. An example of fully offline simulation which evaluates both discrete and continuous operation is the Realistic Robot Simulation (Bernhardt et al. 1995, 2001). RRS has been proven to be useful in robotics machines and benefits from the fact that robotics machines are well defined. However, RRS is not generic. The controller developer has to provide a support for RRS services in order the system transition to work smoothly.

Hassapis (2000) also has demonstrated the control of both simulated discrete and continuous operation by adopting IEC 61331 language. The continuous operation is a model of a multiple input multiple output (MIMO) distillation column. The method is named *soft-testing* which takes into consideration the reaction of the controlled plant in developing the PLC program hence increases the confidence level on the compliance of the software to functional and temporal requirements. The need to obtain a high confidence level on the correct software operation arises from the fact that in most of the cases it is quite dangerous and expensive to test unproved PLC operation by linking it with the actual facilities that it is going to control (Hassapis 2000).

Researchers have a serious interest in adopting real-time simulation which they believe leads to a better understanding of the system design especially for continuous operation. Initial work in this area is hardware-in-the-loop (HiL) which incorporates part of the system (either the machine or controller) as a real hardware meanwhile the other is simulated or emulated. HiL attracts a wide range of research from aircraft (Georgiew 1995, Mansoor et al. 2003), automotive (Grund 1995, Hanselmann 1996, Baracos et al. 2001), motion control (Roche 1993, Zupancic 1998, Linjama et al. 2000, Lu et al. 2003) and robotics (Siegward 1998, Temeltas et al. 2002). Zupnacic (1998) for example, make

use of MATLAB/SIMULINK to automatically generate code for Mitsubishi PLC after the simulate system has been successfully tested in simulated and real-time HiL environment. The system being controlled is a hydraulic plant. Lu et al (2003) also use MATLAB/SIMULINK to produce a real-time dynamic simulation (emulation) model for converter controlled brushless DC motors used for hybrid vehicles. The emulated environment is then connected to the designed real controller hardware. The aim is to examine the real time performance of the designed controller against the model. Thus, there is a possibility of using an emulated model as a replacement to a real system in evaluating the controller under construction.

Although almost all HiL studies are more focussed on real-time evaluation of low-level control system where the main element is of continuous operation, they demonstrate that real-time simulation helps to understand the system better, produces a reliable usable code and increases the level of confidence towards real applications. With the evolvement of simulation package from a simple 2D to 3D solid model representation, studies have also progressed towards a real-time representation for high level discrete event simulation. Real time discrete event simulation is a means of verifying a real time system in which a simulation model may interact with a surrounding environment, such as software components, hardware components or human operators (Cho and Kim 2001). For example, Coca-cola Enterprises (CCE) in Atlanta has revealed how AutoMod (from Brooks Automation) via its Model Communication Module (MCM) *is* able to establish communication between the PLC hardware and the simulation environment which enables them to emulate a production line before it exists. MCM allows the simulation environment to interact with the controller and hence enables emulation of real discrete event operation, especially in material handling environment. CCE claims they are able to develop and test line control in native PLC environment (Hodgson and Katz 2000). Vedapudi (2001) also shows how AutoMod is used to emulate a car assembly line at General Motors. The assembly line model communicates with Allen Bradley PLC emulation software via Dynamic Data Exchange (DDE) protocol. Although AutoMod shows the capability to bridge between virtual and real environment, Fritz et al. (2002) however contested that AutoMod is not capable to

communicate with all type of controller. It is only able to communicate with the controller which complies with its standard protocols[2].

Besides AutoMod, there are also a few other commercial packages which provide similar bridging capability between the simulation package and the real world. However all of them are not totally open package as they are only able to accommodate some particular controller or communicate via certain protocol only. For example, em-PLC (Tecnomatrix 2004) is developed mainly for Siemens PLCs. Cell Control (from Delmia) supports IEC 61131-3 Ladder, Structured Text and Sequential Function Chart, thus only compatible with controller which uses the corresponding language (Cell Control 2004).

Schuldmermann et al. (2000) however has proposed a more generic method of bridging discrete event simulation and real environment. They use standard I/O card as a means to interface between the PLC and the simulation environment. By doing this, the system has only to deal with a single type of device (the I/O card) regardless any type of PLC used on the other end. The simulation software used in the study is Arena (from Rockwell Automation). The technique is called soft-commissioning (SoftCom). However the system is meant for system with discrete operation only.

Introduction of real time discrete event simulation in 3D environment has been proven to be very useful in providing better understanding of the developed system. This attracts researchers to explore similar techniques at machine level simulation which will benefit machine designer and machine builder. However in machine design, the system involved is more complex because it relates to both discrete and continuous operation. Some researches have already made attempts to provide real time simulation for this mixed operation. A research at University of Michigan (Min et al. 2002) has developed a real time simulation for a CNC machine which occupies both discrete logic and continuous control.

They use a real PC-based machine tool controller to run a virtual CNC machine using

[2] AutoMod's Model Communication Module (MCM) only supports communication using either Dynamic Data Exchange (DDE) or Object Linking and Embedding (OLE) for Process Control (OPC).

Computer Aided Software Engineering (CASE) tool, ControlShell (from Real-Time Inovation, RTI Inc.). Both PC-based machine tool controller and simulator software run on a single computer. The controller and simulator communicate through shared memory. Since the controller used to drive the simulator is a real controller, it can then be used directly to drive the real system. This is shown in Figure 3.4.

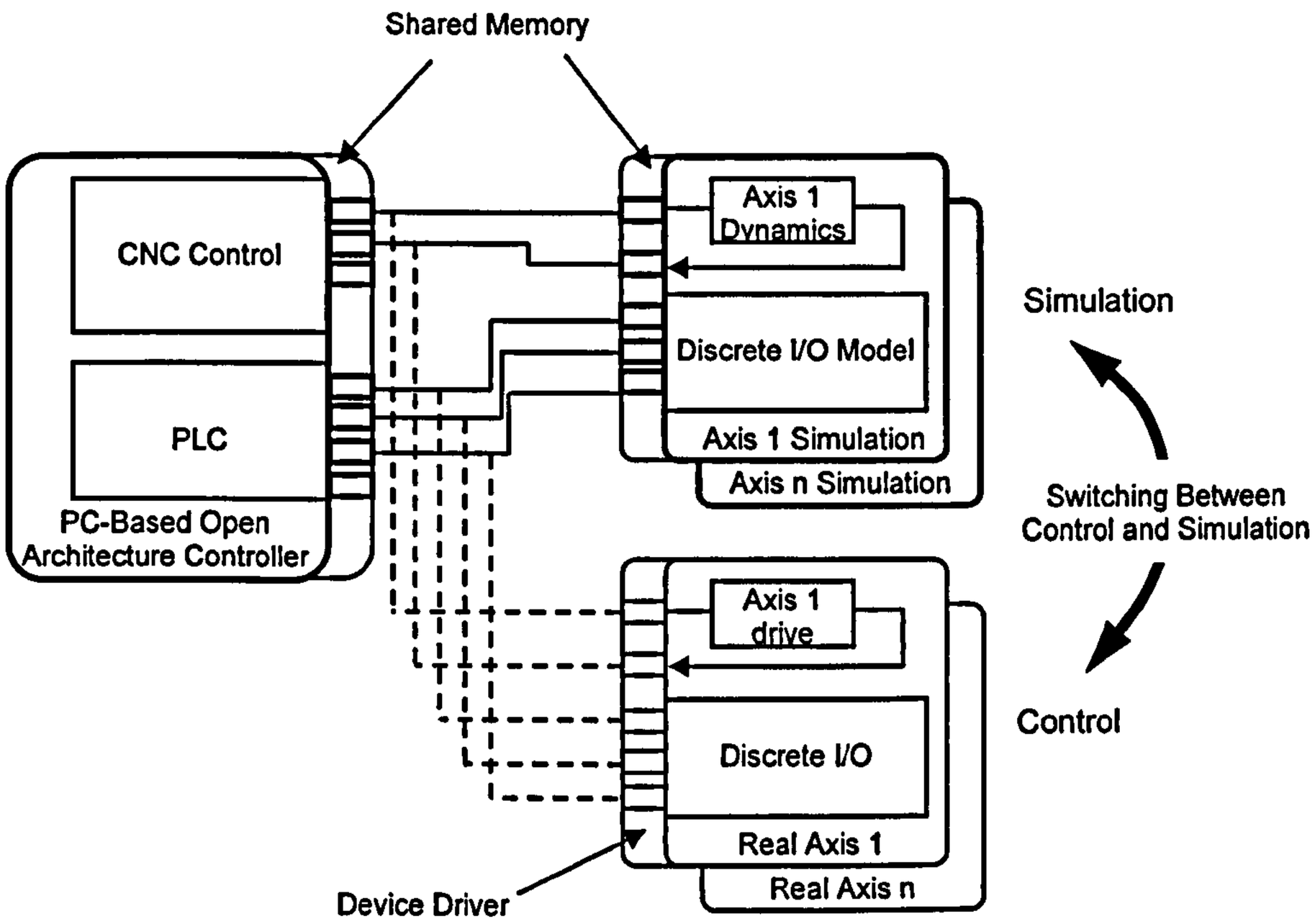


Figure 3.4 University of Michigan's machine tool simulator structure (Min et al. 2002)

In a project called RealSim[3] (Real-time Simulation for Design of Multi-physics Systems), German Aerospace Centre (DLR) together with KUKA have developed tools that ease the design of new robots or of variants of existing robots by taking into account the interaction of mechanics, electronics and software systems of a robot early in the design phase. The main objective is to reduce development cost and time and

[3] The RealSim project is carried out in the IST program of the Commission for the European Community (CEC), contract Number IST-1999-11979.

improve robot performance (RealSim 2004). The robot dynamics is modelled using an object-oriented modelling language - *Modelica* which is used with commercial simulation package *Dymola* supplied by a Swedish company - Dynasim. The result has been demonstrated at the Hannover fair 2001 which shows a virtual KUKA industrial robot been driven in real-time by a standard KUKA control panel and animated with Dymola. This is illustrated in Figure 3.5 below.

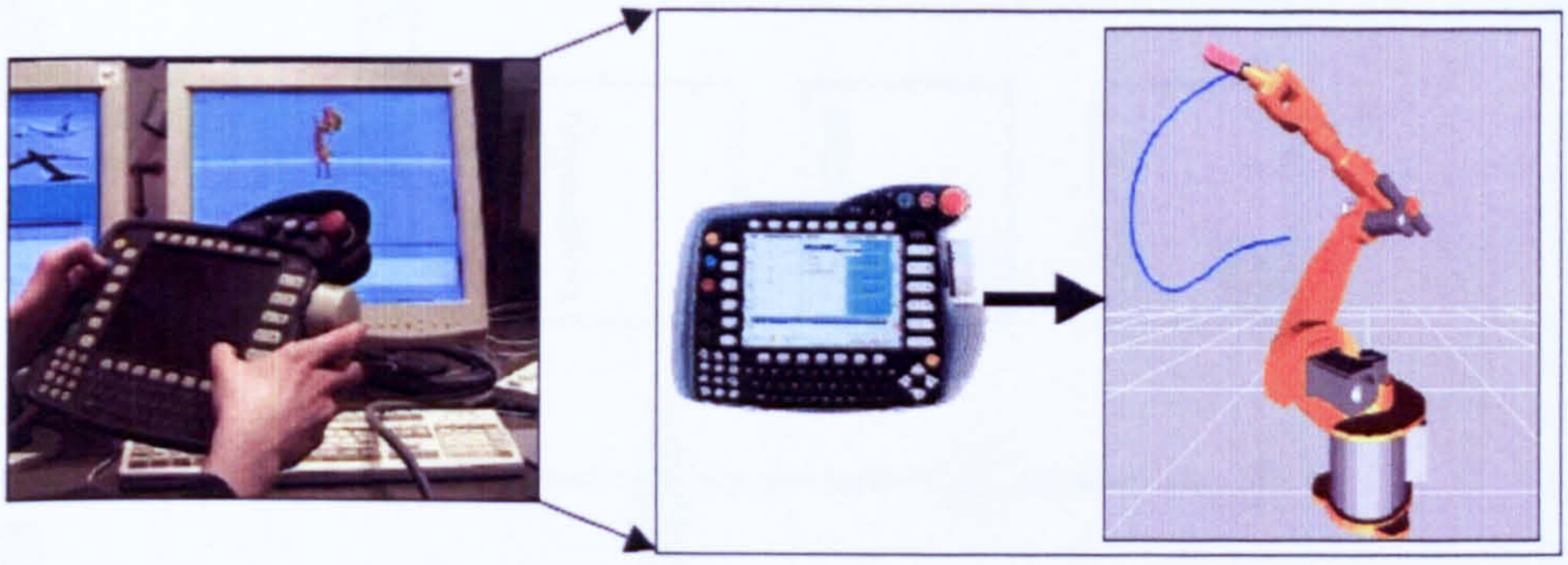


Figure 3.5 Hannover fair demonstration of robot real-time simulation controlled by KUKA control panel (Dynasim 2004)

Danielsson (2002) has developed a different approach to verify industrial controller's discrete and continuous operation. He develops an emulator for an industrial controller (BiFas controller) to control a system consisting of servomotors and sensors. By utilising an emulator for the industrial controller, Danielsson is able to construct a real time system simulator without using any real hardware. He also manages to avoid having to formularise the dynamic model of the system by applying the set point value as the current feedback servo value. Thus, the controller always regards itself has having achieved the required value. The system kinematics is built within the simulation software (Robcad and IGRIP) used to visualise the system in operation.

In this section various approaches used in bridging between the virtual world and real world has been reviewed. To provide a better comparison, Figure 3.6a and Figure 3.6b simplified all these techniques.

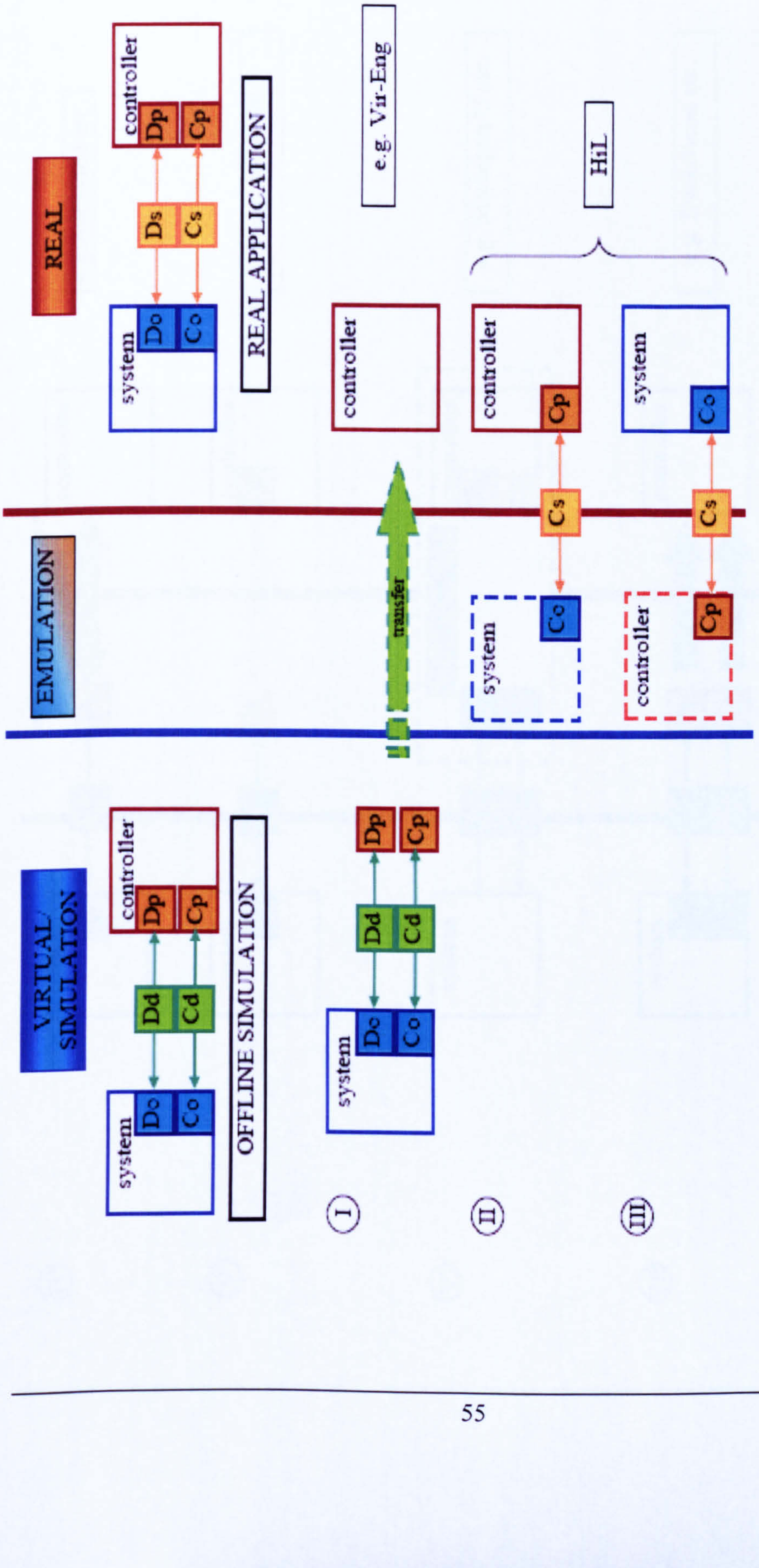


Figure 3.6a Comparison on virtual and real environment bridging techniques (Do=discrete operation, Co=continuous operation, Dd=discrete data, Cd=continuous data, Ds= discrete signal, Cs=Continuous signal, Dp=discrete program, Cp=continuous program)

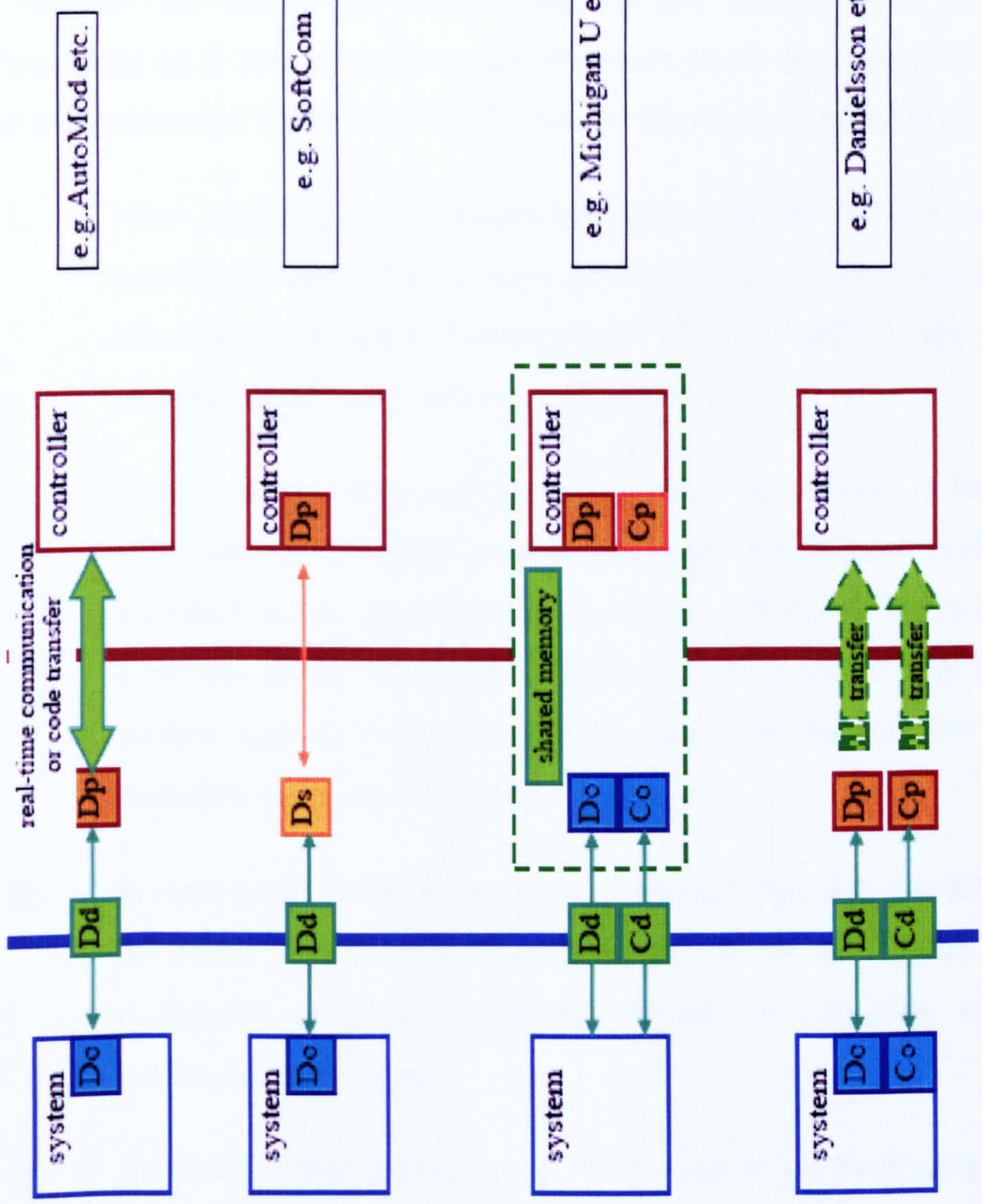


Figure 3.6b Comparison on virtual and real environment bridging techniques (Do=discrete operation, Co=continuous operation, Dd=discrete data, Cd=continuous data, Ds= discrete signal, Cs=Continuous signal, Dp=discrete program, Cp=continuous program)

3.4 RESEARCH MOTIVATION

In earlier sections in this chapter it has been demonstrated that it is possible to bridge the real and the simulation environment. Various techniques in bridging the two environments have also been discussed. However, there are several issues which still have to be addressed which motivate this study. They are described as follows:

i. Most of the research focuses on either real time simulation of low level continuous operation or high level discrete event operation. Real time simulation research at machine level which combines both continuous and discrete operations is still not well explored.

ii. Almost all real-time continuous operation simulation is based on system with known dynamics model. Although Danielsson (2002) is able to construct a real time simulation without having to ascertain the system's dynamics model, his work is based only on a single type of controller. If another type of controller is to be used, an emulator for that particular controller has to be developed.

iii. In most cases, such as AutoMod, the model built for simulation purposes is not usable for an emulation environment (McGreogor 2000). The user has to reconstruct or modify a specific model for emulation application, thus leading to redundancy.

iv. As a truly open architecture in the domain of machine control does not yet exist, all commercial simulation software which support real time environments cannot provide support all types of controller. They either support certain controllers only or the controller manufacturer has to provide their device information before it can be used with the simulation software. With the later always resulting in conflicts of interest between the parties involved.

3.5 SUMMARY

In this chapter various techniques for bridging the real and simulation world have been discussed. The techniques are divided into seven different classes. The techniques range from total separation between the two environments to the use of various emulation techniques for bridging the two. The benefits of using emulation techniques have also been discussed. The chapter ends with a list of issues which motivated this research study.

The design of a satisfactory system architecture is a keystone for making efficient use of advanced simulation tools which accelerate the process of building of industrial machines. Chapter 4 presents the bridging technique adopted as the foundation to realisation of the Real-Virtuality system. This is followed by a proposal for Real-Virtuality architecture.

CHAPTER 4

A REFERENCE ARCHITECTURE FOR A REAL - VIRTUALITY SYSTEM

4.1 INTRODUCTION

The term 'architecture' is widely used in various disciplines involving system development. A system architecture determines the nature or essence of a system. An architecture of a specific system or product is the result of a design process. Architectural specifications do not show all the details necessary for the implementation of the conceived system in reality.

This chapter will define the concept and propose a reference architecture for the Real-Virtuality system. A set of specifications that describe the system components, their interfaces, and their behaviour are introduced.

4.2 THE REAL-VIRTUALITY SYSTEM CONCEPT

The uniqueness of the research comes from the integration of real and virtual environments to form a 'real-virtuality' system for control system developers and machine builders in the context of machine control, which typically features stringent real-time requirements in contrast to process control. The research centred on how a virtual system can be constructed in such it can be used as a real system and operating in the real-time frame. With such capabilities, the real control system can be integrated into the virtual system to form a system which operates realistically but only virtually exists.

The Real-Virtuality (R-V) system focuses on machine level application which involves both discrete and continuous operations. Since the study involves bridging the real controller (real) with machine simulation (virtual), a technique to bridge between the two has to be addressed. In this research study, the bridging technique which is based on signal emulation was utilised. Signal emulation involved both discrete signal emulation and continuous signal emulation. Figure 4.1 shows the proposed bridging technique.

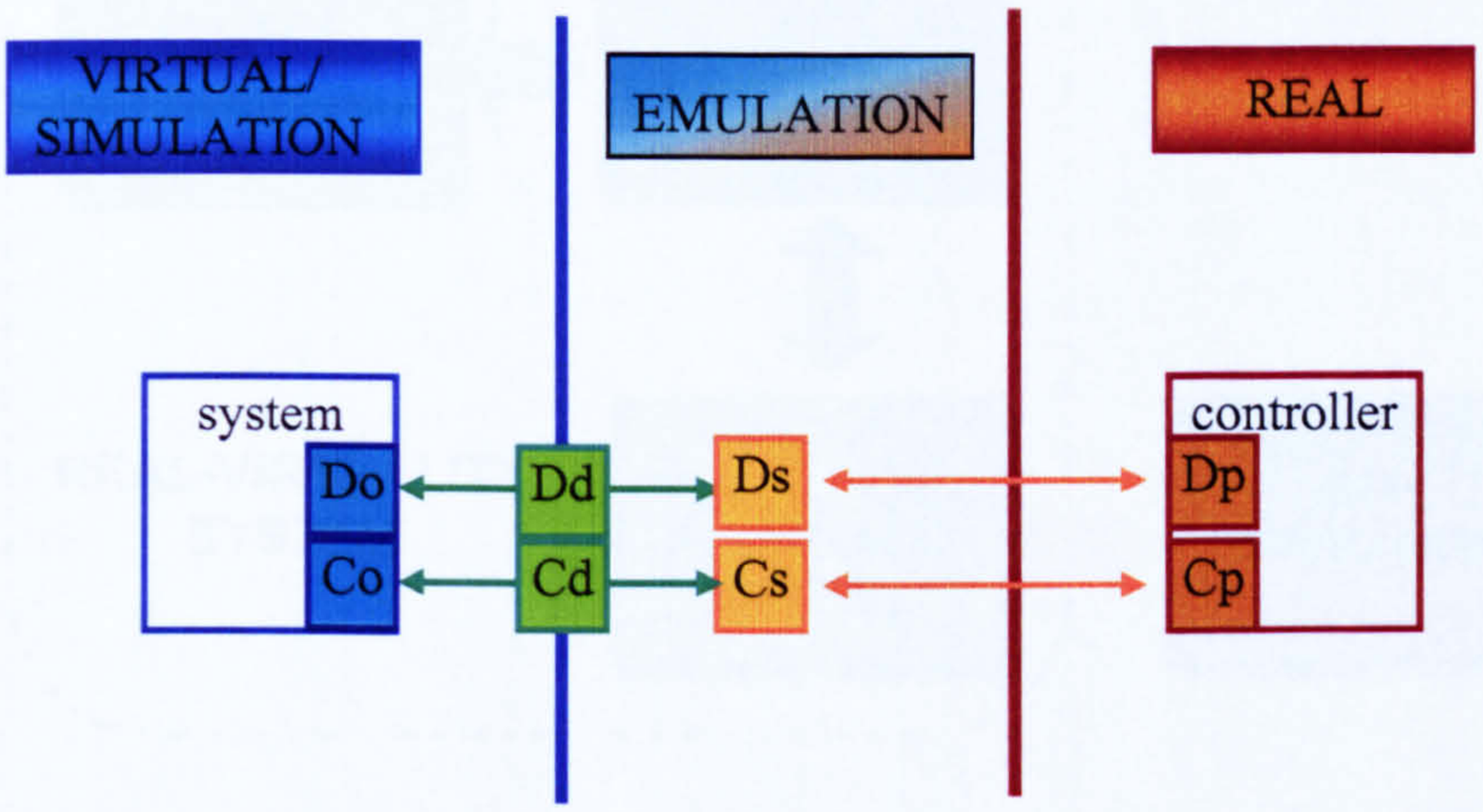


Figure 4.1 Bridging the real world and the virtual world (Do=discrete operation, Co=continuous operation, Dd=discrete data, Cd=continuous data, Ds= discrete signal, Cs=Continuous signal, Dp=discrete program, Cp=continuous program)

The R-V system is designed to create a highly integrated design, simulation, verification and validation environment for realising virtual testing, commissioning and training of a system. In the integrated R-V system the dynamics of the machine system is not a concern. What are of concern are the real-time interactions between the target machine

system to be controlled and the control system. In this case, both the target machine system and the associated control system are of interest. The primary concern is the behaviour and controllability of the target machine or system, not the control system.

The ultimate objective was to realise machine system solutions with the receptivity, rapidity and agility required by control system developers and machine builders. The basic Real-Virtuality system architecture is shown in Figure 4.2.

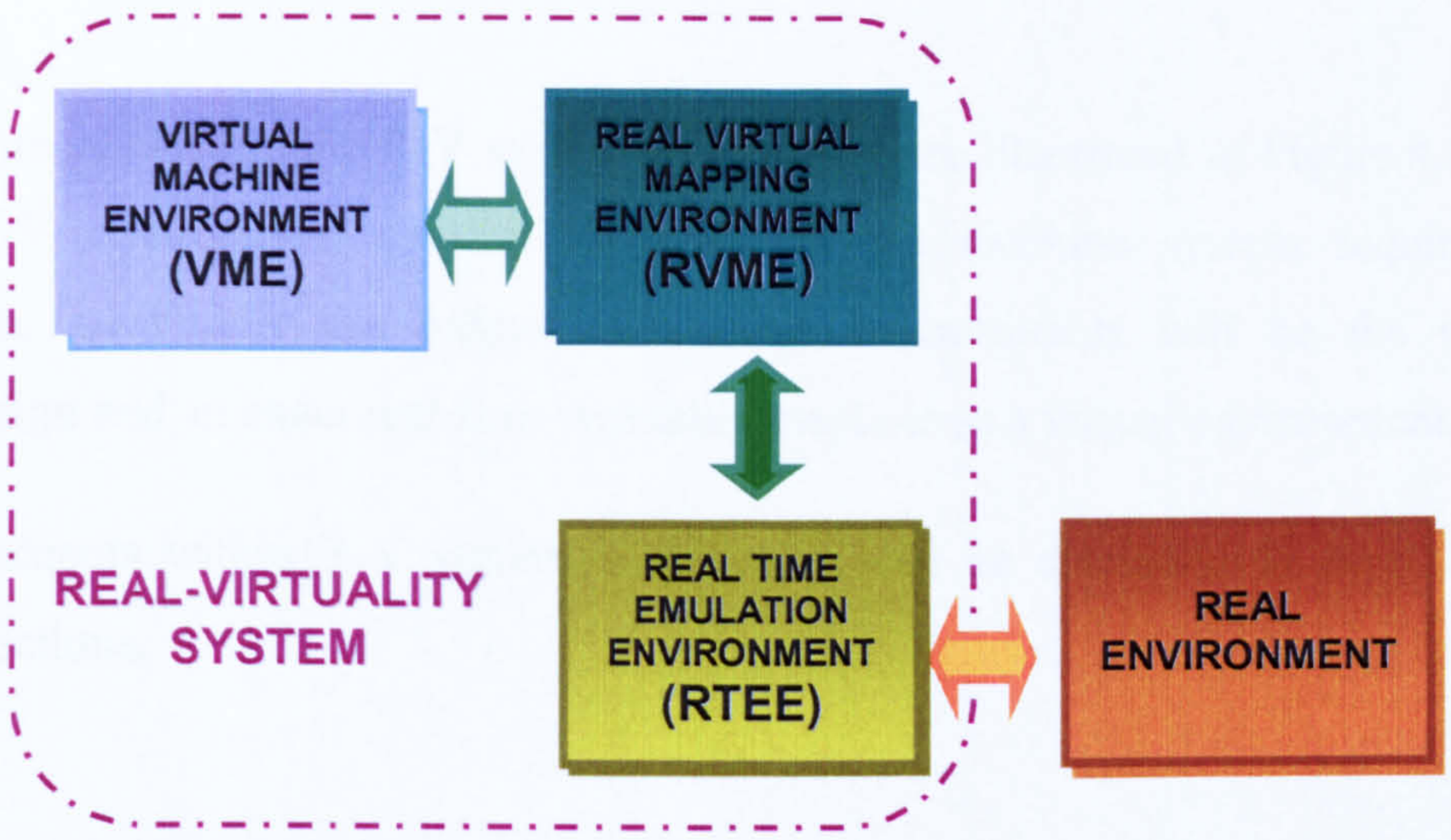


Figure 4.2 The Real-Virtuality system architecture

4.3 REAL-VIRTUALITY SYSTEM ENVIRONMENT

By integrating the real and virtual environments, the 'real-virtuality' system architecture for control system developers and machine builders will consist of three interrelated environments which are:

a) The Virtual Machine Environment (VME), the environment where the target machine/system to be controlled is simulated

(b) The Real-Time Emulation Environment (RTEE), the environment which emulates the real machine/system behaviour.

(c) The Real-Virtual Mapping Environment (RVME), the environment which provides an interface between the real and simulation world.

The environments within the R-V system architecture are illustrated in Figure 4.3. The input to R-V environment can be perceived as the machine system requirement specification, meanwhile the output from these environments will be the proven machine design and an exact real-time workable machine in a virtual environment.

The environments within R-V system architecture will be discussed in detail in the following sections.

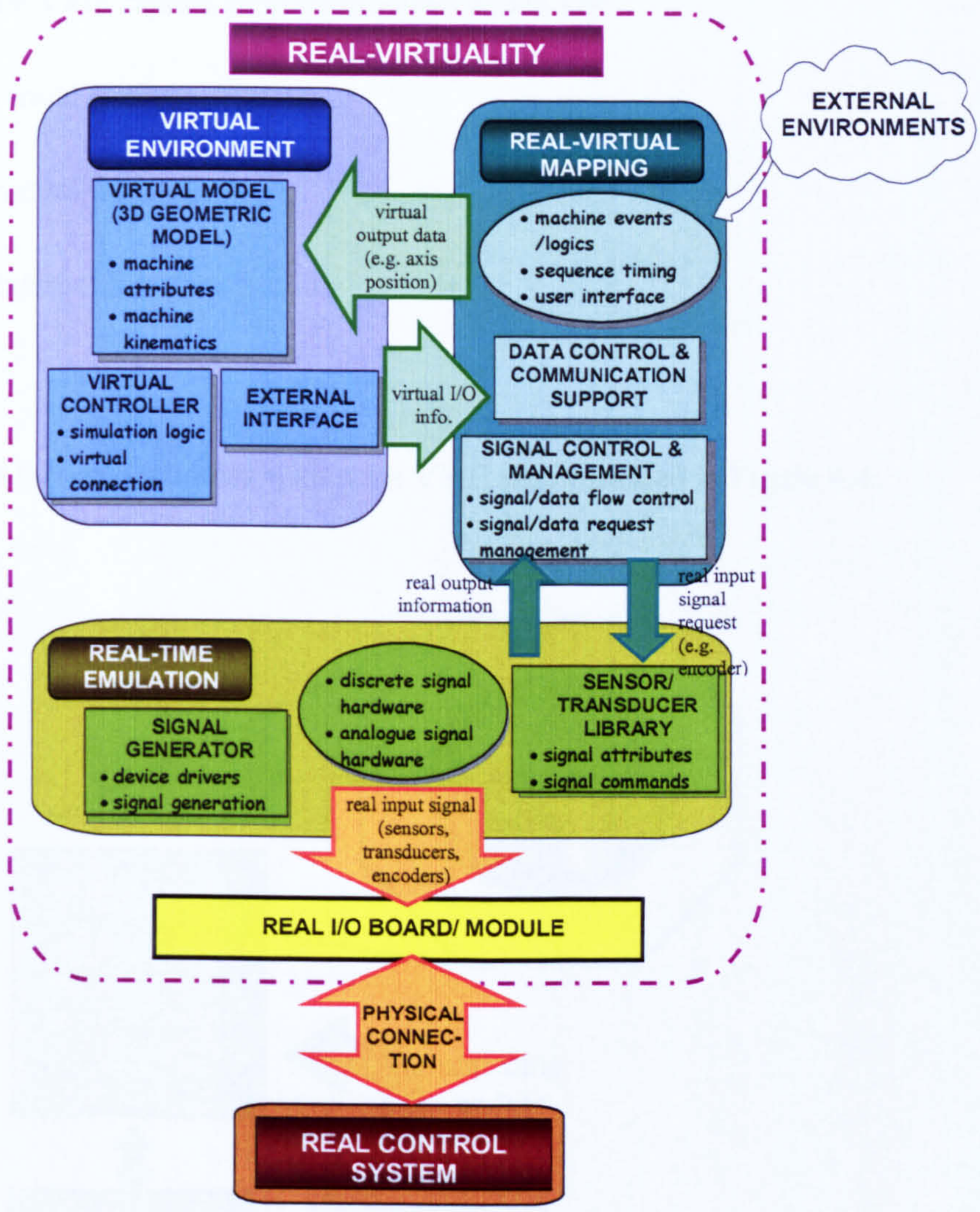


Figure 4.3 The environments within Real-Virtuality system architecture

4.3.1 The Virtual Machine Environment (VME)

The Virtual Machine Environment (VME) is the design environment of the target machine system. This is where the designers construct their machine model. Within the Virtual Environment (VME), the targeted machine or system to be controlled is

modelled. The VME consist of three modules which are:

i. the Virtual Model

ii. the Virtual Controller

iii. the External Interface Module

The relations between modules within the VME are illustrated in Figure 4.4.

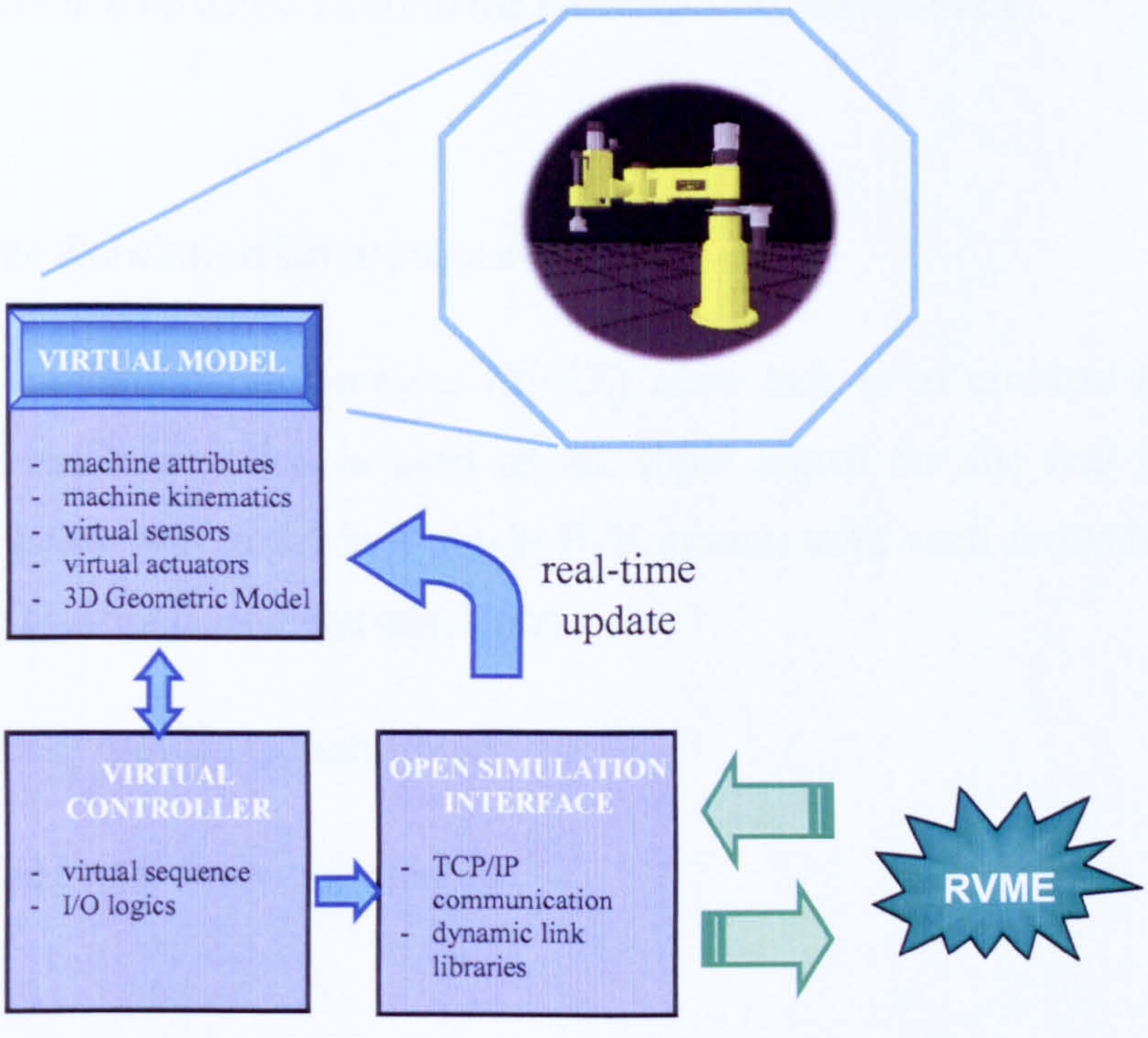


Figure 4.4 The relation between VME modules

The Virtual Model is a 3D geometric model consists of all the machine parts, virtual sensors and virtual actuators. It also has all the machine attributes and kinematics. The

External Interface Module provides facilities to allow data flow in and out of the VME. Virtual Model activities are monitored by the virtual sensors within the environment. Each virtual sensor is connected virtually to the Virtual Controller ports. Thus, the I/O ports values change as the sensors change their state. The Virtual Controller information are read or sent out as required. This is done via the External Interface Module. Real-time data (e.g. axis positions) are continuously pumped into the VME which continuously changes the Virtual Model appearance. Hence, provide real-time actuators motions.

The machine model can also be used for the normal traditional simulation. In such situation the model can be detached from the External Interface Module.

4.3.2 Real-Time Emulation Environment (RTEE)

The Real-Time Emulation Environment (RTEE) main task is to emulate the output signal from the machine which is used as the input signal for the real controller. Multiple RTEE block may exist in a single R-V system with each individual RTEE block consists of two main modules which are:

i. the sensor/transducer library

ii. the signal generator module

R-V Command is used for requesting emulated signals from RTEE. R-V Command is received via the RTEE server. The signal request is processed by each individual sensor/transducer control program. There is a set of sensor and transducer library which is able to provide all type of sensor and transducer signals; and control the signals output. Based on the type of signal and its attribute, the required signal is generated by a signal generator module with the aid of the respective hardware device driver. The information regarding the generated signal is sent back to the client (RVME)

continuously. The relation between all the modules within the RTEE is illustrated in Figure 4.5.

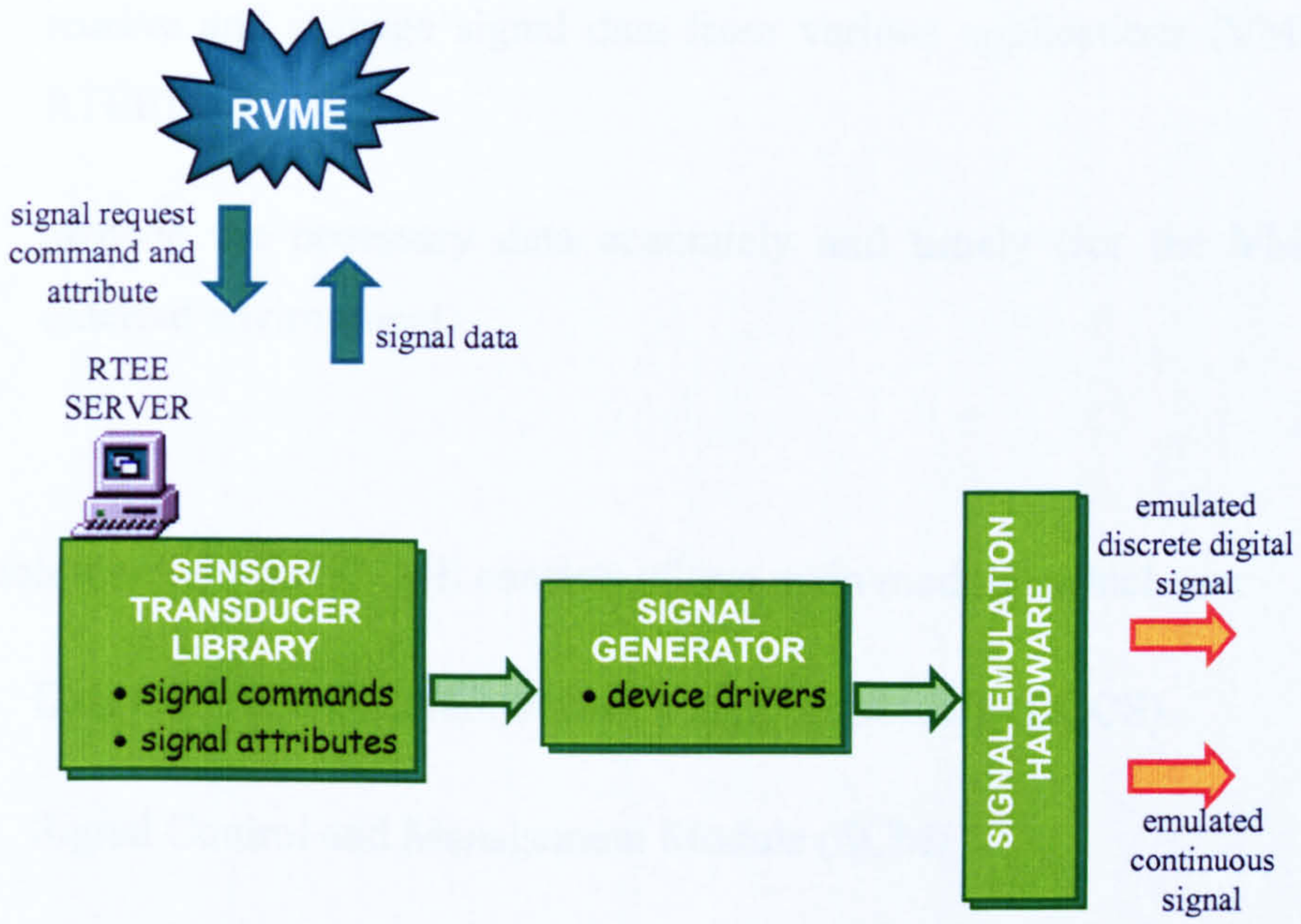


Figure 4.5 The relation between RTEE modules

4.3.3 The Real-Virtual Mapping Environment (RVME)

The Real-Virtual Mapping Environment (RVME) is the core of the integrated real-virtual system. RVME correlates the Virtual Machine Environment (VME), the Real-Time Emulation Environment (RTEE) and the real control system. It handles and manages the virtual and real information required and received from both VME and RTEE so that the real controller can operate as if there is a real machine in operation. The functions of RVME can be simplified as follows:

i. receive and manage signal request commands from various applications (VME, controller, external environment)

ii channel the necessary signal requests to corresponding RTEE module so that the respective emulation signal can be generated accurately and timely

iii receive and manage signal data from various applications (VME, and RTEE)

iv. provide the necessary data accurately and timely (for the VME and external environment)

To provide such functions the RVME consists of two main modules which are:

i. Data Control and Communication Support Module (DCCS)

ii. Signal Control and Management Module (SCM)

Via these two modules RVME manages and controls all signal requests and data flow which makes interaction between real and virtual worlds possible. Figure 4.6 shows the information in and out from the RVME. The DCCS module task is to receive all data from VME and RTEE and store in the appropriate database. It also at the same time manages the connection of all environments to RVME. Data and signal manipulation within the R-V system is based on *last-connect-first-serve* order. The order implies that the last environment which connects to RVME is given highest priority in data manipulations and signal requests. This is because the RVME also have the capability of allowing external data manipulation which can be accessed through its External Communication Module. With this *last-connect-first-serve* order, data can be overridden and interrupted externally whenever necessary and required.

The SCM module task is to manage and control all signal request commands. The SCM module will helps to decide which RTEE block will be instructed to execute a particular

R-V command. It also decides which environment will be given the priority to change a particular signal based on the priority level set by RVME.

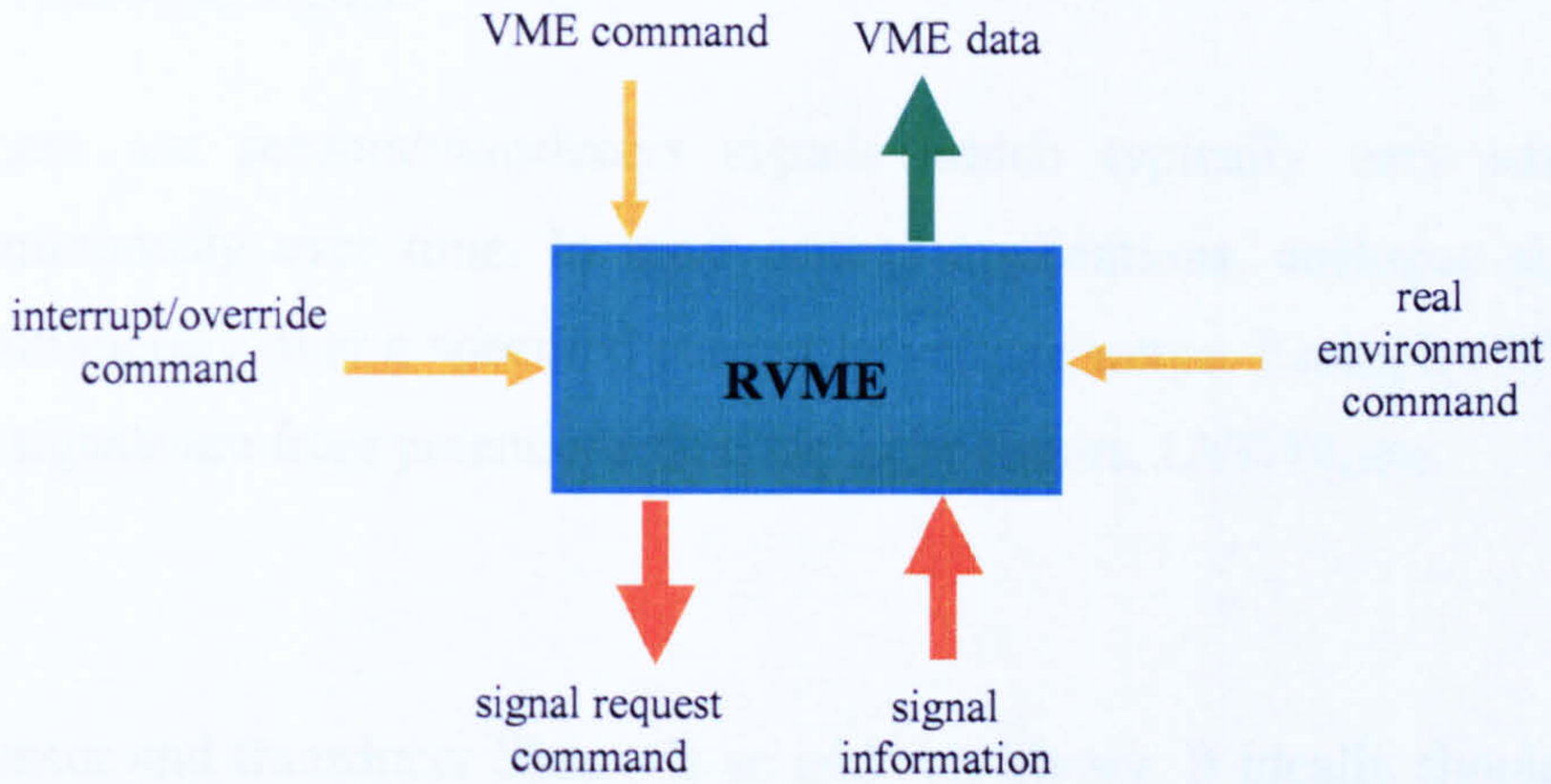


Figure 4.6 Data and command flow in and out of the RVME environment

4.3.4 RVME Signals Classification

As mention in Chapter 2, the physical system can be classified into either continuous or discrete or a blend of both. The R-V system provides support to both continuous and discrete systems. Thus, it should able to accommodate all types of sensor and transducer signals. Sensor and transducer signals within the RVM are classified into three categories. They are:

a. Discrete Digital Signal

These are signals from sensors which are normally used for discrete events. They only have two states; ON and OFF or HIGH and LOW. Examples of this type of signal are from limit switches, proximity sensors, push buttons, etc.

b. Continuous Digital Signal

These are signals from sensors which have two states HIGH and LOW but are produced continuously. Examples of these types of signals are from incremental encoders, absolute encoders, etc.

c. Analogue Signal

These are sensors/transducers signals which typically vary smoothly and continuously over time. In most control applications, analogue signals range continuously over a specified current or voltage range. Examples of these types of signals are from potentiometers, tachogenerators, LVDTs, etc.

RVME sensor and transducer library is an adds-on library. It ideally should be able to accommodate all type of sensors and transducers available in the market.

4.4 R-V SYSTEM COMMUNICATION

The R-V system architecture is based on control signal emulation which makes it independent of any controller hardware. Since the R-V system does not need to address the controller hardware communication standards (propriety standard), the choice of communication protocol within the R-V environment is quite flexible.

Data exchange between all elements of the R-V system is based on protocols set on top of TCP/IP. Using the TCP/IP standard allows various parts of the R-V system to run on different computers or even on different operating systems. TCP/IP is fast, cost effective and reliable. It is multilingual and can act as a carrier for many other protocols.

In general, the TCP/IP protocol can be divided into five layers which are:

i. Physical Layer - deals with pure hardware (e.g. wires, satellite links, network interface cards, etc.).

ii. Device Layer - is a set of software protocols which deals with access methods such as Ethernet, PPP (point-to-point), etc. Ethernet is preferable at this level.

iii. Internet Layer – internet layer also known as Internet Protocol (IP). This layer is responsible for the routing and delivery of data across networks. It ties all the device layers together.

iv. Transport Layer - provides an application layer delivery service. There are two protocols available at the transport layer which is TCP (Transmission Control Protocol) and UDP (User Datagram Protocol).

v. Application Layer – consist of applications and presentation which provides access to the communication environment.

The TCP/IP name is taken from two of the fundamental protocols in the collection, IP at Internet Layer and TCP at Transport Layer.

4.4.1 TCP/IP Programming

TCP/IP programming is about network connection between two computers; client and server. In the architecture, TCP/IP communication uses the very basic level known as *socket*. A socket is an abstraction that represents an endpoint of communication. The operations that can be performed on a socket include control operations (such as associating a port number with the socket, initiating or accepting a connection on the socket, or destroying the socket), data transfer operations (such as writing data through the socket to some other application, or reading data from some other application through the socket) and status operations (such as finding the IP address associated with the socket).

Sockets are standard across almost all operating systems. Hence, it makes the R-V architecture flexible enough to work on any platform. Figure 4.7 shows the typical R-V system environments connection via TCP/IP.

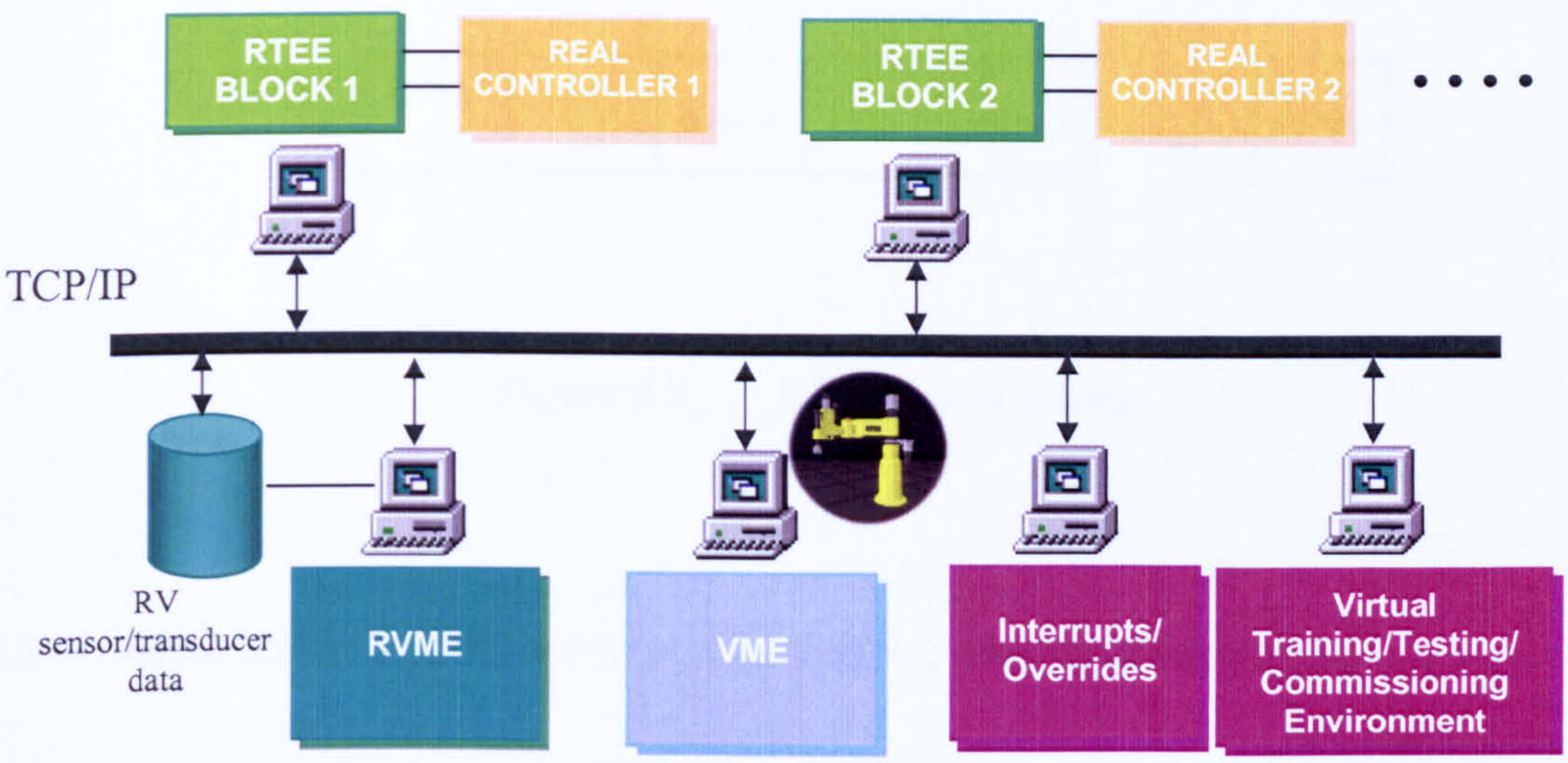


Figure 4.7 R-V environments connected via TCP/IP

4.4.2 R-V Command Data Format

The R-V commands are meant for client signal manipulation of discrete digital signal and continuous digital signal. The data are sent through TCP/IP socket and has to follow a general data format. The R-V commands data have three elements which are:

i. block number - determine which RTEE block

ii. command ID - command identification (to identify which signal is to be emulated and/or the type of operation involved)

iii. signal data - a set of data with information on how RTEE should produce the signal/s

The general format of R-V Command is shown in Figure 4.8 below.

Figure 4.8 RVME data format

4.5 MACHINE DESIGN PROCESS USING THE R-V SYSTEM

In order to use the R-V system efficiently guidelines on its use are required. This section proposes the design process for machine design using an R-V system. The design process is a design methodology which provides a reference for the use of R-V system in building machine systems.

Machine design involves a number òf parties to perform a series of activities. People involved including design engineers, system engineers, control engineers etc. Theoretically, R-V system environments and the associated design process forms a framework for integrated design, simulation and control verification and validation of machine systems. Figure 4.9 illustrates the design process using an R-V environment. Some important features within the design process can be summarised as follows:

i. rectangular boxes represent activities in the design process. Coloured boxes are activities supported by the R-V system

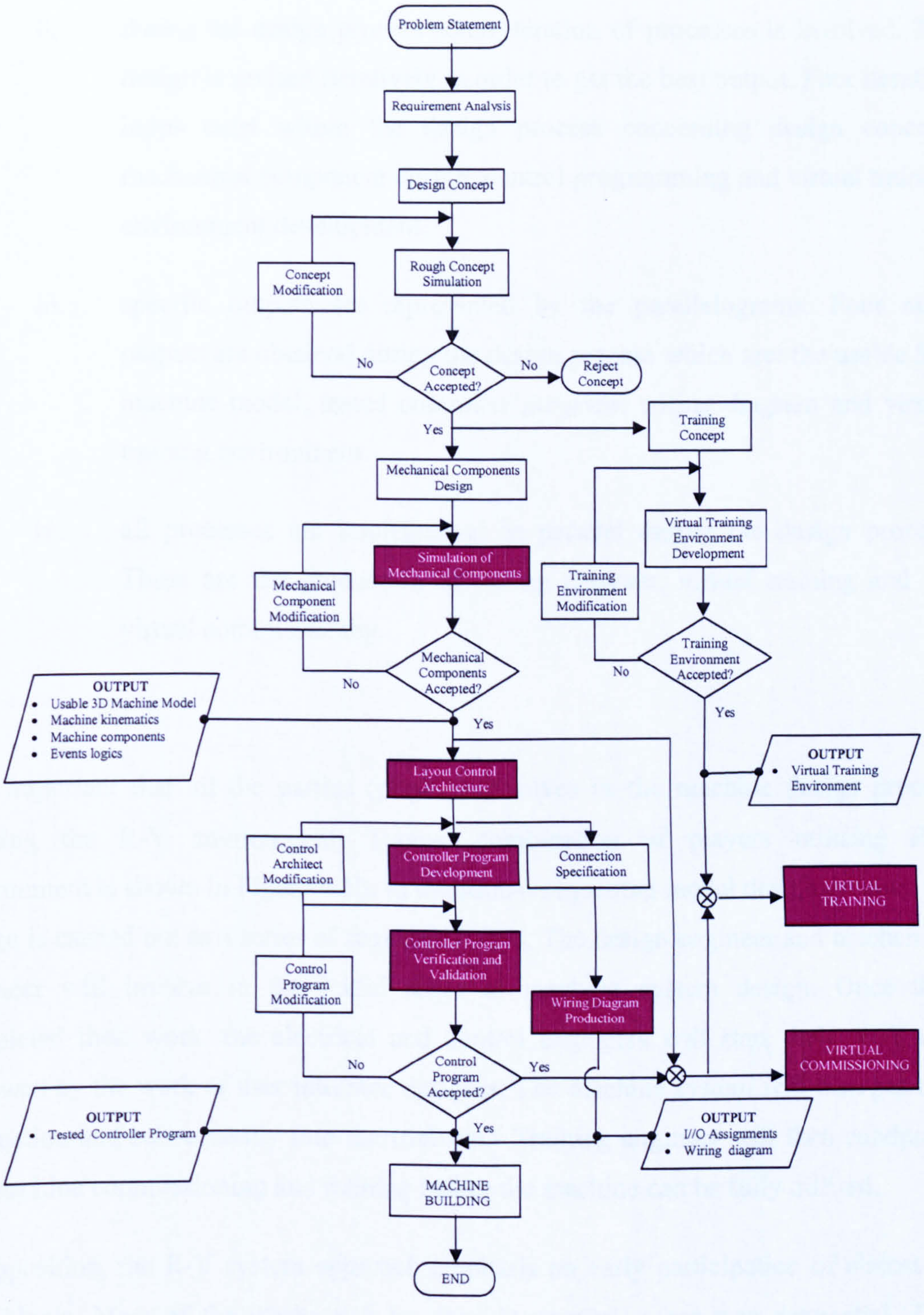


Figure 4.9 The machine design process using R-V

ii. during the design process some iteration of processes is involved. The design is revised iteratively in order to get the best output. Four iteration loops exist within the design process concerning design concept, mechanical component design, control programming and virtual training environment development.

iii. specific outputs are represented by the parallelograms. Four main outputs are obtained during the design process which are: the usable 3-D machine model, tested controller program, wiring diagram and virtual training environment.

iv. all processes are implemented in parallel during the design process. These are the production of wiring diagram, virtual training and the virtual commissioning.

It is important that all the parties (players) involves in the machine design process utilising the R-V environment. Typical combination of players utilising R-V environment is shown in Figure 4.10. In traditional sequential model design process, the design is carried out as a series of sequential steps. The design engineer and mechanical engineer will involve in the initial stage of machine system design. Once they completed their work, the electrical and control engineers will start their work and followed by the work of user interface designer. The machine system will then put into production and subsequently into the trial run. Training engineer will then conducted the machine commissioning and training before the machine can be fully utilised.

In opposition, the R-V system approach emphasis on early participation of almost all individuals. Most of the works will be done in parallel rather then sequential. This concurrent approach helps to shorten the lead time. The approach also allows problems to be envisaged at the early design stage. Eventually, beside the machine designers and engineers; the project manager, marketing team and even the customer can also play a

vital role in the machine building process. Although they might not directly involve with the R-V system they can provide good review on the system before it goes into production. This is because with a completed machine system in R-V environments, the machine routine and performance can be visualised and verified.

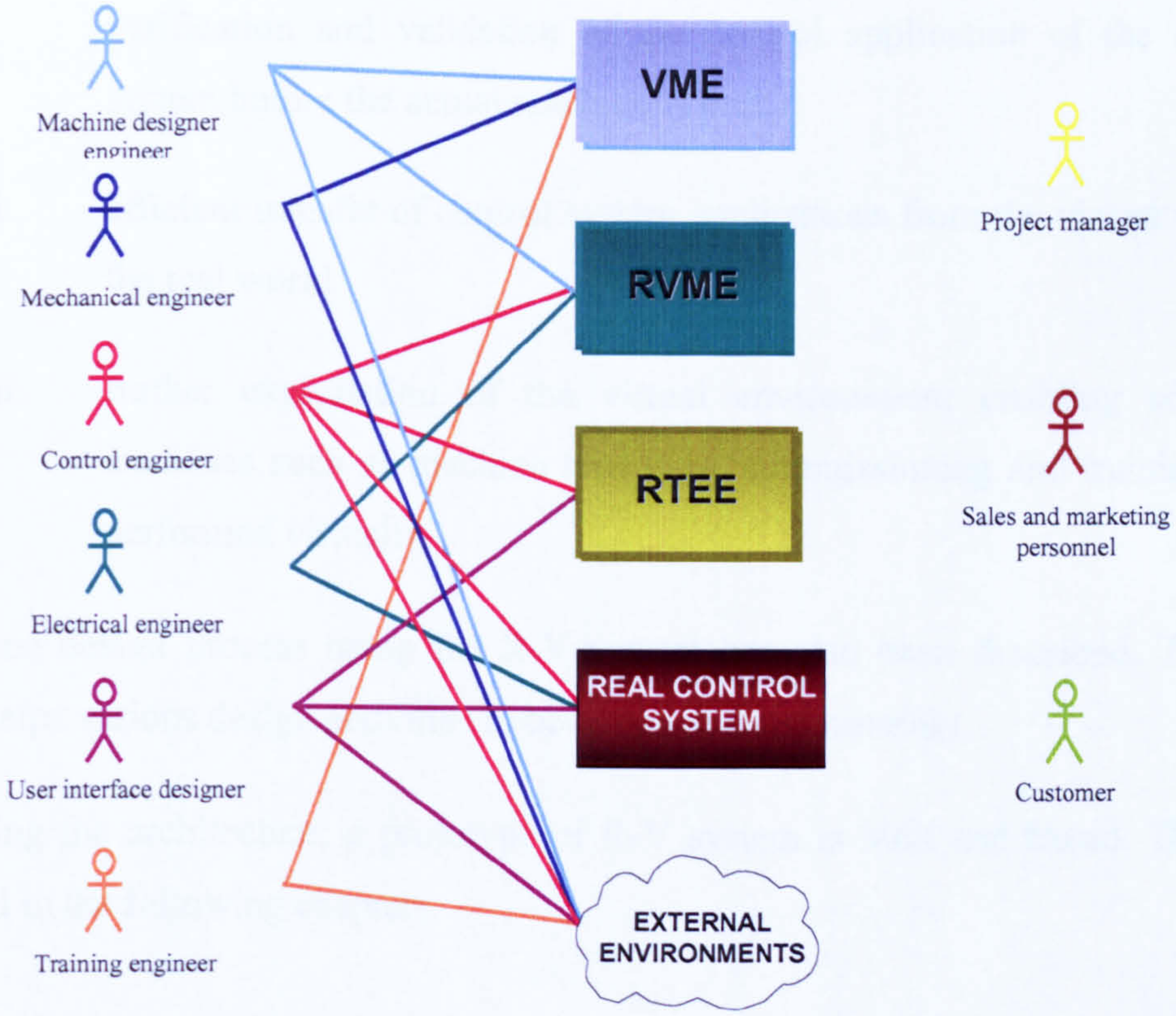


Figure 4.10 The players within the R-V system environments

4.6 SUMMARY

A proposed architecture and a concept of a Real-Virtuality system are presented in some detail in this chapter. The architecture is general in nature and can be executed in any

platform. The architecture is designed in such a way that several machine design activities can be performed concurrently.

The R-V system architecture tries to advance virtual engineering in the context of machine system design and control with the emphasis on:

i. verification and validation of the control application of the machine system before the actual machine is built

ii efficient transfer of control system applications from the virtual world to the real world

iii further exploitation of the virtual environment; enabling additional exercises such as machine trial runs, commissioning and training to be performed virtually.

A machine design process using the R-V system has also been described. The R-V system helps various design activities to be conducted concurrently.

In realising the architecture, a prototype of R-V system is built and tested. These are discussed in the following chapters.

CHAPTER 5

A REAL-VIRTUALITY SYSTEM PROTOTYPE

5.1 INTRODUCTION

A demonstrator system was built to illustrate the use of a Real Virtuality (R-V) system. An R-V system was constructed based on the architecture presented in the previous chapter. The layout of the demonstrator is shown in Figure 5.1 below. The system has five elements, three of which are the main elements of the R-V system. The demonstrator elements are as follows:

i. Virtual Machine Environment (VME)
ii. Real-Time Emulation Environment (RTEE)
iii. Real-Virtual (R-V) Mapping Environment (RVME)

} R-V system prototype

iv. The real controller
v. External Interrupt/Override

} external environment

This chapter will discuss the construction of the R-V system prototype. A modular approach was used during the formulation of the R-V system prototype. All R-V subsystems use client-server Ethernet connection based on TCP/IP protocol for communication. The reason this protocol is chosen is because of its reliability and it provides very fast data transfer. Furthermore, the Ethernet-based type protocol is currently widely accepted in industrial applications (Caro 2006).

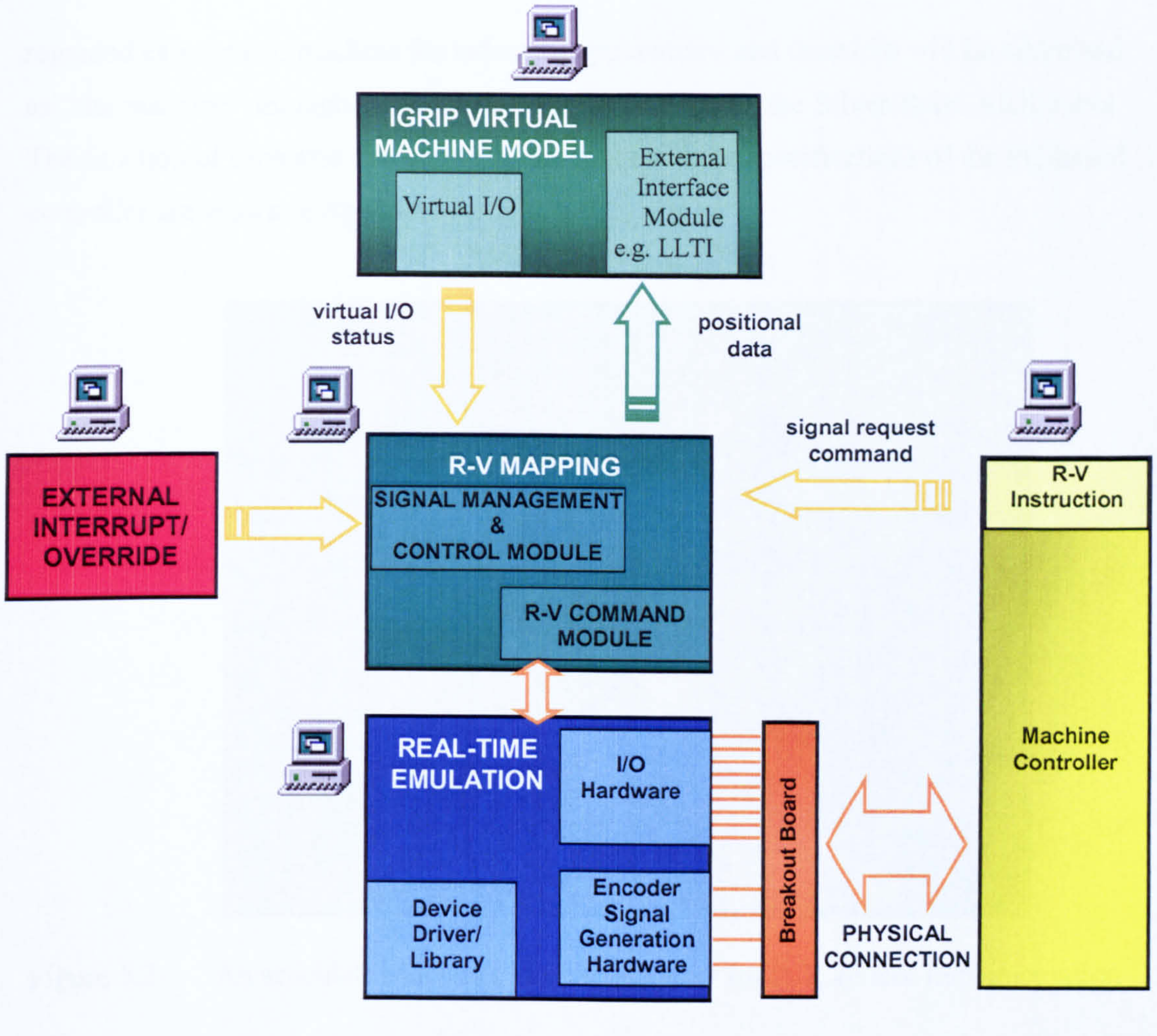


Figure 5.1 The layout of R-V system demonstrator

5.1.1 The Machine

For demonstration purposes, a six degree of freedom (DOF) Silver Reed ARX articulate robot which is available within the Intelligent Machines and Automation Systems (IMAS) laboratory at De Montfort University (DMU) was chosen as a test platform. The robot consists of four moving axes and nine sensors, and represents a machine with mixed continuous and discrete operations. The controller used for the study is a NextMove; a commercial PC-based industrial controller produced by Baldor Corporation. It should be noted that throughout the building of the system, the robot is

regarded as a generic machine for industrial applications and therefore will be addressed as "the machine" throughout the thesis. Figure 5.2 shows the Silver Reed ARX robot. The function of each axis and sensor of the robot and the specifications of the PC-based controller are shown in Appendix A.

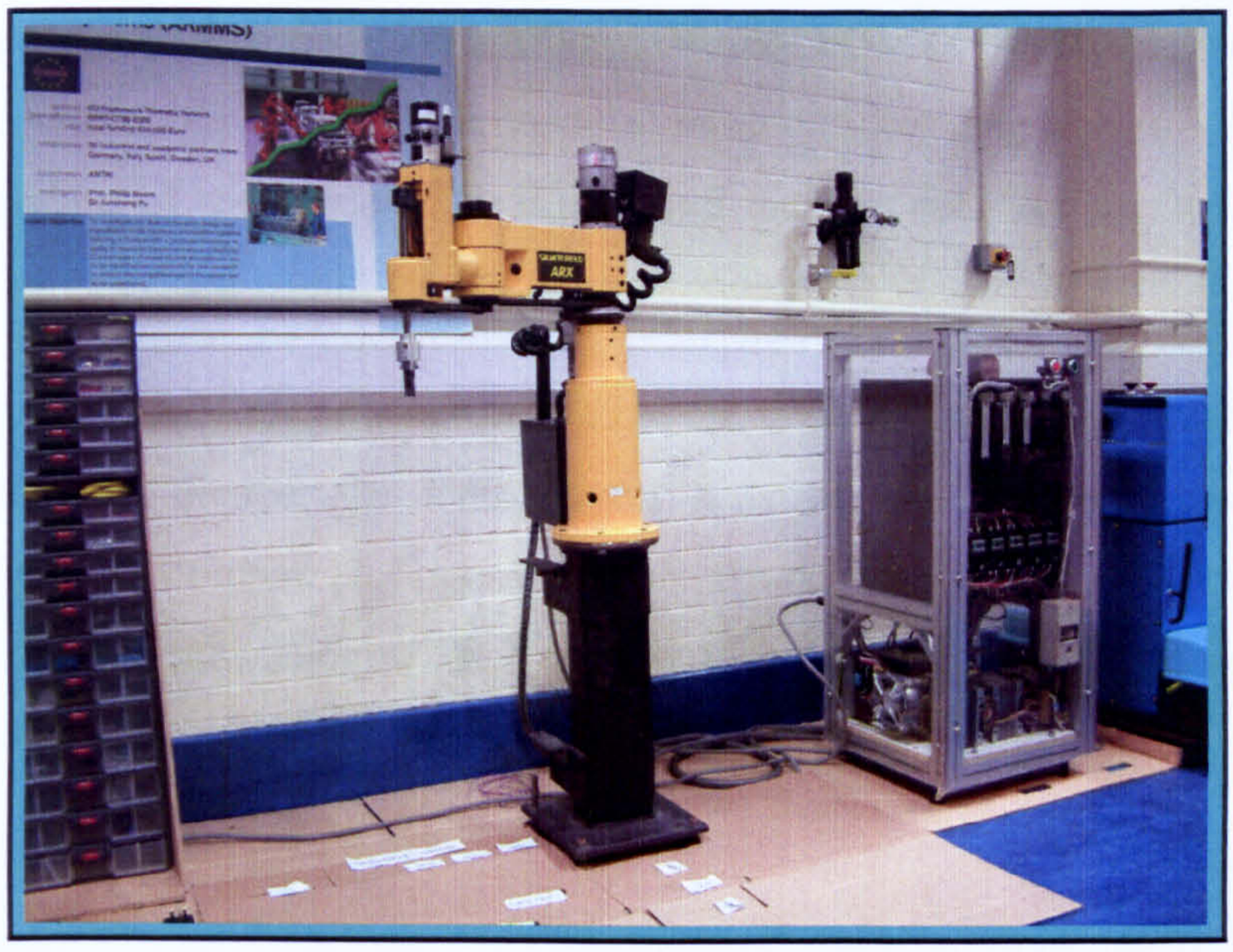


Figure 5.2 An articulate robot as a *generic machine* for R-V system implementation

5.2 DEVELOPMENT OF VIRTUAL MACHINE ENVIRONMENT

In the demonstrator, the 3-D model of the target machine was constructed using a 3-D simulation package called IGRIP (Interactive Graphics Robot Instruction Program) from Delmia Corporation. IGRIP is scalable robotic simulation software for modelling and off-line programming complex robotic workcells. Actual machine geometry, motion attributes, kinematics, and I/O logic can be incorporated to produce extremely accurate simulations. Although IGRIP is meant for 3-D robotic simulation, it can also be used for mechanical simulation of any type of machine.

Cheng (2000) provides an overview of how a system is modelled within the IGRIP environment. He briefly describes how to build the machine model, assigning its DOF, presenting its kinematics, assigning the device I/O and programming the device behaviour using the software programming language. In IGRIP a machine is regarded as a workcell consist of various devices. Devices are constructed from parts and can be assigned with specific DOF and motion attributes. Virtual I/Os are used to represent I/O communications within the workcell. Procedural programming language is utilised to control the behaviour of the model. The model built can be used for various offline analyses which are provided within the IGRIP environment.

The target machine for R-V usage is constructed in the same way as models constructed for offline simulation. Hence, the methods for constructing a machine model as described by Cheng are comparable. In other words, the constructed model can be used for both, normal offline simulation and R-V applications. Figure 5.3 shows the machine (ARX robot) model built in IGRIP.

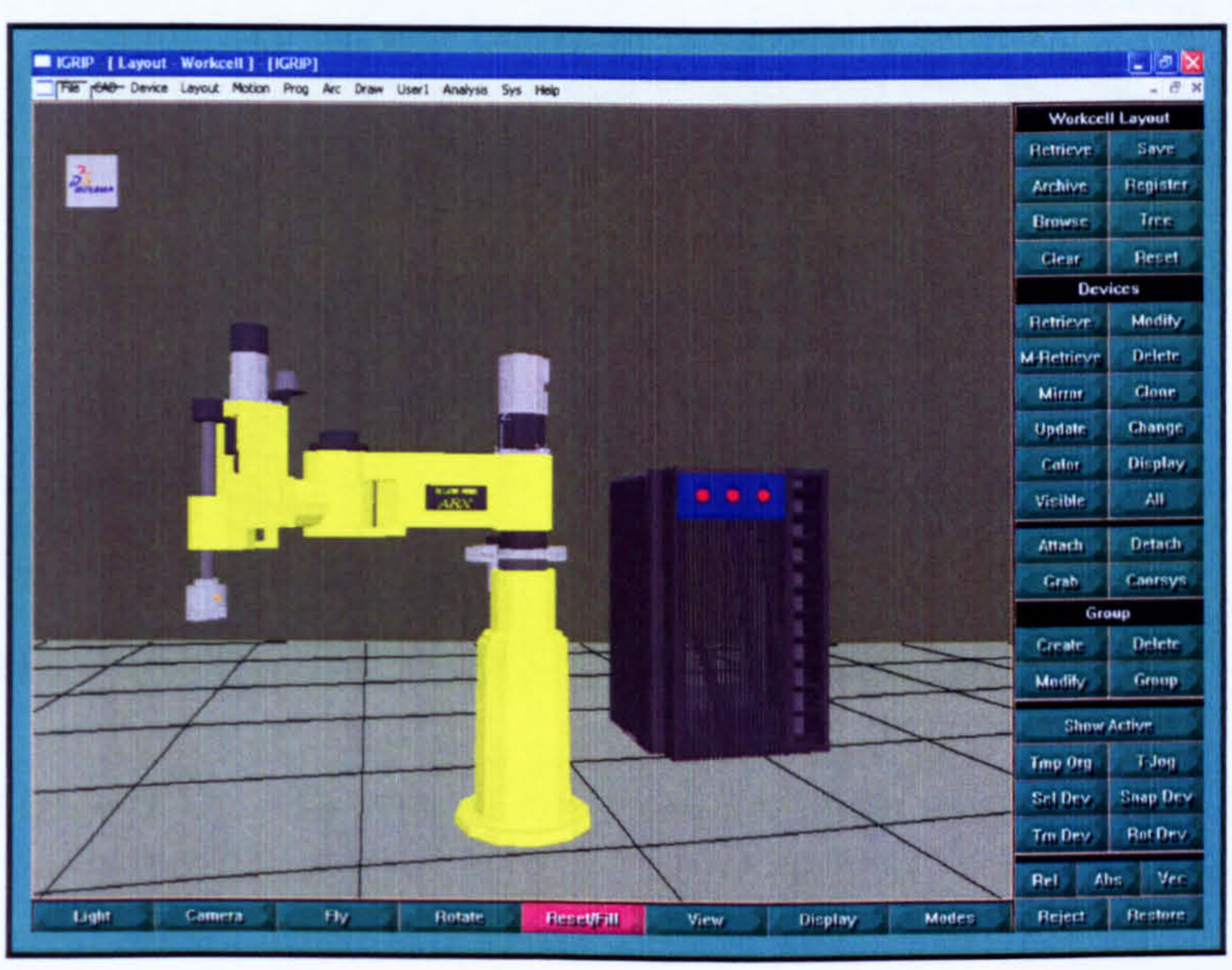


Figure 5.3 The machine model built in IGRIP

For the R-V applications, the machine model has to have the capability to communicate with external environments. IGRIP is an open simulation software which facilitates interaction with components outside its environment. This is achieved via IGRIP Low Level Telerobotics Interface (LLTI) module. IGRIP LLTI is discussed in Section 5.2.2.

5.2.1 Virtual Sensors and Virtual Controller

The virtual sensors are constructed in a simple way which is based on the position of certain moving joints of the model. Each joint is configured with its forward and reverse limits. The configuration of each joint is shown in Figure 5.4

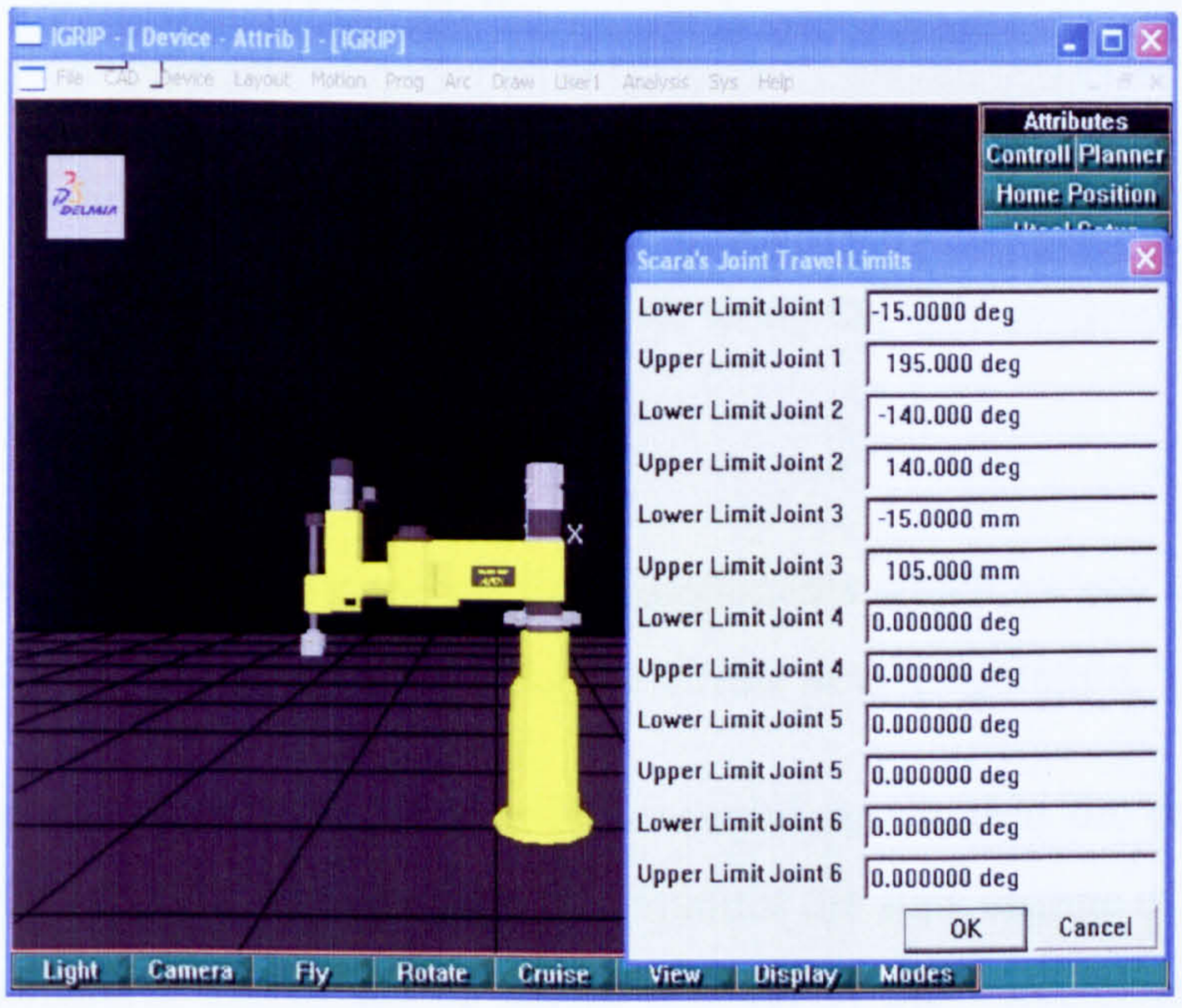


Figure 5.4 The virtual machine joint limits configuration

The construction of the virtual sensor is implemented through IGRIP's Graphical Simulation Language (GSL) programming language attached to the virtual machine.

Although Eriksson (1996) has shown that it is possible to simulate sensor behaviour in IGRIP, but his proposed method is not adopted in this research as it is beyond the scope of this study. In this study, the virtual sensors are constructed by providing a ± 1.5° of the axis joint limit. For example, the forward limit switch of Joint 1 is activated when the Joint 1 is at position 193.5° (Joint 1 maximum forward limit is 193.5°)[10]. Typical GSL commands which perform virtual sensor tasks are shown in Figure 5.5:

```
IF (JOINTVAL ( 1) > 193.5) THEN
    DOUT[1]=TRUE

ELSE
    DOUT[1]=FALSE
  ENDIF
```

Figure 5.5 Implementation of virtual sensor using the GSL programming language

The advantage of IGRIP is that the moving components which exceed their joint limits will change colour. This helps in verifying the virtual sensor operation.

IGRIP is incorporated with interdevice I/O connections. Within the demonstrator the IGRIP interdevice I/O connections is used to monitor the state change of virtual sensors. The R-V demonstrator makes use of IGRIP interdevice I/O connections to mimic the digital I/O signal connection in the real environment. The IGRIP's interdevice I/O connections allow devices within the IGRIP environment to be assigned with a set of virtual inputs and outputs. Since the machine model should represent the real machine, all virtual sensors on the machine model are regarded as devices that produces output signals. Hence, the virtual sensors are connected to the output channels of the IGRIP

[10] 195° is the forward limit for Joint 1; a ±1.5° clearance was set within the IGRIP's GSL program to represent a rough behaviour of the forward limit sensor.

interdevice I/O connections. A Virtual Controller which has input and output channels is also constructed in IGRIP. All the virtual sensors outputs are connected to the inputs of the Virtual Controller via its interdevice I/O connections. The state of the Virtual Controller is monitored during R-V execution. Although the use of a Virtual Controller seems redundant[11], but it helps to maintain the machine model compatibility for use in offline simulation. Figure 5.6 and Figure 5.7 show the virtual I/O connection of the virtual machine and the Virtual Controller interdevice I/O connections configuration respectively.

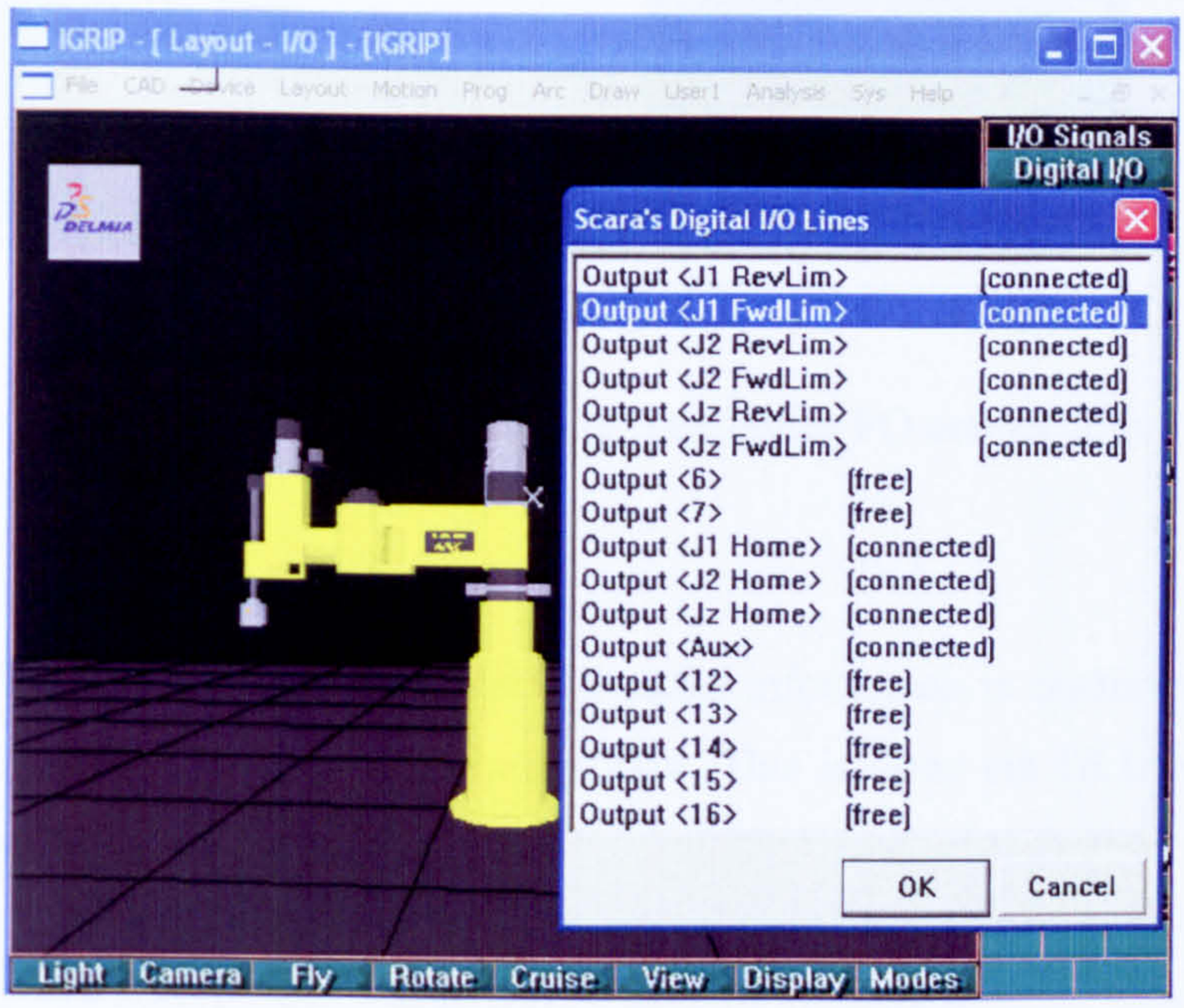


Figure 5.6 The virtual machine sensor interdevice I/O connections

[11] Instead of connecting the virtual sensors to the output of the interdevice I/O connection, they can be connected to the inputs and these states can be monitored during R-V execution though.

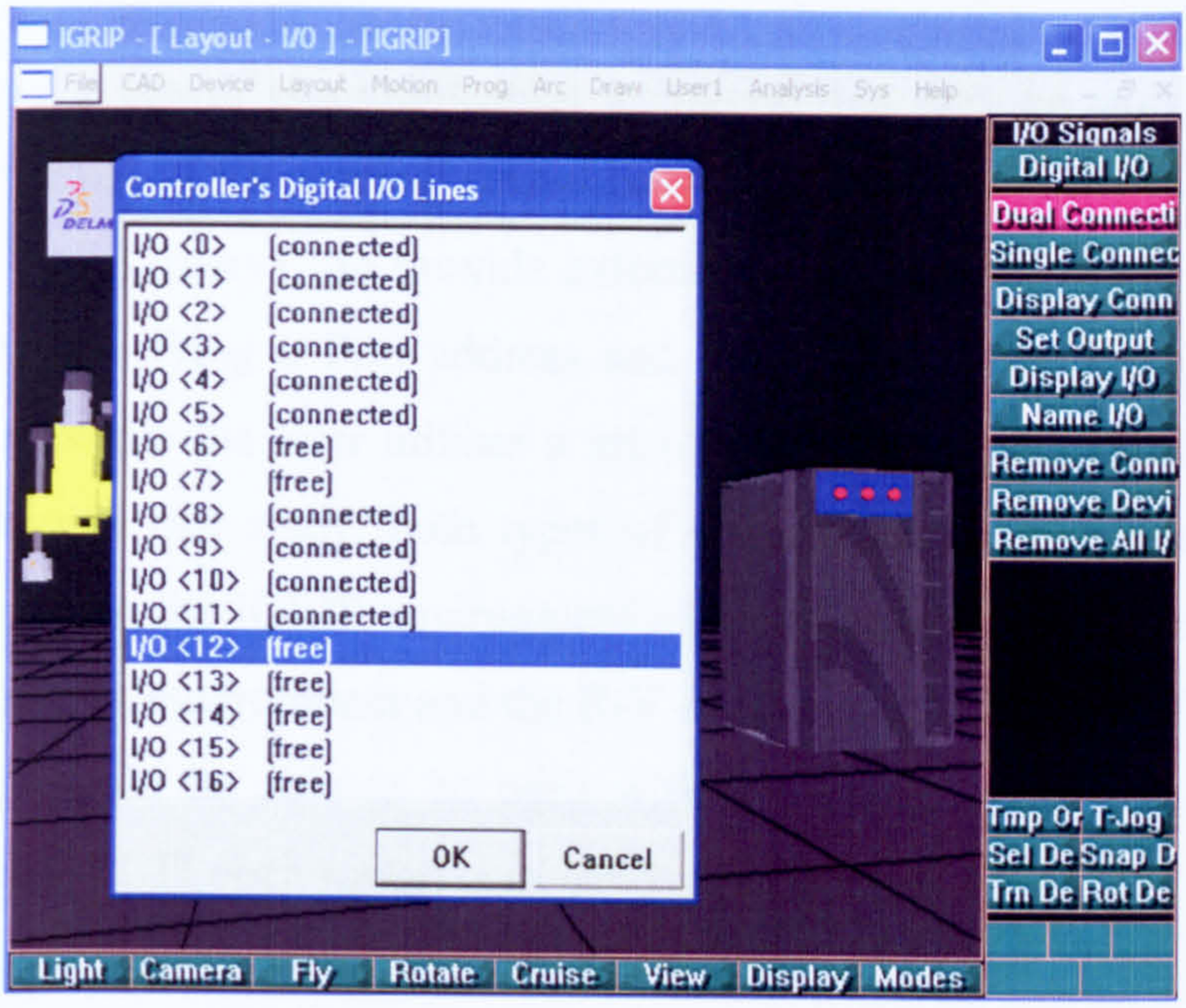


Figure 5.7 The Virtual Controller interdevice I/O connections

During the execution of the simulation, the I/O states information is continuously sent out to the R-V Mapping environment via a socket. This is done via DLL within the IGRIP LLTI module.

5.2.2 External Interface - IGRIP Low Level Telerobotics Interface (LLTI)

The Low Level Telerobotics Interface (LLTI) is fundamentally based on a distributed computing concept. Models in IGRIP can be controlled in such a fashion where the user interface resides in an external program and communicates via sockets to the main system kernel. The interfacing process may be on the same physical machine or on any

networked computer. The interface is low level, and interrupt-driven to ensure both maximum data flow into the system and optimum graphics performance.

There are two types of communication technique that may be applied via LLTI. The user can either use direct external connection to a device through a socket or use a DLL (Dynamics Link Library)[12] to provide external motion control in real time. The use of socket is by specifying a Port address and Hostname which allows connection to a process. Meanwhile the later utilises a set of user I/O routines existing within IGRIP shared library. In this study, both types of communication techniques are used. The DLLs built within the IGRIP environment allow changing of data between the Virtual Model Simulation environment and the R-V Mapping environment. The DLLs built are generic and should be able to be used with any other machine model simulated in IGRIP. Typical LLTI open socket commands are shown in Figure 5.8.

```
void llti_init_controller()
{
/**********>>>>>>> begin execution <<<<<<<**********/
int i;
 portnum = 4000;

   if ( net_init_socket_client( "localhost", portnum, &sockdsc ) != 0 )
   {
     printf(stderr,"Can't connect to server localhost:%d\n",portnum);
   }

    for (i=0; i<21;i++)
            packet[i] = 0.0;

    if (net_write_llti( sockdsc, packet ) != 0 )
   {
    printf("Socket aborted by peer during write.\n");
   }

   return;
}
```

Figure 5.8 An example of LLTI open socket command

[12] In IGRIP, the DLL is also known as Dynamic Shared Library (DSO) which correlates to its use on UNIX platform

The LLTI commands are processed continuously. Each time the system polls the mouse, it also checks all LLTI connections for any activity. If commands are processed, then a graphics update occurs. During a running simulation, each LLTI connection is checked several times per simulation update. If commands are processed, the graphics update is deferred to the next simulation graphics update. This results in minimal impact of LLTI on simulation performance. The maximum LLTI update rate depends on the workstation hardware, but can be as high as 30 Hz.

The data formats to send to and from IGRIP should conform to the LLTI packets (data format understood by IGRIP). IGRIP provides several opcodes which helps easy update of the model. Most of the available LLTI commands are provided in the interest of simple, high speed positioning of a device. Figure 5.9 shows an example use of LLTI routines together with the execution of LLTI packets.The LLTI opcodes and their specifications are presented in Appendix B.

```
DllExport

float *llti_read_controller()

{
int io_read;

    packet[0]=20.0;          //reserve 4 arrays for communication header
    packet[1]=0.0;
    packet[2]=0.0;
    packet[3]=0.0;
    packet[4]=0.0;
    mot_inq_digital_io(0,0,&io_read);        //get all 16 controller  I/O status
    packet[5] = io_read;
    mot_inq_digital_io(0,1,&io_read);
    packet[6] = io_read;
    mot_inq_digital_io(0,2,&io_read);
    packet[7] = io_read;
    mot_inq_digital_io(0,3,&io_read);
    packet[8] = io_read;
    mot_inq_digital_io(0,4,&io_read);
    packet[9] = io_read;
    mot_inq_digital_io(0,5,&io_read);
    packet[10] = io_read;
    mot_inq_digital_io(0,6,&io_read);
```

```
packet[11] = io_read;
mot_inq_digital_io(0,7,&io_read);
packet[12] = io_read;
mot_inq_digital_io(0,8,&io_read);
packet[13] = io_read;
mot_inq_digital_io(0,9,&io_read);
packet[14] = io_read;
mot_inq_digital_io(0,10,&io_read);
packet[15] = io_read;
mot_inq_digital_io(0,11,&io_read);
packet[16] = io_read;
mot_inq_digital_io(0,12,&io_read);
packet[17] = io_read;
mot_inq_digital_io(0,13,&io_read);
packet[18] = io_read;
mot_inq_digital_io(0,14,&io_read);
packet[19] = io_read;
mot_inq_digital_io(0,15,&io_read);
packet[20] = io_read;

if (net_write_llti( sockdsc, packet ) != 0 )
 {
  printf("Socket aborted by peer during write.\n");
 }
return( NULL );
}
```

Figure 5.9 An example use of LLTI routines together with the execution of LLTI packets.

5.3 FORMULATION OF REAL TIME EMULATOR ENVIRONMENT (RTEE)

As discussed in Chapter 4 (see Section 4.3.4) the type of signal existing in machine operation can be divided into three types which are as follows:

i discrete digital signal,

ii continuous analogue signal and

iii continuous digital signals.

Emulation of the first two types of signals is relatively straight forward, the emulation of the third is found to be very challenging. In this study it was decided to focus only on discrete digital signal and continuous digital signals. Continuous digital signals are considered more important than analogue signal for machine applications especially when involving quadrature encoders. Current trends shows that all modern machine uses servo motors which have digital encoders as their feedback element. These digital encoders produce continuous digital signals (this is described in Section 5.3.2). It should be noted that the quadrature encoders are most prevalent in industry because of their ability to provide direction of rotation besides its ruggedness and simple wiring (EPC 2007). Furthermore, at the time of the research[13], there was no hardware yet available to mimic the continuous digital signals behaviour of quadrature encoder in comparison to analogue signals[14]. Thus, the ability to produce one is considered novel and important to prove the feasibility of the R-V system.

5.3.1 Discrete Digital Signal Emulation Hardware

Emulation of discrete digital signal is straight forward. For simplicity, a 16-channel digital output card from National Instrument was used as the emulation hardware. The card is PC-based card capable to produce 5V TTL output to emulate common sensor signals. However, in the event when a higher voltage is needed[15], a relay can be placed at the output as a voltage step-up device.

---

[13] At present there is an encoder simulator available in the market. The product is introduced in October 2006 by Plant Control and Automation Pty. Ltd., an Australian company (see Appendix C). However the simulator can only be operated manually which is not appropriate for R-V applications.

[14] Analogue signals generation can easily be produced using a digital-to-analogue converter which is easy to build and is commercially available. Although clock/counter signal can represents continuous digital signal but it should be noted that quadrature encoder signal not only is a continuous digital signal but should consist of two signals with 90° shift.

[15] In industry some sensors produces 24Voutput voltage rather then 5V.

5.3.2 Continuous Digital Signal Emulation Hardware

For the demonstrator to function, it is important to be able to emulate the behaviour of the quadrature incremental encoder. As the emulated encoder signal produced within the RTEE is being used as the input signal to the real controller in continuous feedback operation, a defect signal can easily trigger an error in the controller and crash the whole system. Thus, the ability to mimic the encoder behaviour is crucial.

In order to be able to produce the encoder signal emulation hardware, the quadrature encoder signal behaviour has to be understood. The following are the characteristic of quadrature encoder signal:

i. the output of an encoder is two continuous 90° shift clock signal, normally named as A and B.

ii. in a clockwise motor movement, the outputs signal B will lead A and alternately, output signal A will leads B in counter clockwise motor movement.

iii. encoder signal is normally in the form of square wave with the frequency that may vary between 0 to 80 MHz.

iv. signal frequencies for both A and B change continuously based on the motor speed. The timing of the frequency change has to be accurate as expected by the controller. Discrepancies between the encoder signal and the expected signal will cause an error to the controller.

In order to achieve the above, the hardware for the encoder signal emulation was designed based on the schematic diagram shown in Figure 5.10. Within the circuit, based on a single clock signal, two outputs A and B are generated. Output A is based on the falling edge of the clock signal and B on the rising edge. A signal divider is used in

such one output signal is produced after two clock signal edges (either rising or falling) has occurred.

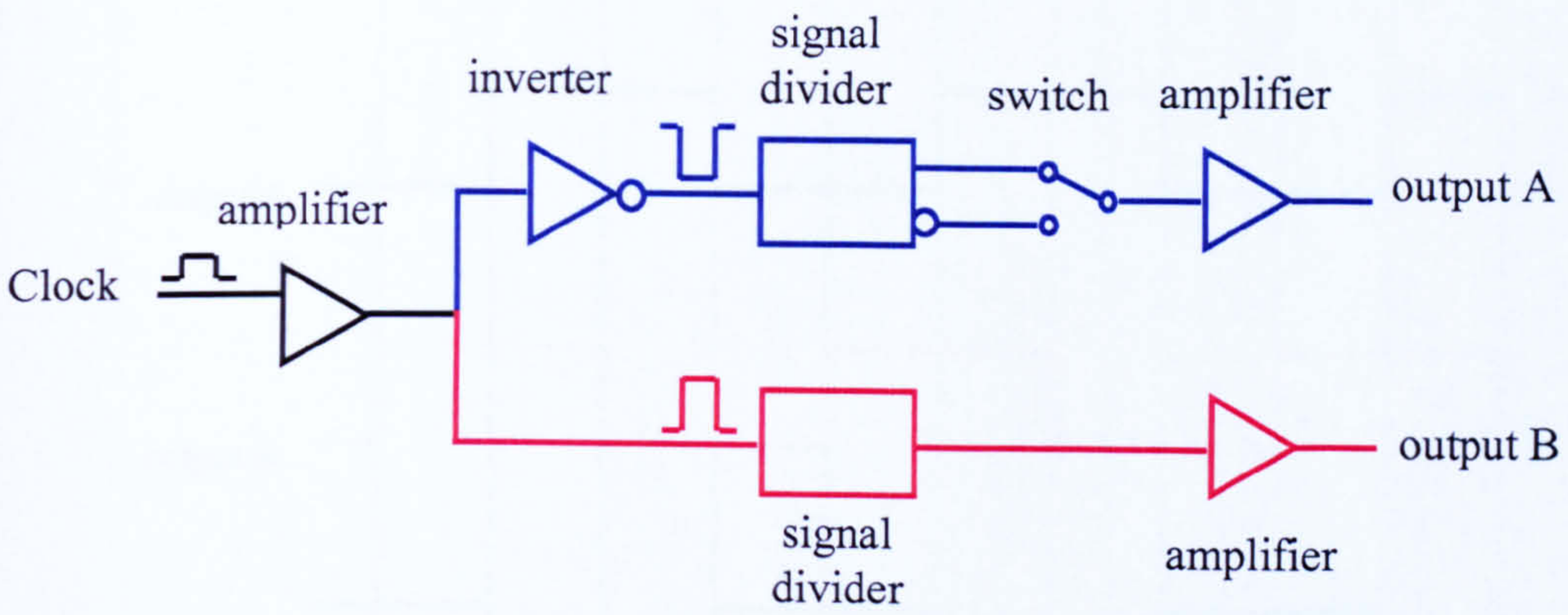


Figure 5.10 Schematic diagram of encoder signal circuitry

A digital switch with external trigger is used to produced an inverted A ($\bar{A}$) signal. An inverted A signal will cause output A leading output B. Hence, emulates the counter clockwise rotation of the encoder. The timing diagram of the clock, output A, output B and inverted output A (i.e. $\bar{A}$) is illustrated in Figure 5.11. Amplifiers are used to strengthen the clock signals so that a perfect square wave is maintained throughout the process.

It should be noted that the clock frequency generated is changing throughout the whole encoder emulation process. Thus, the clock signal generator should able to change its clock frequencies timely and accordingly. An attempt to use a commercial clock/counter PC-based card shows that it is not capable of changing its clock frequencies from one frequency to another. The card required some time to process each clock request command. Hence, either producing delayed clock signals or producing an error in the command program.

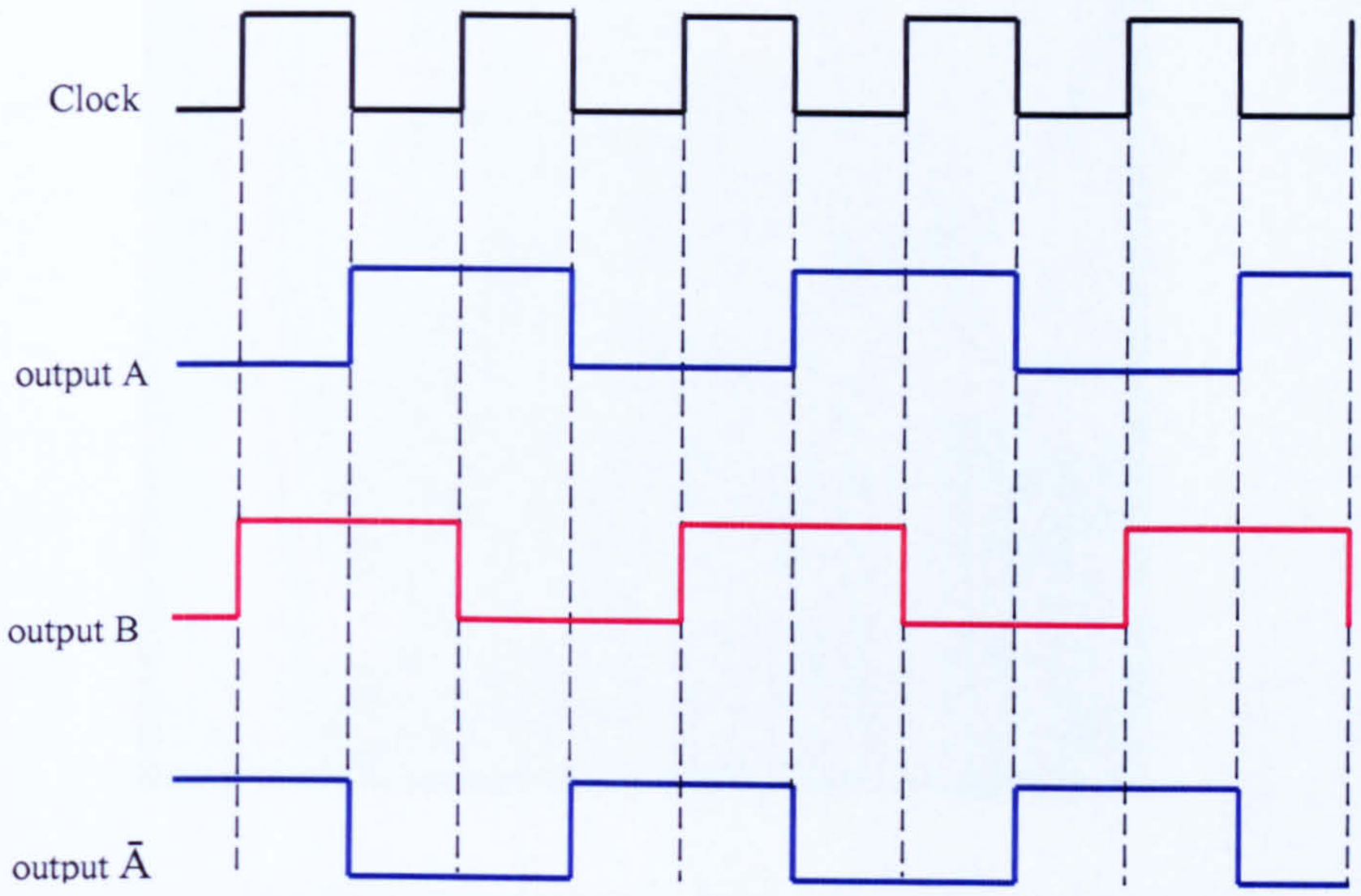


Figure 5.11 Timing diagram of input clock output A, output B and output Ā

For the demonstrator, a commercial stepper motor driver is used to produce the clock signal for the encoder signal emulator circuit. The stepper motor driver is PC-based. The use of stepper motor driver is twofold. Firstly, the stepper motor driver is capable to produce the required clock frequencies. Secondly, there is similarity between stepper motor operation and servo motor operation. Thus, it is capable to adapt with almost all of the motion commands. The circuit built is shown in Figure 5.12 and the schematic diagram of a complete circuit which is able to emulate four encoder signals is shown in Appendix D.

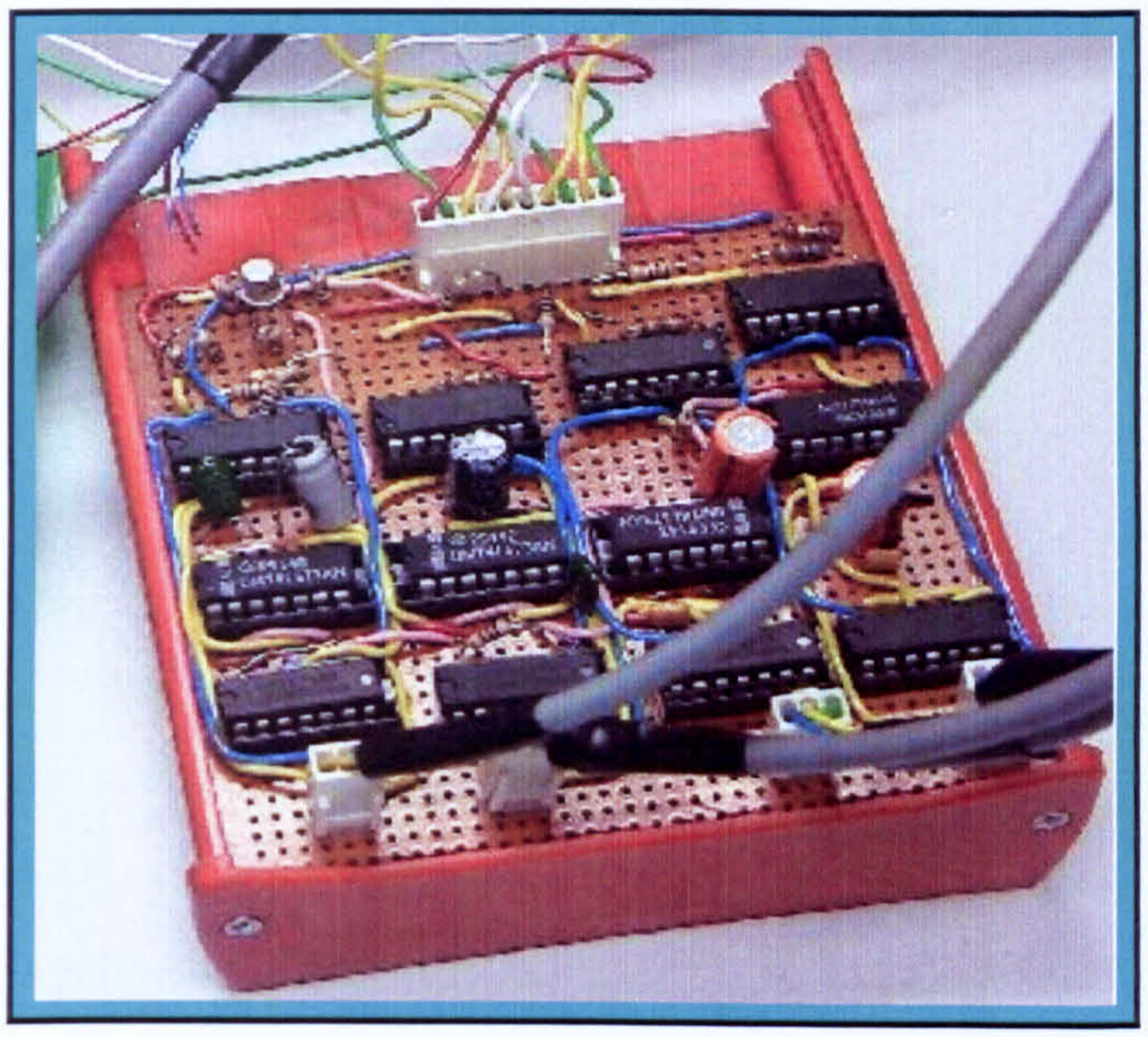


Figure 5.12 Encoder signal generation circuit

5.3.3 Management of Real-Time Emulation Environment Signal

Visual C++ programs were developed to manage individual signal operations in the RTEE environment. Two programs have been developed to control both discrete digital signal and continuous digital signal individually. Each program acts as a device server. The device servers receive commands form Real-Virtual Mapping Environment (RVME) to generate the required emulated signals. Separating the program into individual hardware server helps maintain system modularity and provides easier replacement of hardware. Adopting the TCP/IP socket connection also offers flexibility in a way that the emulation hardware can also be placed in any machine.

On receiving signal request command, the respective programs make use of individual hardware driver libraries to produce an accurate (timely) signal.

5.4 DEVELOPMENT OF REAL-VIRTUAL MAPPING ENVIRONMENT (RVME)

A Windows based application was developed using a Visual C++ environment which serves as a user interface; allowing a user to configure the real-virtual system. The configuration environment is meant for two purposes. Firstly, to allow the user to configure the emulation signals so that they are produced correctly as required. Secondly, to allow the user to configure the communication channels. The program also performs all the three Real-Virtual Mapping Environment (RVME) tasks described in the R-V architecture (discussed in Chapter 4). Figure 5.13 shows a sample of the R-V mapping user configuration environment.

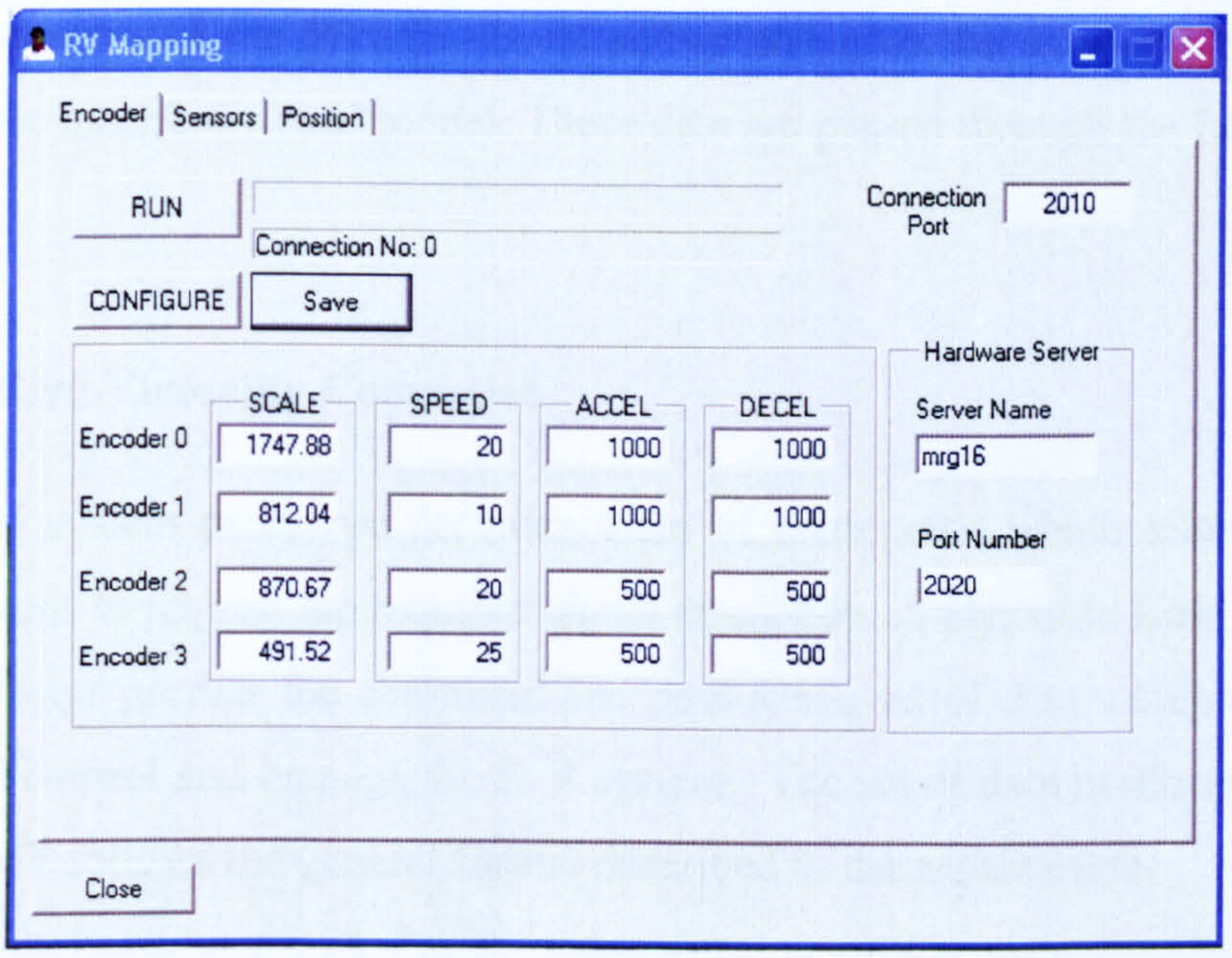


Figure 5.13 The R-V mapping user configuration environments

Communications between RVME and other environments are achieved at a very low level using Windows socket which make the communications flexible and easy to use.

By adopting Windows socket, the RVME program acts as a server to multiple clients and also clients to multiple servers. As a server it receives R-V commands from all the client environments and as clients it is connected to various emulation hardware servers.

The RVME clients are divided into two categories of which are regarded as the *fixed client* and *temporary client*. *Fixed clients* are all the environments which are part of the R-V system architecture (VME and RTEE). *Temporary clients* are user built environments which are incorporated into the R-V system for additional purposes. Commands from *temporary clients* are given higher priority over the *fixed clients* as they are meant to interrupt or override the R-V system behaviour. The priority among the *temporary clients* (in case more then one *temporary client* exists) is given to the first client that connects to the RVME follows by the subsequent clients.

VME connects to the RVME via its server and uses the available positional data to update the machine virtual model. These data are passed through the LLTI connection.

5.4.1 Real-Virtuality Command

The R-V system prototype provides a set of commands which allows the connected applications to request the required signal emulation. A Dynamic Link Library (DLL) is used to helps process the command and produces a set of data which allow the RVME to easily control and manage the R-V system. The set of data produced by the DLL for the RVME follows the general format described in the architecture.

The R-V commands are meant for client signal manipulation of discrete digital signal (for sensor signal generation) and continuous digital signal (for encoder signal generation).

5.4.1.1 Client Sensor Signal Request Command

Since RTEE only provides 16 channels of discrete digital signal emulation, this set of discrete digital signals is regarded as the first block and is identified as BLOCK No. 0 which consist of one word of data. This is as shown below:

BLOCK NO. 0

CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0

(If more channels are used, more blocks will be present with each containing the same amount of data.)

In requesting the discrete digital signal, there are two techniques that can be used. Firstly, requesting a discrete digital signal of a specific channel. Secondly, requesting discrete digital signal of multiple channels (at once). The general format for discrete digital signal request command is as follows:

IORequest (<*server name*>, <*port no.*>, <*blk.no.*>, <*cmd.id*>,<*data1*>, <*data2*>)

The *server name* and *port no.* is the machine where RVME sits and the port number used for communication. The *blk.no.* is the RTEE block number. The denotations for the rest of the data are as shown in Table 1 below.

Table 1: Discrete digital signal command data assignment

Motive	cmd. id	data0	data1
change discrete channel signal status of specific channel	0.0	channel no.	Status (1=5V high 0=0V low
change discrete channel signal status of multiple channel	0.1	Block No.	channel word (in decimal)

5.4.1.2 Client Encoder Signal Request Command

The real industrial controller normally supplies a set of motion control commands which determine how the motor can be operated. Hence, the RVME is designed to be able to adapt to all these possibilities. Since the encoder emulation hardware prototype uses an industrial controller stepper motor driver to generate its clock signal, these do not cause any problem. The stepper motor controller has motion control commands which are basically identical to the servomotor motion control commands. Adopting the stepper motor motion control commands allow to provide adequate R-V commands for encoder signals manipulation.

The R-V system prototype is designed to have a generic form of signal request command. The general form of R-V encoder signal request command for client applications is as follows:

EncoderRequest (<*server name*>, <*port no.*>, < *blk.no.*>, < *cmd. id*>, <*data0*>, <*data1*, <*data2*>, <*data3*>)

where,

blk.no	-	RTEE block number
cmd. id	-	command identification
data1 to data4	-	data for the corresponding motion command

At present, there are six (6) types of R-V encoder signal manipulations available. The command identification available and its corresponding data are as shown in Table 2 below.

Table 2: Encoder emulation command and its respective data assignment

Motive	cmd. id	data0	data1	data2	data3
emulates single axis moving with certain speed	0.0	Axis no.	Axis speed	n/a	n/a
emulates single axis moving with certain acceleration	0.1	Axis no.	Axis acceleration	n/a	n/a
emulates single axis moving with certain acceleration	0.2	Axis no.	Axis deceleration	n/a	n/a
emulates multiple axes moving in certain relative distance	1.0	Axis 0 distance	Axis 1 distance	Axis 2 distance	Axis 3 distance
emulates multiple axes moving in certain absolute distance	2.0	Axis 0 distance	Axis 1 distance	Axis 2 distance	Axis 3 distance
emulates axes stopping	3.0	Axis 0 (1=stop 0=ignore)	Axis 1 (1=stop 0=ignore)	Axis 2 (1=stop 0=ignore)	Axis 3 (1=stop 0=ignore)
reset all encoder parameter	9.0	n/a	n/a	n/a	n/a
reconfigure encoder parameter based on new user input	10.0	n/a	n/a	n/a	n/a

These commands are generally the very basic command which is commonly used for servomotor applications. Additional commands can easily be included for the more advance application.

5.4 SUMMARY

In this chapter, some detail of how a Real Virtuality system can be constructed has been described. The main component that determines whether the proposed system can function or not is the Real Time Emulation Environment (RTEE). TCP/IP sockets; server and client connections are used through out the design to provide a modular flexible system architecture. The following chapter will discuss how to configure and implement the R-V system.

CHAPTER 6

THE REAL-VIRTUALITY (R-V) SYSTEM CONFIGURATION AND IMPLEMENTATION

6.1 INTRODUCTION

This chapter describes how the R-V system has been designed and implemented for machine design and development. Several settings have to be made established before the R-V system can be used for any application. For the implementation, it is assume that:

i. The user has completed the target machine model in the Virtual Machine Environment (VME).

ii. The user has decided on the components to be used in the system.

iii. The user has produced a system wiring diagram.

6.2 CONFIGURING THE REAL VIRTUAL MAPPING ENVIRONMENT

The use of an R-V system starts by configuring the Real Virtual Mapping Environment (RVME). Configuring the RVME generally involves two stages which are:

i. identifying the server and client machines and port assignment for communication

ii configuring the device component specifications

The RVME user interface comes with three user configuration Windows which allows the configuration of virtual encoders, sensors and positional data. The following sections discussed how the configuration is produced.

6.2.1 Mapping Real -Virtuality Encoder

The R-V encoder allows a user to determine how the servomotor on the target machine is used. Since servomotor operation indirectly tells how encoder produces its output signals, it is therefore quite important to understand the kinematics of the servomotor. The basic servomotor kinematics equations which relate angular acceleration, angular velocity and angular displacement can be described as follows:

$$\alpha = \frac{d\omega}{dt} = \text{gradient of } \omega\text{-}t \text{ graph}$$

and

$$(\theta - \theta_0) = \int \omega \, dt = \text{area under the } \omega\text{-}t \text{ graph}$$

where,

θ	-	angular displacement
ω	-	angular velocity
α	-	angular acceleration
t	-	time

Typical servomotor velocity profile for movement between two points is as shown in

Figure 6.1.

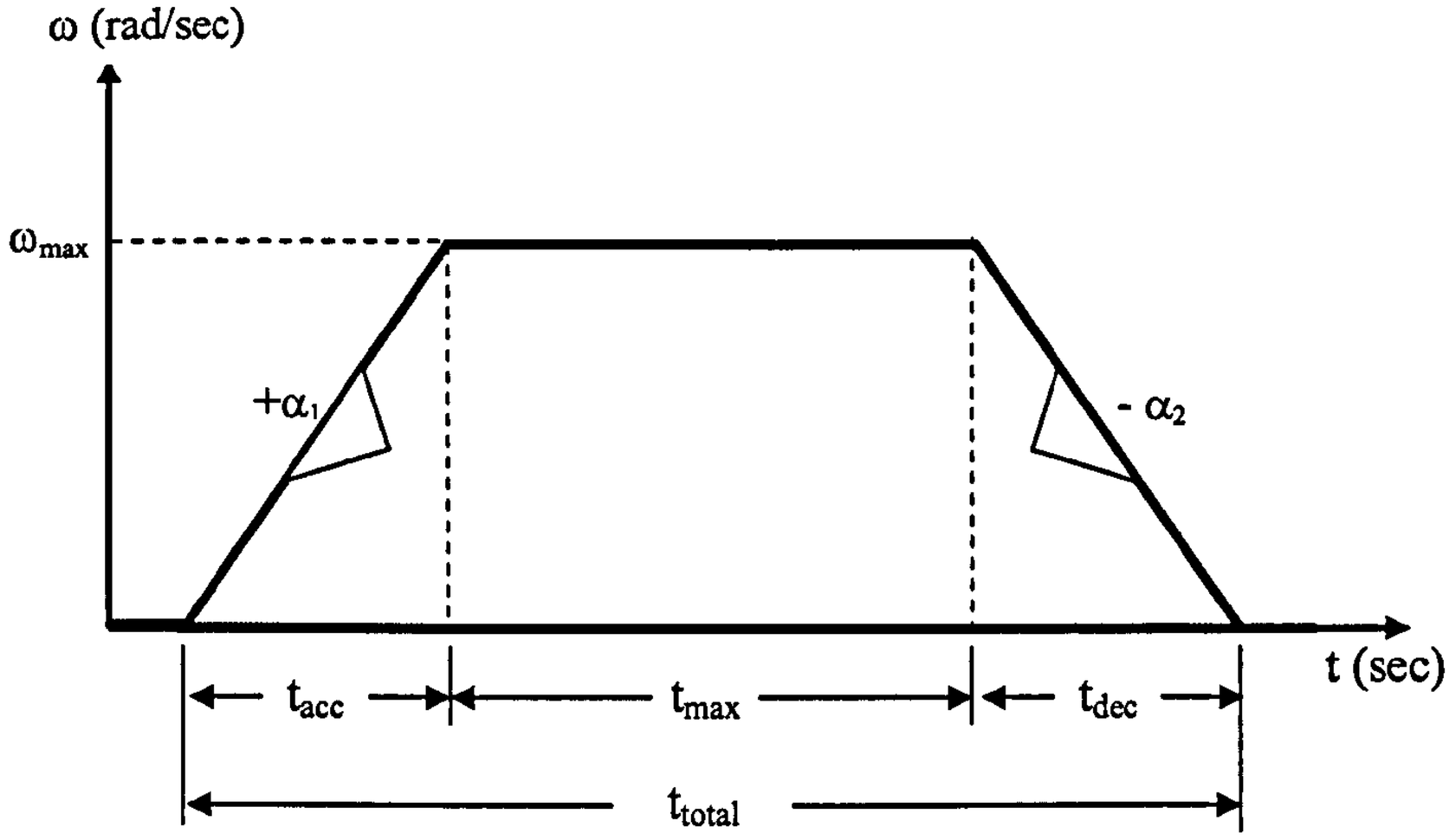


Figure 6.1 Typical motion velocity profiles

Since the gradient of ω - t graph is equivalent to angular acceleration and area under the ω - t graph is equivalent to motor distance travelled, time taken to reach two points (t_{total}), can be formulated as follows:

$$\alpha_1 = \frac{\omega_{\max}}{t_{acc}} \quad \Rightarrow \quad t_{acc} = \frac{\omega_{\max}}{\alpha_1} \qquad [6.1]$$

$$\alpha_2 = \frac{\omega_{\max}}{t_{dec}} \quad \Rightarrow \quad t_{dec} = \frac{\omega_{\max}}{\alpha_2} \qquad [6.2]$$

$$(\theta - \theta_0) = \tfrac{1}{2}(t_{\max} + t_{total})(\omega_{\max}) \qquad [6.3]$$

and,

$$t_{\max} = t_{total} - t_{acc} - t_{dec} \qquad [6.4]$$

Substituting equations [6.1], [6.2] and [6.4] into equation [6.3] with initial angular displacement $\theta_0 = 0$,

$$\theta = \tfrac{1}{2}[(t_{total} - \frac{\omega_{\max}}{\alpha_1} - \frac{\omega_{\max}}{\alpha_2}) + t_{total}](\omega_{\max}) \qquad [6.5]$$

If the motor deceleration is equals to its acceleration; i.e. $\alpha_1 = \alpha_2 = \alpha$, then

$$\theta = (t_{total} - \frac{\omega_{\max}}{\alpha})(\omega_{\max})$$

i.e. $$\boxed{t_{total} = \frac{\theta}{\omega_{\max}} + \frac{\omega_{\max}}{\alpha}} \qquad [6.6]$$

However in real applications, it is quite normal for the motor to ramp up (and also ramp down) as quickly as possible to achieve its maximum speed (or to a stop). In such case, the value of t_{acc} and t_{dec} are very small which will result in a velocity and displacement profile shown in Figure 6.2 and Figure 6.3 respectively.

In such case, the approximate value of t_{total} will be,

$$t_{total} = \frac{\theta}{\omega_{\max}} \qquad [6.7]$$

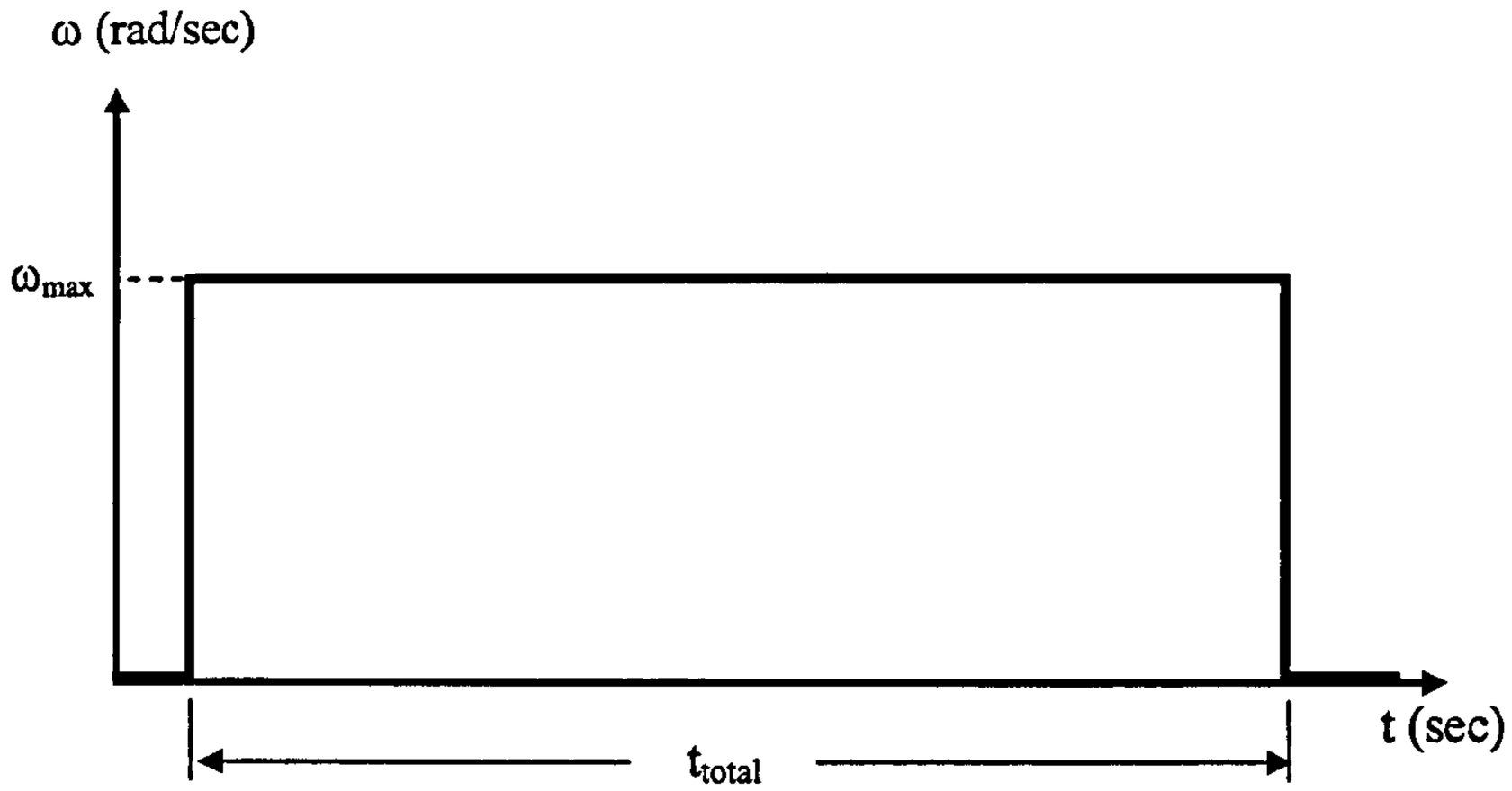


Figure 6.2 Velocity profile for motion with high acceleration and deceleration

Figure 6.3 Displacement profile for motion with high acceleration and deceleration

Some parameters need to be configured before the RTEE can produce the required encoder signal. Based on the above, the parameters needed to be configured are the

maximum speed, acceleration and deceleration which will determine how fast it takes for a servomotor to moves from one point to another. However, encoders which encode motor parameters have various scales. Hence, the encoders' scales should also be configured.

Figure 6.4 shows the virtual encoder configuration environment. These values are used to generate the initial encoder's motion profile for each axis. However, these parameters can be changed during the system operation by means of the R-V Command if needed.

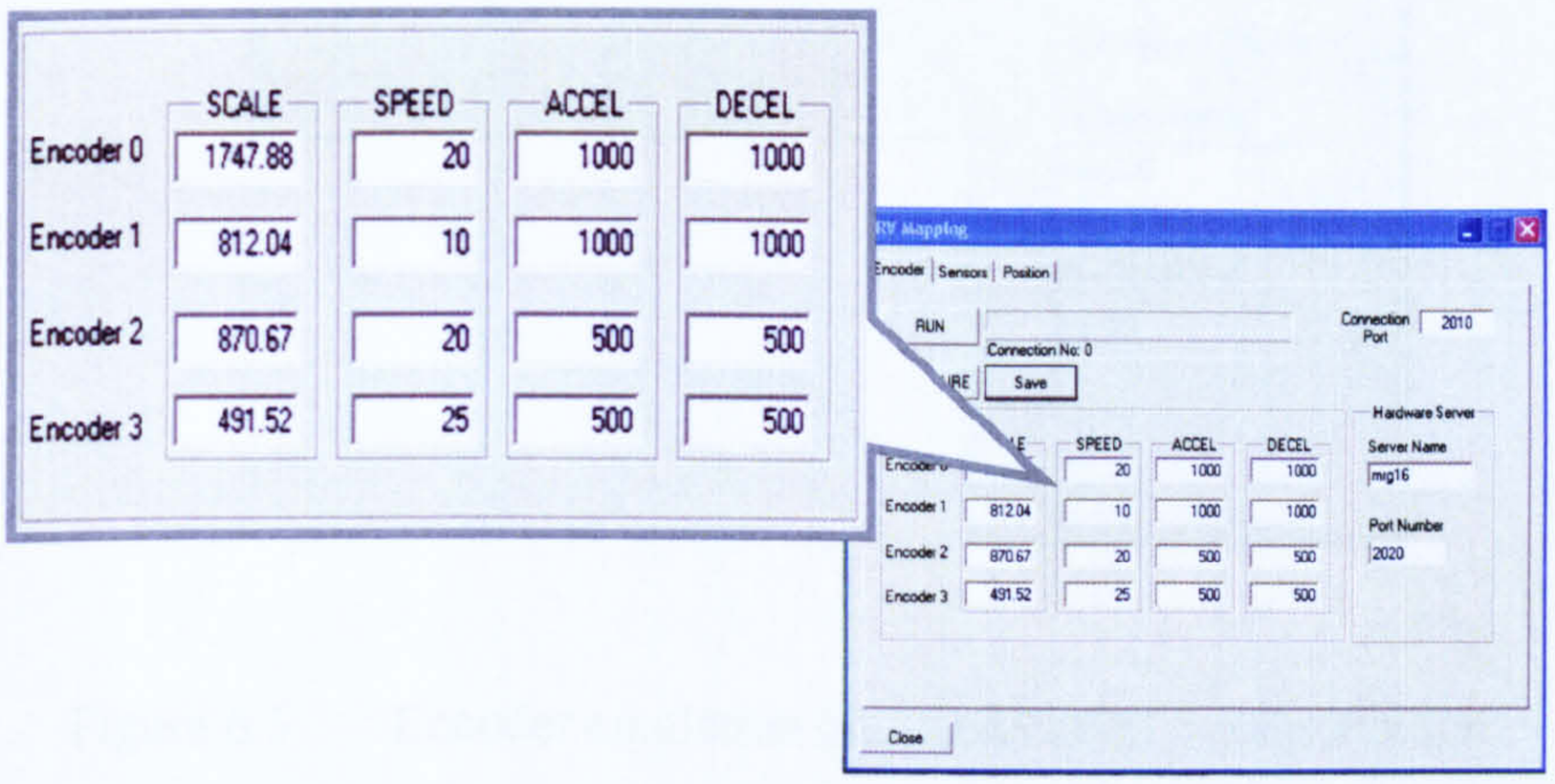


Figure 6.4 Configuring the initial encoder emulation parameter

Alongside the initial encoders' parameters, the communication ports for the encoder client and the hardware server should also be configured. Encoder client port is the port where external application sends request command for encoder signal generation. Hardware server is the machine (i.e. computer) where the encoder emulation hardware sits. For example, for the configuration shown in Figure 6.5 indicates that any application can send encoder request command via *port 2010* and the emulation hardware is sitting in a machine name *mrg16* which communicates via *port 2020*.

Once all these configurations have been implemented, the system is ready to accept all the encoder R-V commands. It is started when the RUN button is pressed.

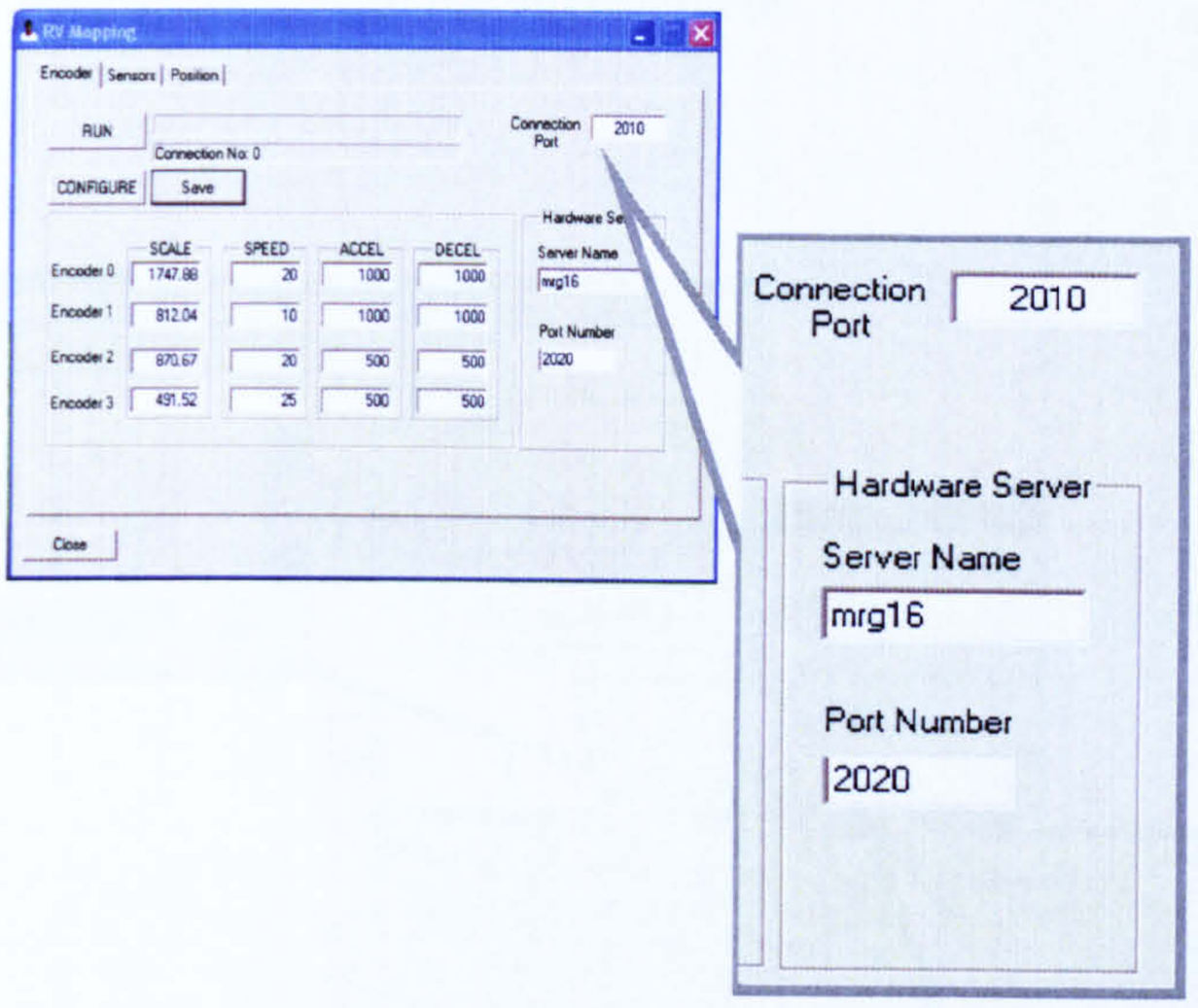


Figure 6.5 Encoder emulation communication configurations

6.2.2 Mapping the Real Virtuality Sensor

Mapping the R-V sensor is achieved using a configuration Windows which allows user to determine which *virtual sensor* corresponds to the *real input* of the machine controller. This is shown in Figure 6.6 below. *Virtual sensors* are the sensors which exist in the Virtual Machine Environment (VME). The virtual sensors channels arrangement depends on where each sensor is assigned in the virtual controller (in VME). The *real input* channels are the emulated sensors signals which are digital inputs to the real controller. In other words, the real input channels are the 16 channels which exist within the sensor emulation hardware.

The mapping between the virtual sensor channels and the real input channels are not necessarily of the same value. However, within the demonstrator, the same channel numbers are used.

The communication ports for the R-V sensor client and the hardware server name and port also have to be configured within the environment.

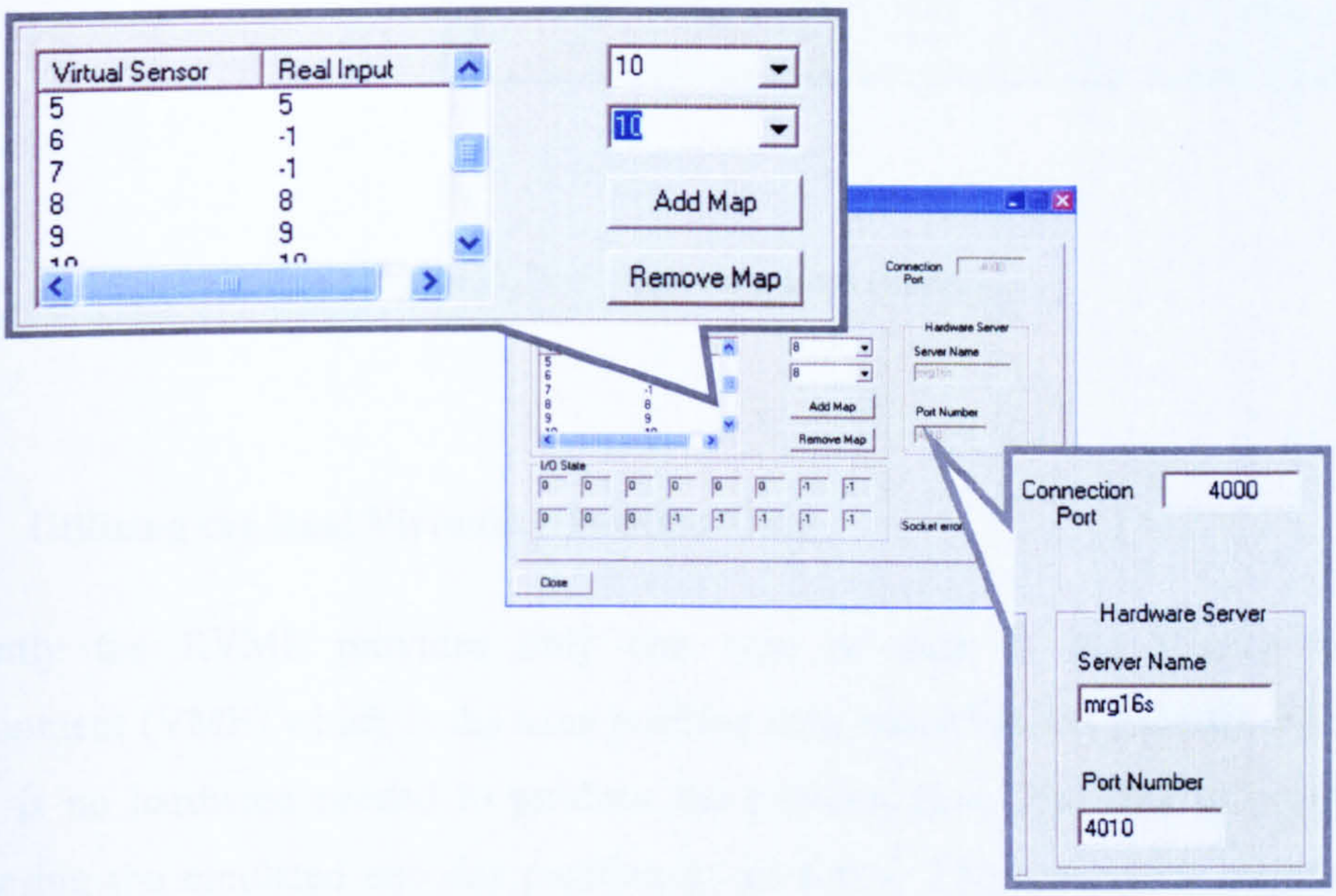


Figure 6.6 Sensor mapping environment

Once the R-V system is running, the emulated sensor states can be verified within the RVME. This is shown in Figure 6.7 below. The states are interpreted as follows:

0 - sensor is not active

1 - sensor is active

-1 - no sensor is connected.

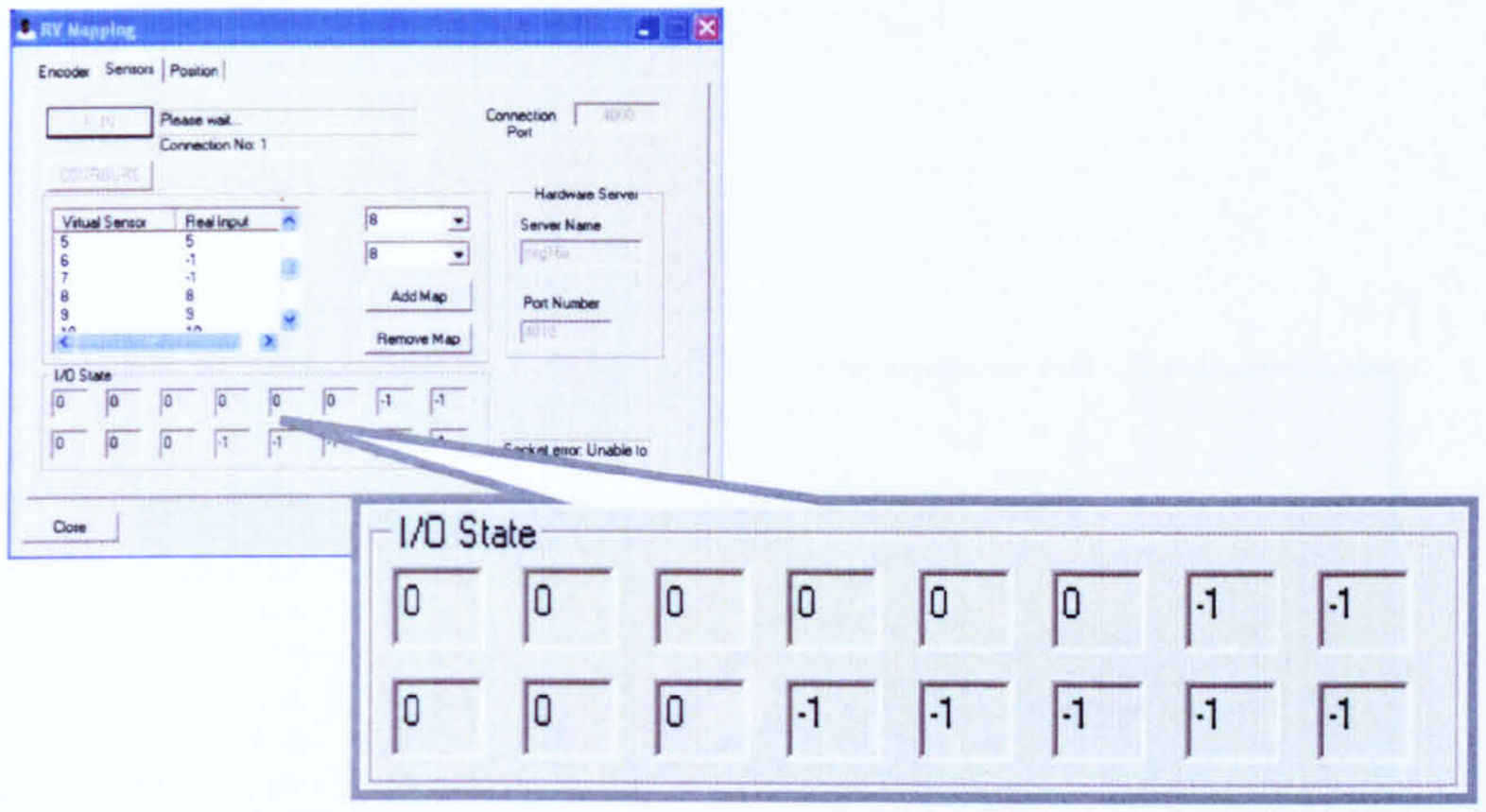


Figure 6.7 Sensor status display

6.2.3 Utilising the Real Virtuality Position Data

Currently the RVME provides only one type of data to the Virtual Machine Environment (VME) which is the axes *position* data related to the encoder movement. There is no hardware needed to produce the position data. The data is acquired by monitoring the emulated encoder position at all times. The position is recorded and updated constantly until the system hardware is reset.

Position data is always available within the R-V environment. The position data are based on the changes of the active encoders positions (with its initial start position as zero). Users can configure which port is to be used as the data communication port. Once the port has been specified and the RUN button is pressed, the data will be continuously sent out to the connected client applications. Figure 6.8 shows the position data configuration Windows.

The position data for all the axes are transferred continuously to any client connected to the RVME data port. VME will also know which data belongs to which axis so that the

correct axis position can be updated on the machine model.

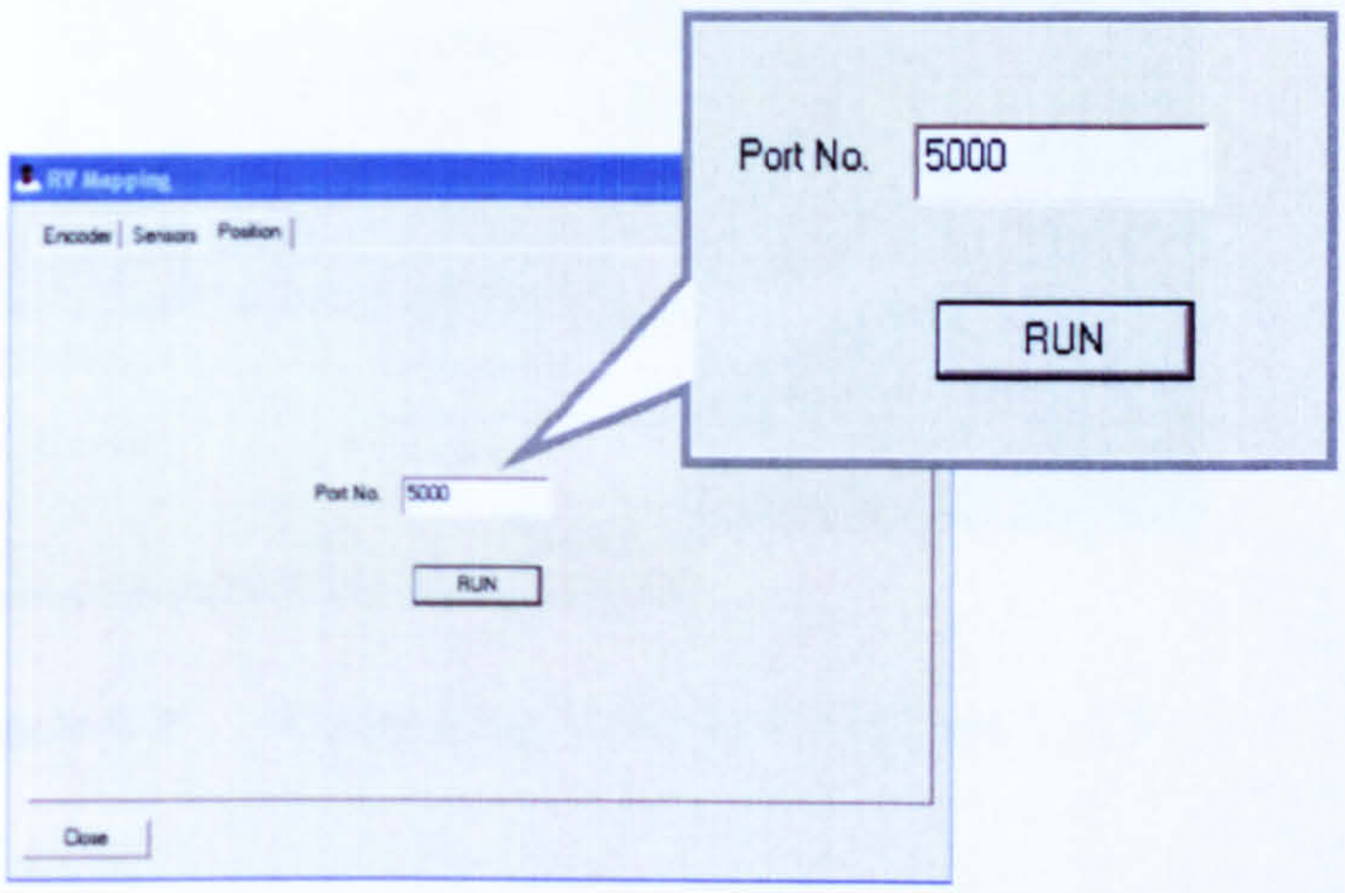


Figure 6.8 Position data configuration Windows

6.3 CONFIGURING THE VIRTUAL MACHINE ENVIRONMENT

The completed virtual model is connected to the Real Virtual Mapping Environment (RVME) via its external interface module. In IGRIP, this is done by integrating the developed R-V LLTI Dynamic Link Library (DLL) routine to the model which will allows socket communication to the RVME. This is shown in Figure 6.9.

Once connected, the VME starts receiving position data from RVME. The position data is updated continuously as long as VME is connected to RVME.

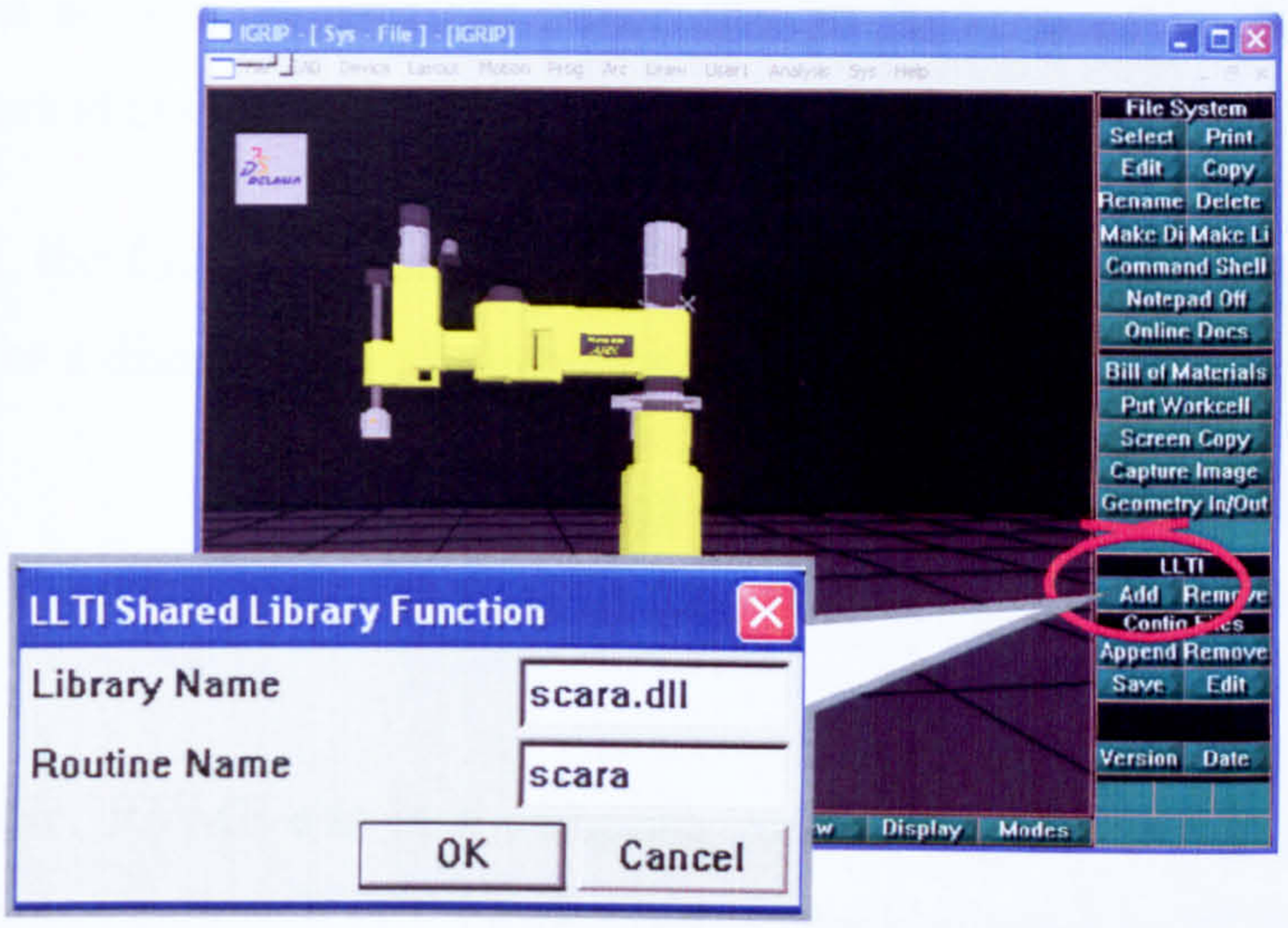


Figure 6.9 Connecting VME to RVME via LLTI

6.4 EMULATION SIGNAL REQUEST COMMAND

The emulated signals will only be generated if a request command is made via the hardware server. RVME manages all signals request and pass the request to the respective hardware server.

Since all signals request is made through RVME, it is important to know on which computer (server) does the RVME runs and which port is used for communication (accepting the command) before an R-V command is used. The *server name* and *port number* normally stay the same throughout the use of R-V commands.

6.4.1 Client Sensor Signal Request

Sensor signal commands controls the states of virtual sensors available within the discrete digital signal emulator. There are two techniques available to change the state

of the virtual sensors. A user can either change the state of an individual sensor or several sensors at one time.

For example, the following command will produce a 5V (high) digital signal for CH7 (channel 7) for a discrete digital signal emulator hardware:

IORequest ("mrg16s", 2010, 0, 0.0, 7, 1)

In this example, RVME sits in a computer named *mrg16s* and is using port number *2010* for communication. RTEE block used is block number is '0' (since there is only one RTEE block). On the other hand the following command will reset the channel to 0V (low):

IORequest ("mrg16s", 2010, 0, 0.0, 7, 0)

To produce a 5V digital signal for CH5, CH7 and CH14 simultaneously, the following command can be used:

IORequest ("mrg16s", 2010, 0, 1.0, 0, 16544)

The rest of the channels will have a low state (0V). The integer value "16544" is formulated as follows:

The 16 channels are represented by a 16-bit binary numbers with the lowest channel as the lowest bit as shown below. (High states are represented by '1' and low states as '0').

CH15	CH14	CH13	CH12	CH11	CH10	CH9	CH8	CH7	CH6	CH5	CH4	CH3	CH2	CH1	CH0
0	1	0	0	0	0	0	0	1	0	1	0	0	0	0	0

Mathematically,

$$0100000010100000_2 = 40A0_{hex} = \mathbf{16544_{10}}$$

The following command can be used to reset all channels to 0V low:

IORequest ("mrg16s", 2010, 0, 1.0, 0, 0)

6.4.2 Client Encoder Signal Request

Initial configuration which represents a servomotor motion profile can easily be configured in the RVME user Windows interface as described in Section 6.2.1. However, it is quite typical that certain parameters such as speed or acceleration are changed from time to time during machine operation.

The following command will change the speed of Axis No. 0 to a new speed value equal to 500 (in the axis movement unit):

EncoderRequest ("mrg16s", 2010, 0, 0.0, 0, 500,)

In this example, RVME sits in a computer named *mrg16s* and is using port number *2010* for communication. RTEE block number is '0'.

A servomotor movement is emulated with a movement either in absolute movement or relative movement. A relative movement with a distance of 100 for Axis No. 0 and 120 for Axis No.1 will looks like the following:

EncoderRequest ("mrg16s", 2010, 0, 1.0, 100, 120, 0, 0)

The following command will generate an encoder signal for Axis No. 1 and Axis No. 2

which resembles absolute movement of 100 and 120 respectively. No encoder signal is generated for Axis No.0 and Axis No.3:

EncoderRequest ("mrg16s", 2010, 0, 2.0, 0, 120, 100, 0)

There are several other commands which can be used for encoder signal manipulations. The parameters to be used are as tabulated in Table 4 in Section 5.4.1.2

6.5 R-V SYSTEM WIRING

The R-V system uses the real controller to drive the system; the actual wiring technique should be employed on the controller. The controller input signals wires (i.e. from all sensors and encoders) come from the RTEE hardware. The controller outputs are not a concern; hence, no wiring is needed. In order words, at this stage the assignment of all controller I/Os has already been identified and well documented.

Figure 6.10 shows the wiring arrangement for the R-V system prototype. The wires from the discrete digital signals emulator (sensors) are connected to the NextMove controller breakout board input terminal meanwhile the encoder signals emulators wires are connected to the encoders' connectors.

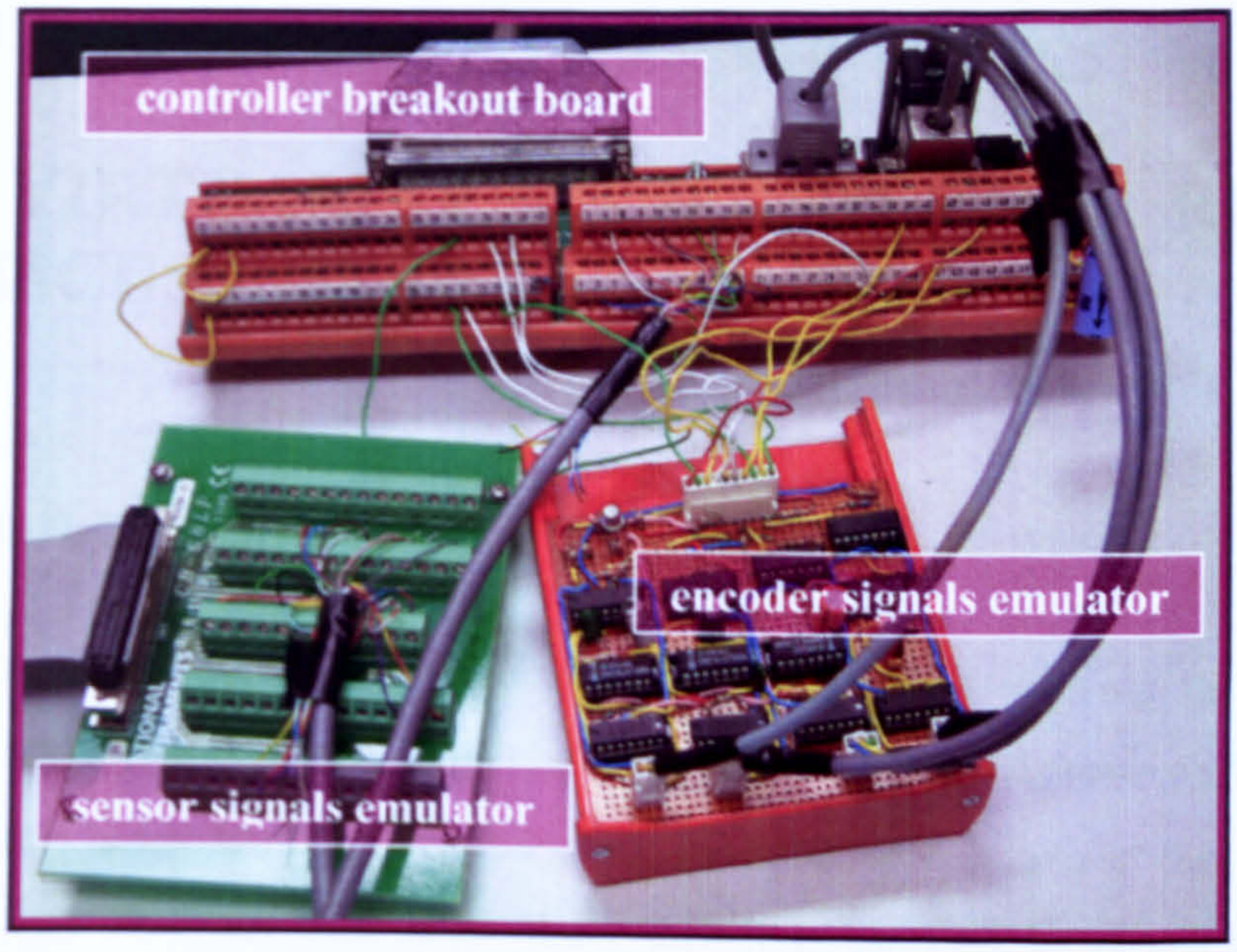


Figure 6.10 Input signal wires from RTEE connected to a controller breakout board

6.6 SUMMARY

In this chapter the detail of how to design, configure and implement the R-V system has been explained. Most of the configuration relates to the Real Virtual Mapping Environment (RVME) which maps between the virtual machine environment and the real world. These include the configuration of the signal emulator parameters and client-server identification. Detail on the use of R-V commands to manipulate the RTEE environment has also been described.

The following chapter presents some results obtain from tests undertaken on the R-V system formulated.

CHAPTER 7

REAL VIRTUALITY CASE STUDIES, RESULTS AND DISCUSSIONS

7.1 INTRODUCTION

The validation of the R-V system architecture was done by utilising the R-V system prototype in actual applications. This chapter explains the employment of R-V system with three test cases which were used to evaluate and validate the R-V system architecture. The first test case is to show that it is possible to develop, verify and validate a real controller program using the R-V system. The second test case is to show the real time capability of the R-V system. The last test case is to show the possibility of using an R-V system for an advanced application.

Results from the test cases are presented and discussed subsequently.

7.2 TEST CASE I: REAL-TIME CONTROLLER PROGRAM DEVELOPMENT AND VERIFICATION USING AN R-V SYSTEM

In this test case, a motion controller (NextMove) program together with a user interface which can be used to control and manipulate the target (actual) machine was developed. The R-V environment was used to assist the development of the program. Utilising the R-V environment, the control program can be verified and validated throughout the program development cycle. R-V system helps to provide real-time visualisation of how the machine responds when a control program is executed.

The controller program was developed using the Visual C++ programming language[16] which provides a human machine interface (HMI) in a Windows environment. All the R-V sensors and encoders are connected to the NextMove breakout board beforehand. The R-V arrangement for the case study is shown in Figure 7.1.

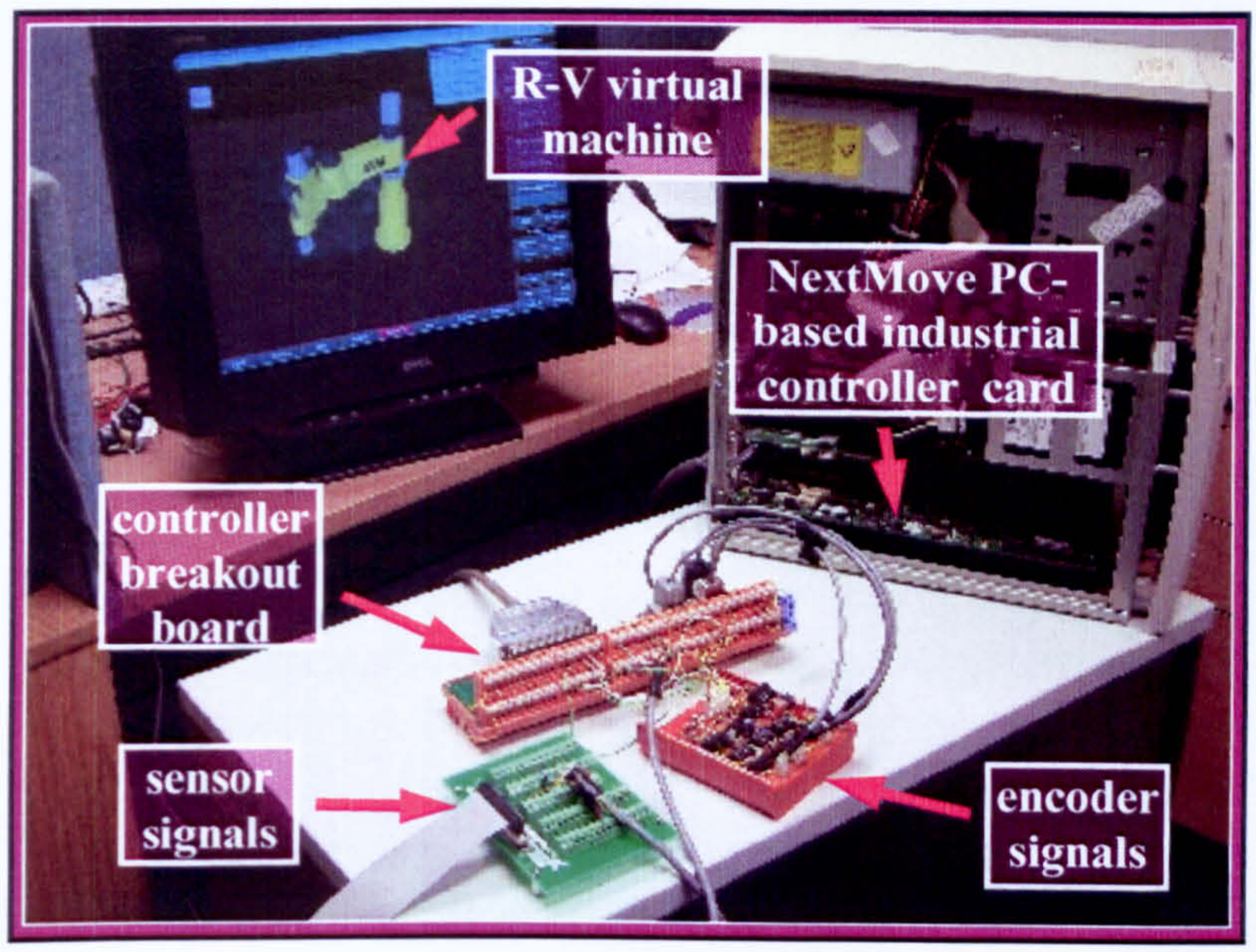


Figure 7.1 R-V arrangements for control program development and verification

For this research study, the controller program was designed to perform three functions. This is illustrated in the user Windows environment shown in Figure 7.2. The available functions are as follows:

i. jogging each individual axis forward and backward with variable speed

ii. move all axes simultaneously in a sequence of moves for one cycle

[16] The NextMove can also be programmed in another language called MINT which can be loaded into the controller memory. Visual C++ was chosen for ease of communication using Windows socket.

iii. move all axes simultaneously in a sequence of moves (up to five sequence) continuously (loop)

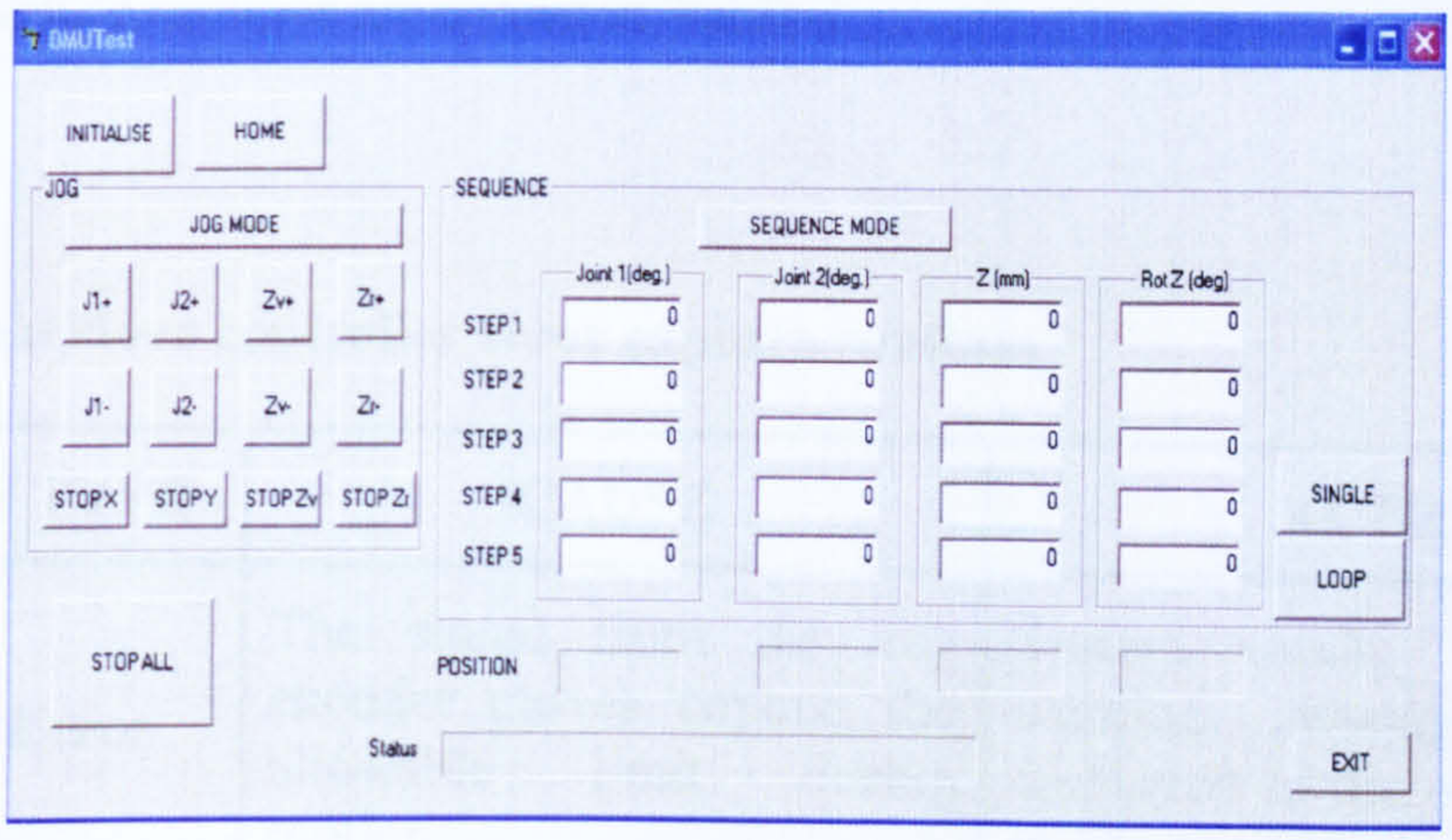


Figure 7.2 NextMove controller user's Windows-based human machine interface

The controller program formulated is an actual program which is to be used on the real machine. The only difference between the program developed in an R-V environment and the one used for real machine program is that, all controller commands which trigger motor movements have to be followed by an R-V encoder request command. However, these additional R-V commands have no effect in the real operation of the program. Thus, they can either be removed later or left as it is before the program is used on the real system.

The popular three term PID (Proportional-Integral-Derivative) control strategy is used for the target machine. During the controller program development stage, the control parameters (Kp, Ki, Kd) are not introduced in the program. They are indeed not needed and have no effect to the R-V system environment. The control parameters however can be included if they are already known. In practice the control parameters are normally tuned during the real system test run. Thus, the correct control parameters are normally

added at a later stage.

Three error capture routines have also been embedded within the program. These routines determine how the controller responds when certain errors occur in the system. The error capture routines are shown in Table 3.

Table 3: NextMove controller error capture routines

ERROR TYPE	CAUSE	ACTION
Following Error	The signal from the any encoder moves beyond the allowable limit during motion.	System totally stops. Error warning message will be displayed to the user. User has to fix the error[17]. Initialisation is needed.
Limit Error	Axis reaches it maximum or minimum distance limits. Forward Limit Switch or Reverse Limit Switch will activates.	The corresponding axis stops. Error warning message will be displayed to the user. The axis can only be moved in the direction away from the sensor that triggers the error.
Axis Error	Axis fails because of motor failure	System totally stops. Error warning message will be displayed to the user. User has to fix the error. Initialisation is needed.

An audible warning can be heard if any of the above errors are triggered.

[17] In real application, this error shows that the control parameter is not properly tuned. In R-V system, this shows the encoder emulation signal is not correctly produced.

7.2.1 Using The Controller Program Formulated On The Real Machine

Once the developed program has been tested and verified in the R-V environment, it is used on the real machine. The machine's servomotors are tuned beforehand and the appropriate control parameters are included.

The PID tuning method used is the "trial and error" method. The steps on tuning the machine axes (robot axes) are as follows (Stenerson 2003):

i. Turn all gains (Kp, Ki, Kd) to zero

ii. Adjust the proportional gain (Kp) until the system begins to oscillate

iii. Reduce the proportional gain until the oscillation stops and then reduce it by about 20 precent more.

iv. Increase the derivative term (Kd) to improve the system stability

v. Raise the integral term (Ki) until the system reaches the point of instability and reduce it slightly.

Stenerson (2003) suggests the use of an oscilloscope to view the system performance during tuning. However, since this research study is not about evaluation of PID control techniques or control tuning, visual performance is used during the tuning. From the tuning the control parameters shown in Table 4 are included in the program.

Once the PID control parameters have been included the developed machine control program was able to be used on the real machine. There is no modification needed to be done on the machine control program. However, when running the control program on the real machine, the *following error* function has to be turned off because of the inaccuracy in the tuning the PID parameters.

Table 4: PID control parameters

Axis	Kp	Kd	Ki
Axis 1	0.25	2.0	0.0
Axis 2	0.3	1.8	0.0
Axis 3	0.2	5.0	0.0

7.2.2 Results and Discussion of Test Case I

The test case has shown that the R-V system is capable of assisting in developing, testing and verifying machine control systems. The R-V system allows a programmer to test their control system code without having to have access to the real equipment.

The transfer between the R-V environment to the real system was found to be smooth with no modification or alteration (accept the inclusion of the control parameters) have to be done to the control program developed. A video file on the CD incorporated with this thesis includes test case I. The files in folder *switch R-V to Real*[18] demonstrates how a machine control program developed in R-V system is used directly with the real system (provided the correct machine control parameters are included).

Specific outcomes from this test case can be drawn as follows:

i. the signal emulation hardware is capable of producing the required output signals.

ii. a working control program also proves that all sensor and encoder connections were correctly assigned. A wrong connection or terminal assignment is easily detected during the control program verification.

[18] The video is presented in three different file format *. mpg, *.wmv and *.rm which can be played using most media player such as Microsoft Windows Media Player® or Real Player®.

iii. the virtual sensors formulated within the Virtual Machine Environment are good enough to represent the real sensors behaviours.

Overall findings from the test case were as follows:

i. the control program developed using the R-V system has been tested, verified and validated; ready for execution on the real machine.

ii. it is possible to use a virtual simulation package, such as IGRIP, for real time simulation.

iii. the signal emulation technique proposed is viable and has been shown to work with acceptable performance.

7.3 TEST CASE II: R-V SYSTEM REAL-TIME CAPABILITY EVALUATION

This test case covers two main aspects, firstly, to evaluate the capability of the signal emulators to produce the required signals and secondly, to demonstrate that the R-V system has real-time capabilities.

The initial test was on the encoder emulator. Each encoder emulator is initiated to produce a motion signal for movement between two positions. The individual emulator encoder speed and position data are then recorded. The configuration for each axis is as shown in Table 5. The speed, acceleration and deceleration units that correspond to the movement type of each individual axis.

Table 5: The R-V encoder emulator configuration for accuracy evaluation

Axis	SPEED	ACCEL.	DECEL.	MOTION
Axis 1	20	1000	1000	0° to 100°
Axis 2	20	500	500	0° to 120°
Axis 3	10	1000	1000	0 mm to 75 mm

Consequently, a five loop kinematics test was carried on both the R-V system and real system. Both systems were given the same movement sequences. A simple program was also written to allow capturing the speed and positional data from the controller.

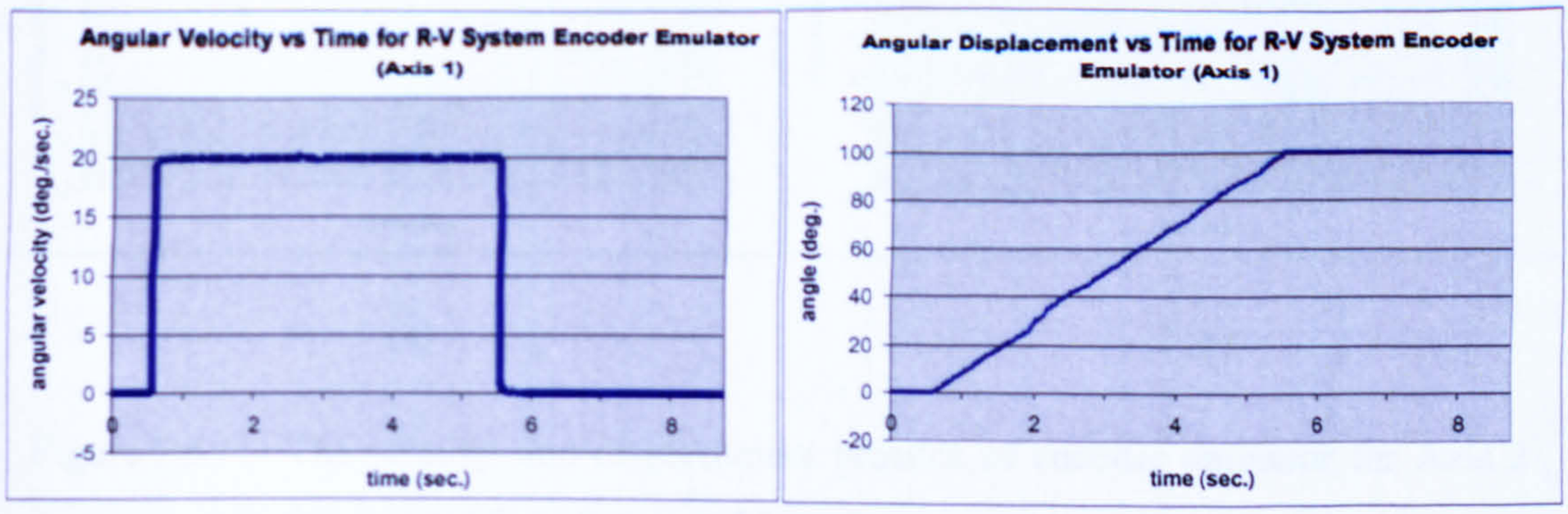
The real system data was recorded from the positional data capturing program meanwhile positional data from the RVME is used for the R-V system. Comparison was then made between both sets of system data. .

To better access the real-time capability of R-V system, two identical NextMove controllers were then used to drive the R-V system and the real machine separately. Employing the continuous motion sequence available in the control program, both the R-V system and real system were programmed to have the same continuous motion sequence. Subsequently the systems are started simultaneously. Visual analysis was then done to check whether the R-V system and the real system demonstrates synchronised motion.

7.3.1 Results and Discussion of Test Case II

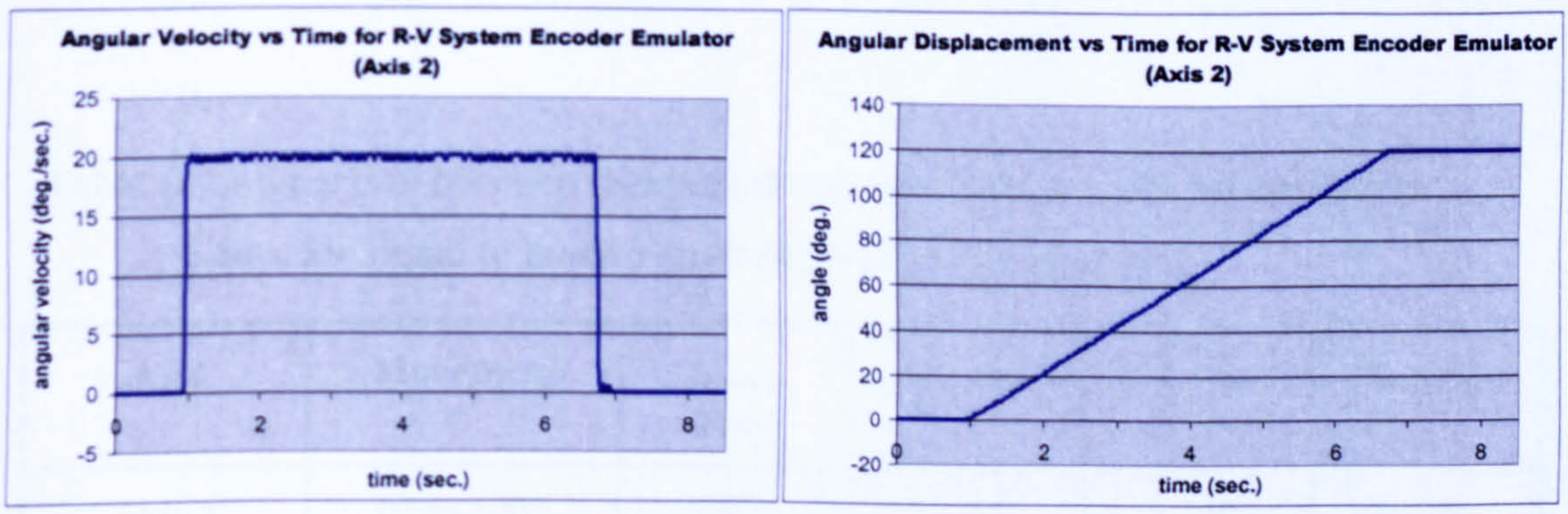
Figure 7.3 to Figure 7.5 show the graphs of velocity and displacement in respect of time for the three axes of the encoder emulators. The recorded encoder data from the tests is

presented in Appendix E.



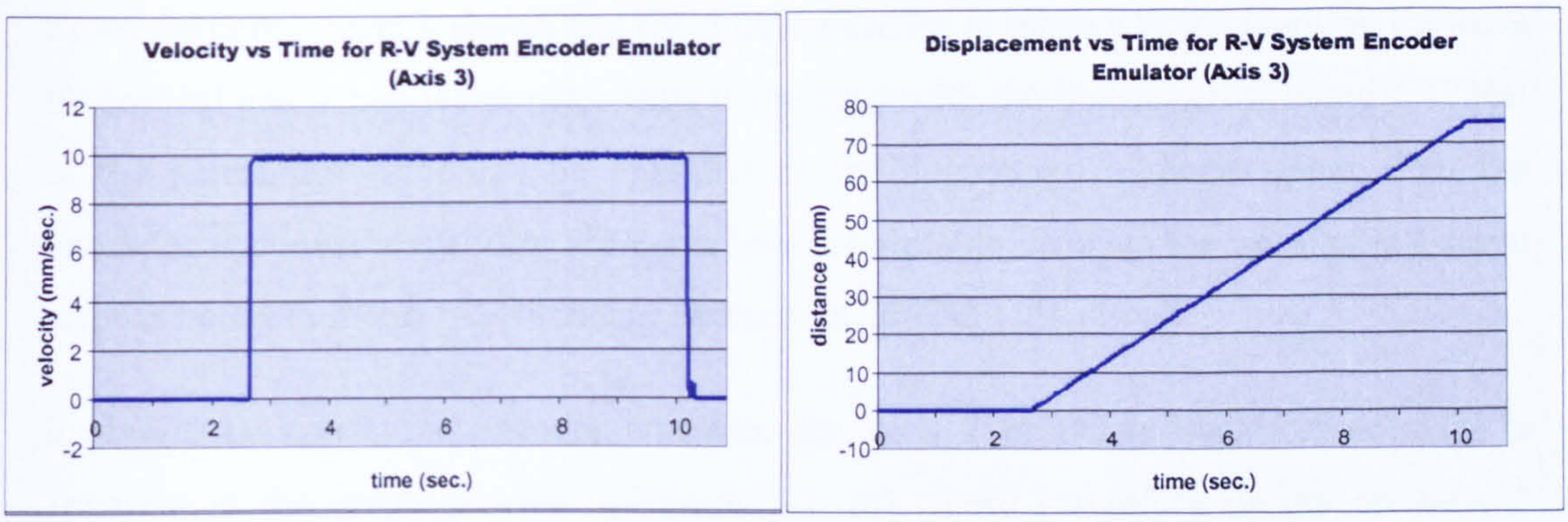
(a) (b)

Figure 7.3 The angular velocity and displacement profiles of encoder emulator for Axis 1 (movement from 0° to 100°)



(a) (b)

Figure 7.4 The angular velocity and displacement profiles of encoder emulator for Axis 2 (movement from 0° to 120°)



(a) (b)

Figure 7.5 The velocity and displacement profiles of encoder emulator for Axis 3 (movement from 0 mm to 75 mm)

From the three set of graphs, it shows that the emulated encoder speed and displacement profiles conform to the theoretical profiles described in Section 6.4.2. Comparison is also made on the time taken to reach the desired position calculated using equation [6.6] and from the tests. The results are tabulated in Table 6. The details of the calculations are shown in Appendix F.

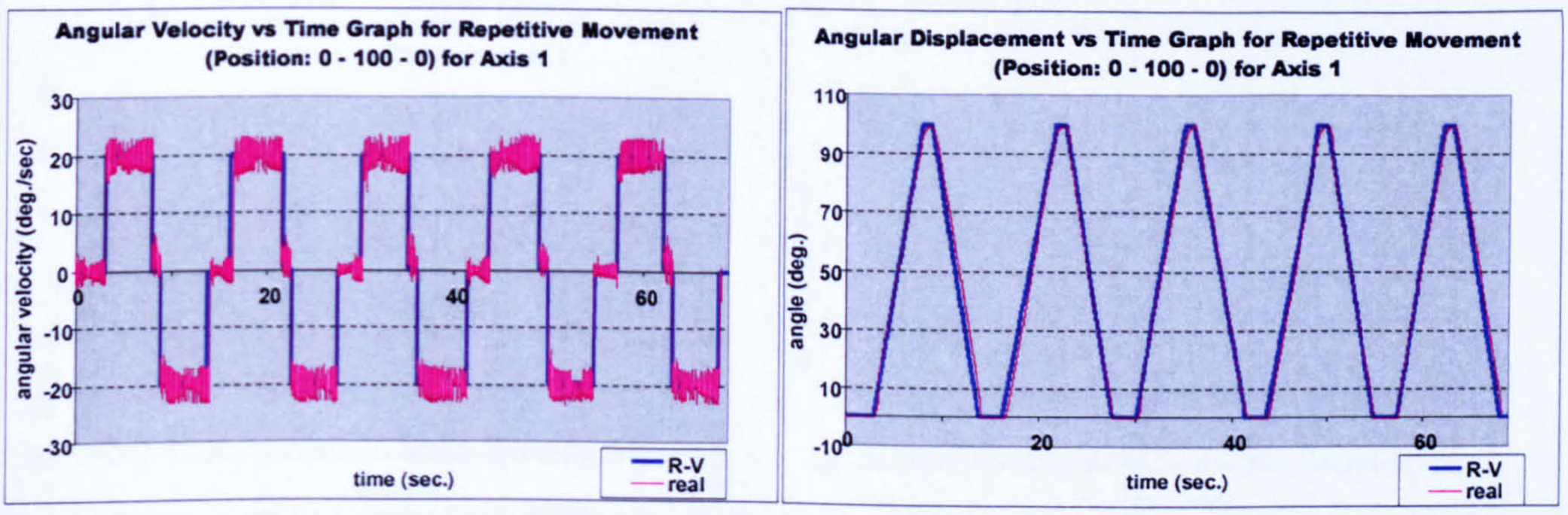
Table 6: Comparison between theoretical and actual time taken for emulated encoder signal to reach a desired position for all axes.

Axis	Movement	t_{theory} (sec.)	t_{actual} (sec.)	$t_{diff.}$ (sec.)	% error
Axis 1	0° to 100°	5.020	5.012	0.008	0.159
Axis 2	0° to 120°	6.040	6.135	0.095	1.573
Axis 3	0 mm to 75 mm	7.510	7.498	0.012	0.160

From the comparison it shows that the time difference to reach a target position between theoretical and actual is not more then 0.1 second with the highest error being 1.573%. If the percentage error can be regarded as the percentage of servo error, then the emulator encoders' capability are considered acceptable because the satisfactory servo error is between 2% to 5% (National Instrument 2007).

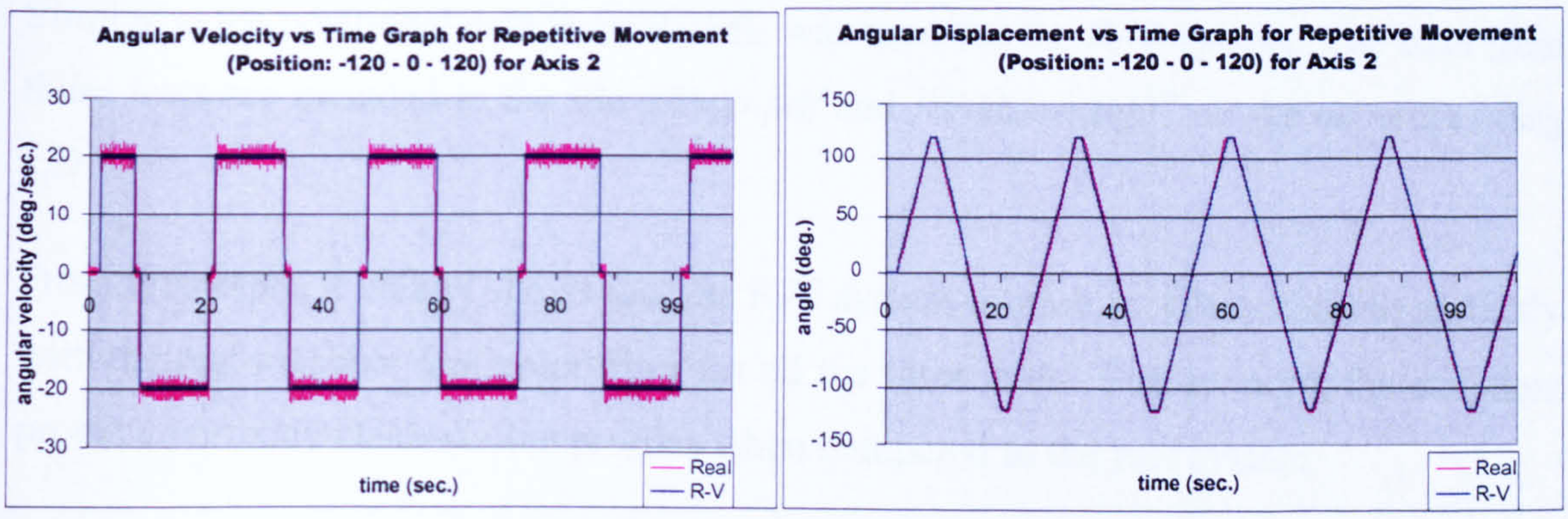
It should be noted, the encoder emulator for Axis 2 produces the most error. It is apparent in the graph (shown in Figure 7.3 (a)) where instability occurs on Axis 2 encoder emulator output. This instability was probably due to electrical noise and/or external interference.

Figure 7.6 to Figure 7.8 shows the results from five loops tests carried out on both the R-V system and real system. The position and velocity graphs of the real system are superimposed onto the encoder emulators' position and velocity graphs.



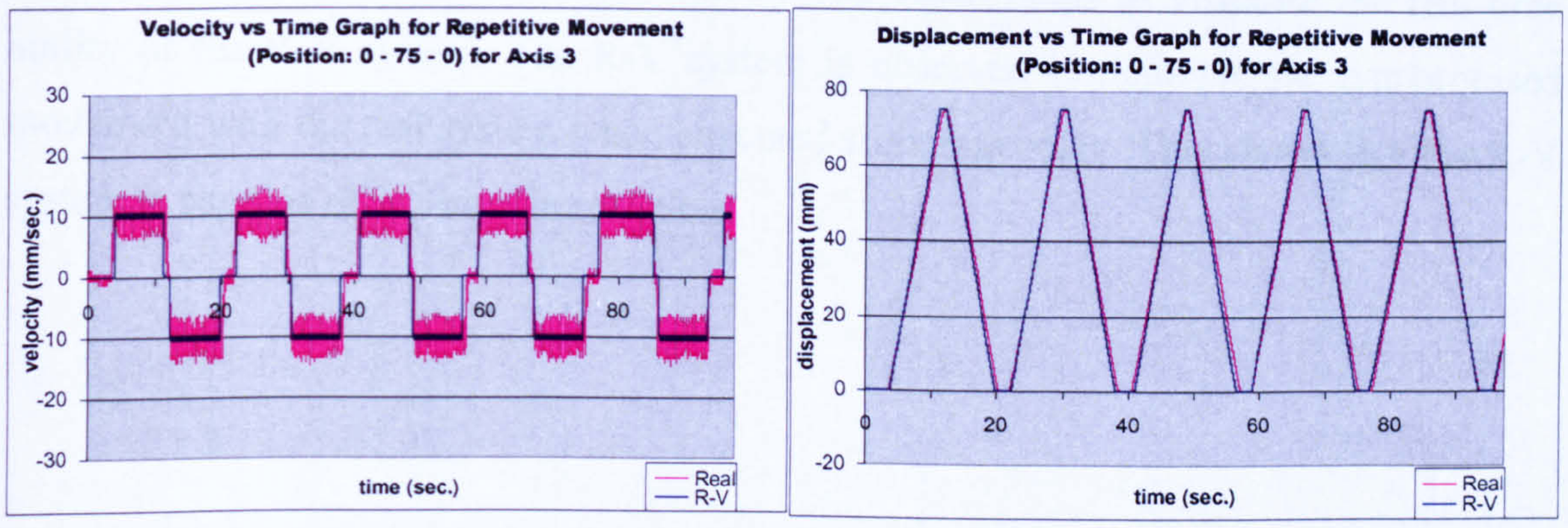
(a) (b)

Figure 7.6 Angular velocity and displacement profile comparison between R-V system and real system for Axis 1 (movement from 0° to 100°)



(a) (b)

Figure 7.7 Angular velocity and displacement profile comparison between R-V system and real system for Axis 2 (movement from 0° to 100°)



(a) (b)

Figure 7.8 Velocity and displacement profile comparison between R-V system and real system for Axis 3 (movement from 0 mm to 75 mm)

The real system data was obtained from the positional data capturing program. Similarly, the positional data from RVME was used for the R-V system. The data from these tests are included in the file *RVtest.pdf* and *REALtest.pdf*[19] on the accompanying CD.

From the graphs it clearly shows that the R-V system motion profiles coincide perfectly with the real machine motion profiles for all the three axes. Furthermore, the encoders produce perfectly clean motion profiles when compared to the real system.

The discrepancies in tuning the PID parameters result in unclean motion profile of the real machine; it can be clearly seen in all three sets of graphs. However the servomotor for Axis 2 is seen to be tuned acceptably. Hence, it shows quite a clean motion profile and overlaps perfectly with the R-V system profile.

Visual analysis of the real-time capability of the R-V system is recorded and the video is included in the accompanying CD. There are three files in the folder *real time R-V*; all are the same but in different video format. The video helps to visualise the real-time ability of the R-V system. The R-V system is observed to demonstrate synchronised movement with the real system when executed simultaneously. This shows that the R-V system is capable of working in real-time.

7.4 TEST CASE III: INTEGRATION OF EXTERNAL APPLICATION WITHIN R-V SYSTEM

The R-V system architecture was designed to being able to incorporate external applications into its system. External application can be develop to interact with the R-V system while the system is in operation. This feature provides the potential of R-V

[19] The **.pdf* file format should be able to be read using Adobe Reader® which is available for download from Adobe website at http://www.adobe.com/.

system to be used as a test platform for various means.

Test case III intended to evaluate further the use of the R-V system in an example application. An application called *Test Scenario* was developed to produce a test scenario for the controller. The *Test Scenario* was a program written in Visual C++ and it present a user with Windows which provide options to fail various components virtually. Figure 7.9 shows the *Test Scenario* user application Windows which allows a user to fail any motor or sensor in the system.

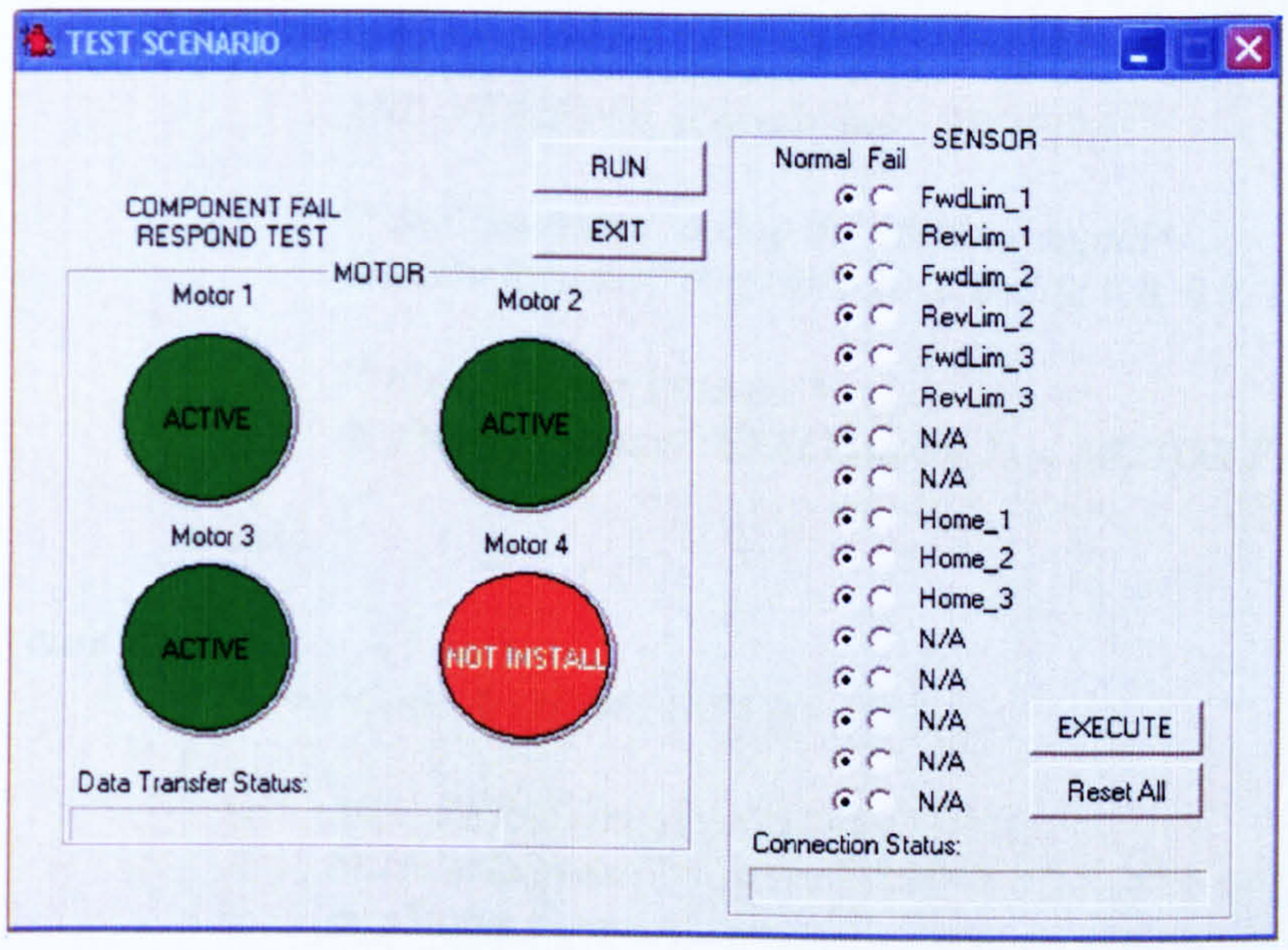


Figure 7.9 *Test Scenario* user application Windows for failing component/s in the R-V demonstrator

The application, in general, provides a way to test the controller program responses when a component fails. It also helps the user to see and visualised what happens to the machine when certain motors or sensors fail. With this application users can test the controller error trapping routines which are normally difficult or even impossible to test

on the real machine when in operation. Whether the error capture routines perform correctly or not is reflected on both the controller program and the Virtual Machine Environment (VME). The controller program error trapping routine is shown in Figure 7.10.

```
switch (axis)
    {
            case 0:
                    if (*errorCode & erFOLLOWING_ERROR)   // check for error
occurrence
                    {
                            /* stops the servomotor*/
                            MIL_VERIFY(m_pNextMove->STOP(0));

                            /* R-V command to stop the encoder signal*/
                            EncoderRequest("mrg16s", 2010,0 ,3.0, 0.0, 0.0, 1.0,0.0)

                            /* display error message*/;
                            m_sStatus.Format("ERROR Joint 1..... MOTOR FAIL");
                    }
                    break;

            case 1:
                    if (*errorCode & erFOLLOWING_ERROR)
                    {
                            MIL_VERIFY(m_pNextMove->STOP(1));
                            EncoderRequest("mrg16s", 2010, 0, 3.0, 0.0, 0.0, 1.0,0.0);
                            m_sStatus.Format("ERROR AXIS Z..... MOTOR FAIL");
                    }
                    break;

            case 2:
                    if (*errorCode & erFOLLOWING_ERROR)
                    {
                            MIL_VERIFY(m_pNextMove->STOP(2));
                            EncoderRequest("mrg16s", 2010, 0,3.0, 0.0, 1.0, 0.0,0.0);
                            m_sStatus.Format("ERROR Joint 2..... MOTOR FAIL");
                    }
                    break;
    }
```

Figure 7.10 Controller program Error trapping routines

The test is also able to show the capability of the RVME to work with multiple clients. The *Test Scenario* application has to be connected to the RVME server before it can manipulate the R-V environment. The connection is made automatically when the program is executed because the server name and port number are already included in the program.

7.4.1 Results and Discussion of Test Case III

Test case III involved deliberately 'Failing' some components whilst the control program was running. The first test involved failure of a servomotor. Using the *Test Scenario* program, the servomotor for Axis 2 was stopped. This is as shown in Figure 7.11. What in fact occurred is that the *Test Scenario* program sends an R-V command to stop the emulated encoder signal for Axis 2.

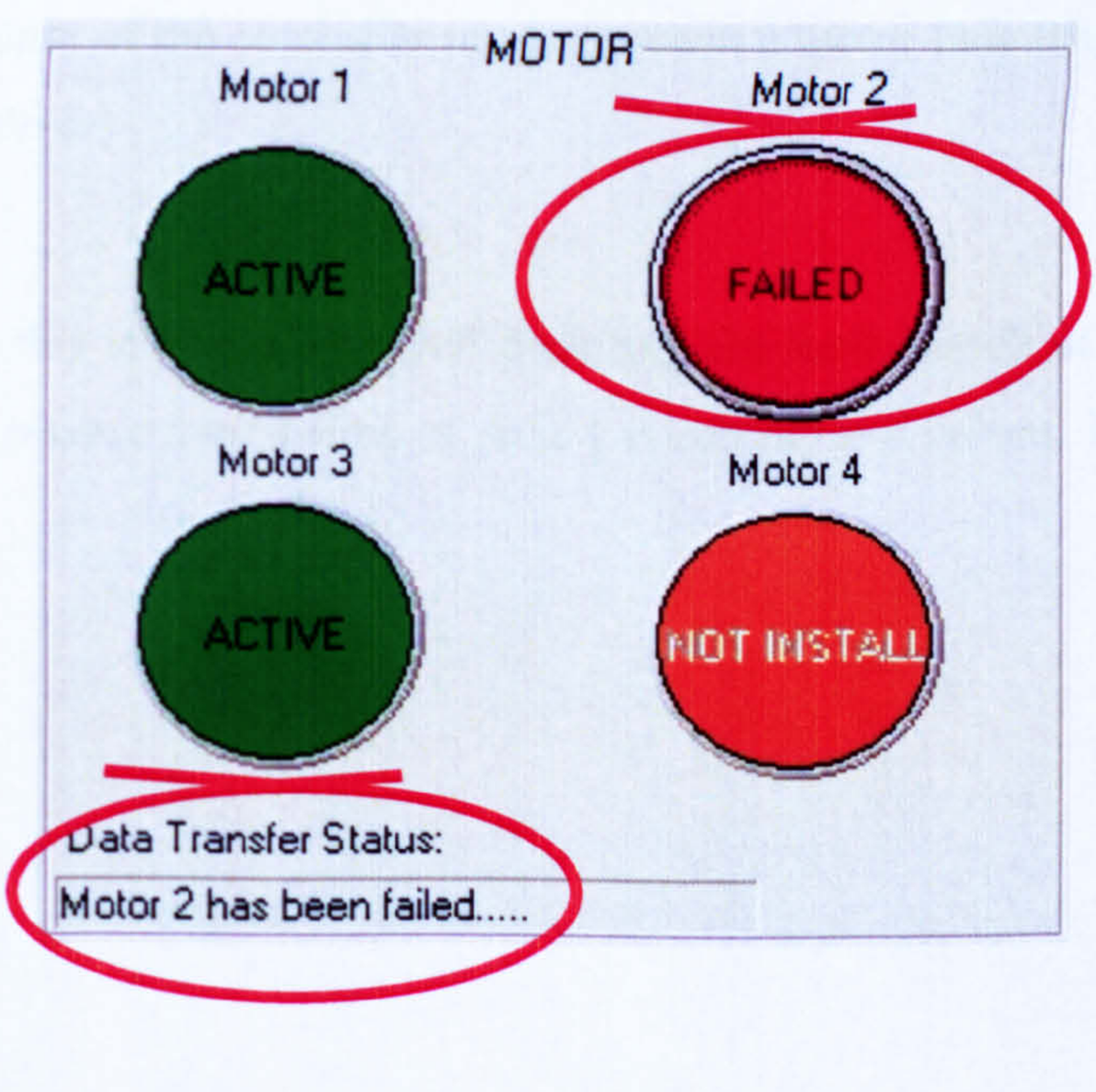


Figure 7.11 Deliberate failure of Axis 2 motor on the R-V system demonstrator using the *Test Scenario* program

The controller program responds with an audible warning and the respective error is displayed as shown in Figure 7.12. The virtual machine in the VME also stops immediately since the error routine was designed to stop all axes immediately when an error occurs. This has shown that the error trapping routine works correctly. The test also showed that the RVME is able to handle more then one client. It also showed that the *temporary client* (i.e. the *Test Scenario* program) supersedes the *fixed client*; the R-V sub-systems.

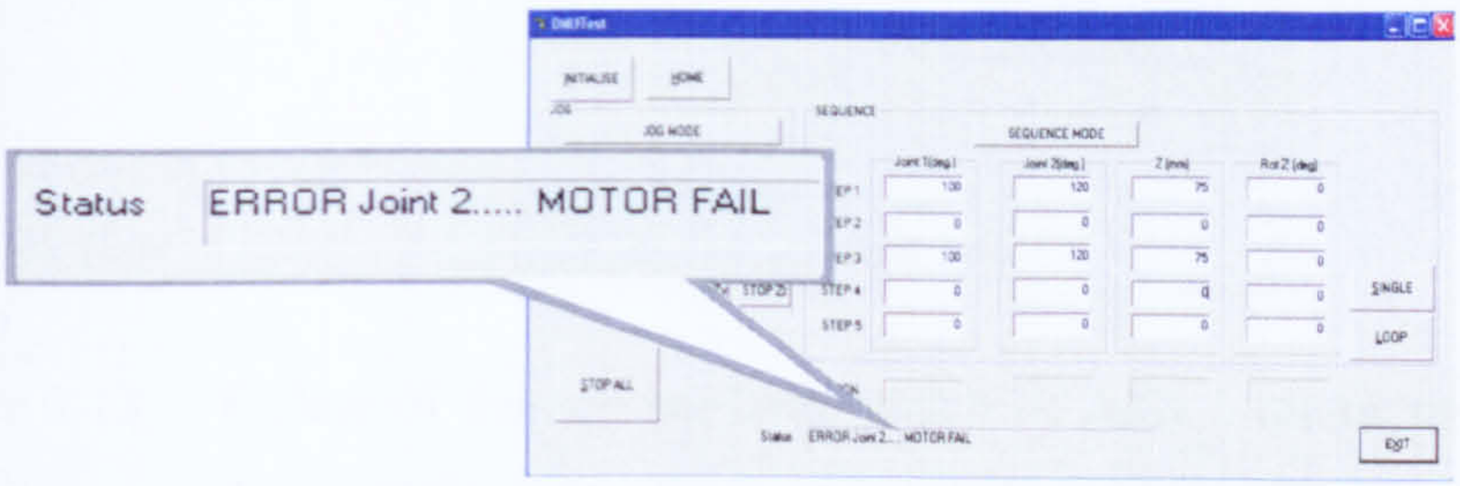


Figure 7.12 Response of the controller program when a motor fails (triggered by the *Axis Error*)

The second stage in this test case involved disabling the limit switch of the machine axis. In this test, the *reverse limit switch* of Axis 1 is subject to a failure. This is shown in Figure 7.13.

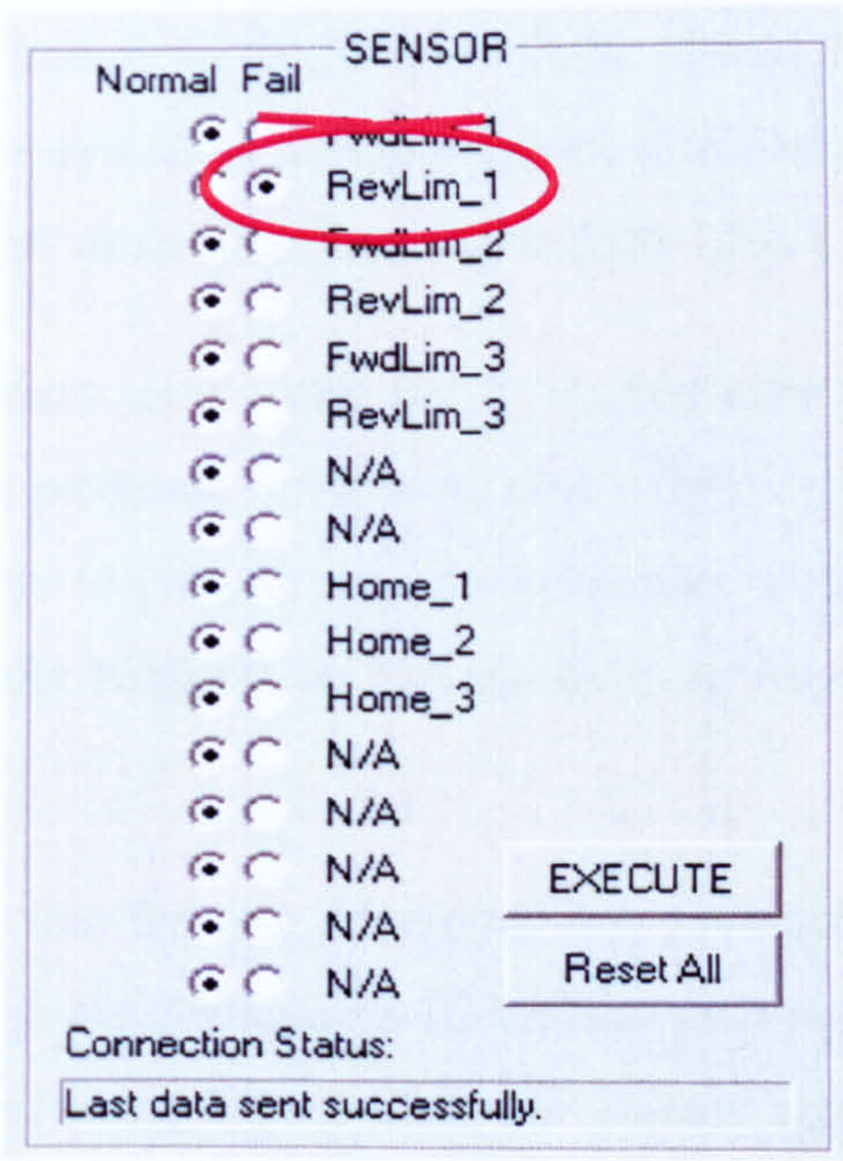


Figure 7.13 Failure of the *reverse limit switch* of Axis 1 motor on the R-V system demonstrator using the *Test Scenario* program

With a functioning *reverse limit switch*, the controller produces an audible warning and displays an error message when it reaches its limit as shown in Figure 7.14. However, when the *reverse limit switch* is disabled and Axis 1 is jogged beyond its reverse limit, no audible warning is heard or error message is displayed.

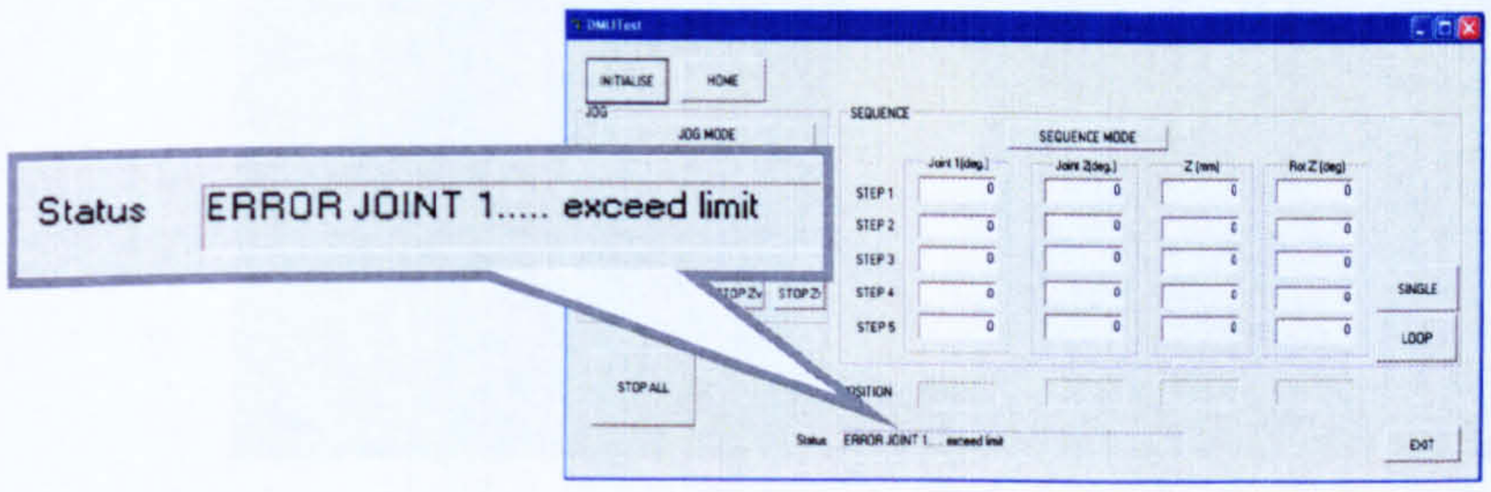


Figure 7.14 Response of the controller program when an axis exceeds its limit (triggered by the *Limit Error*)

The *reverse limit switch* failure can be viewed on the virtual machine in the VME. The respective virtual machine axis turns red showing the machine axis has exceeded its axis limit. Figure 7.15 show the virtual machine where Axis 1 has exceeded its limit.

From these tests it has been shown that the R-V environment is not only capable of being used for controller program verification and validating but the Virtual Machine Environment (VME) helps to provide a machine builder with a capability to visualise and understand what really happens on the machine in response to certain operating conditions.

This test case has shown that the user developed test environment can be incorporated into R-V system. Although the formulated *Test Scenario* program was rather simple, it illustrates the potential of using R-V system for further applications such as virtual testing, virtual commissioning and virtual training.

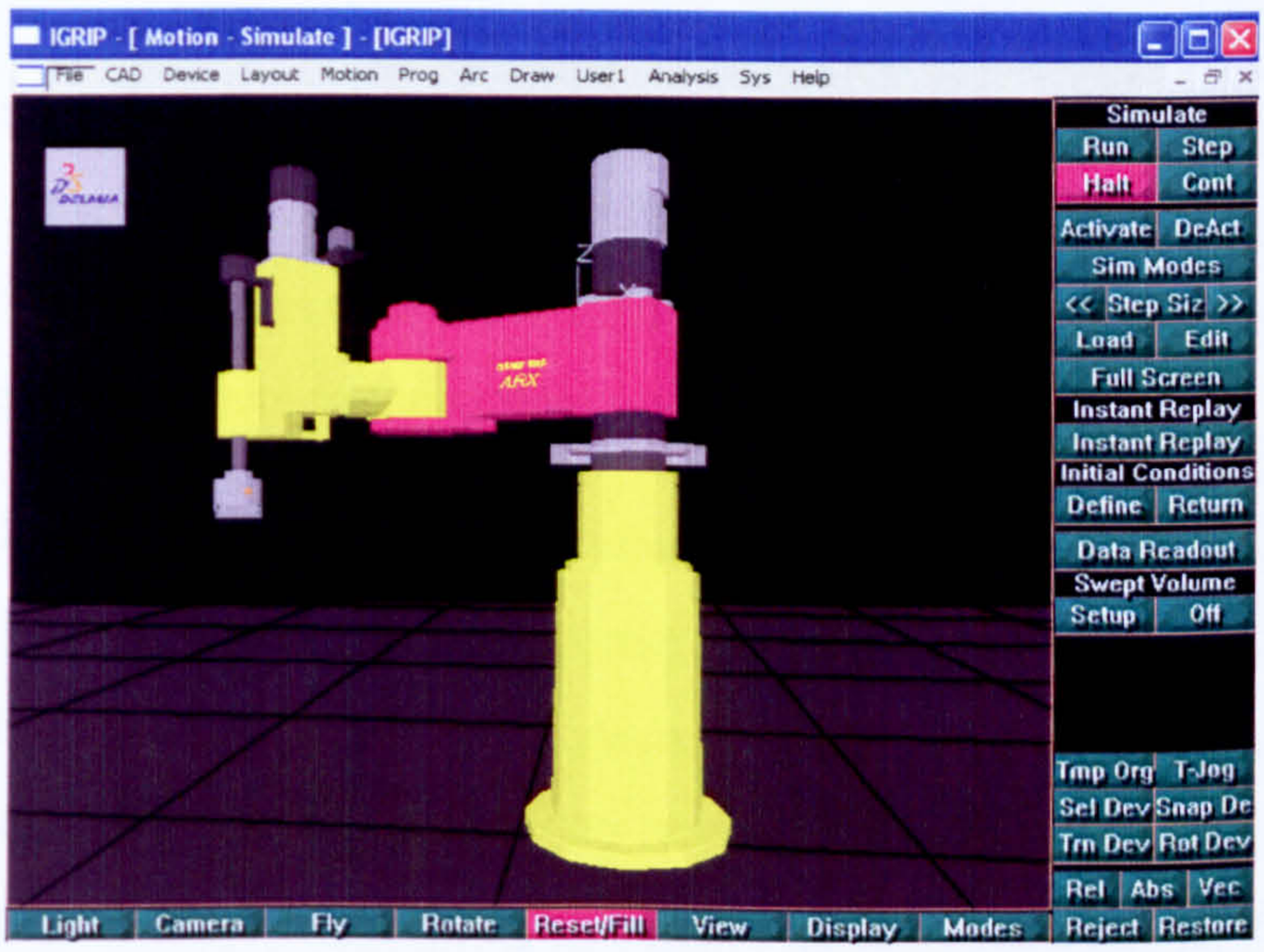


Figure 7.15 An axis which has exceeded its limit turns red in the VME

Test case III also shows the benefit of using simulation environments. Testing component failure on real machine is dangerous and not really an option. Using a machine model for component failure testing not only safe but can be done in variety of ways. Furthermore, there will be no machine downtime and no additional cost.

7.5 FURTHER DISCUSSION

The test cases described in this chapter have shown some of the R-V system capabilities. Two videos are attached with this thesis to give a better understanding of the R-V system and its capabilities. It should be noticed that the IGRIP simulation update rate (maximum 50Hz = 0.02 second) is enough to provide good visualisation of the machine operation. The machine can be seen moving smoothly with no noticeable discrepancy to the actual machine.

All communication between the R-V sub-systems use Ethernet connections; the tests show that the Ethernet (TCP/IP) protocol works well for the R-V system prototype. Even though the R-V system uses more then five (5) Ethernet connections at one time; it doesn't cause any problem to the system operation.

Although only one type of industrial controller (NextMove) is used for evaluation of the demonstrator, the R-V system should not have any problem to work with any other type of controller. This is because the R-V system is based on emulation of input signals which is not dependent on any proprietary standard. The R-V system should also be able to work with any other simulation packages. This is because the R-V system task is to provide real-time machine data to the simulation package. It is up to the software whether it allows external data to be linked to its environment. However, most graphical simulation platforms do provide external interface features.

7.6 SUMMARY

This chapter assesses the effectiveness of the R-V system in supporting machine system design, testing, verification and validation. Three test cases were conducted to evaluate the R-V system in term of its competencies, capabilities and potential.

Test case I has shown that actual machine control program can be formulated within the R-V system environments. Once the program has been verified and validated it can be used almost straight away within the real controller. Test case II shows the real time capability of the R-V system. The test includes the evaluation of the emulated signals to verify whether the hardware is capable or not to produce the required signal accurately and timing. Test case III explores the potential of using R-V system for further applications. The test case also shows the R-V system capability to handle multiple client environments.

Results form the test cases have been presented and discussed in this chapter and are concluded in the following chapter.

CHAPTER 8

CONCLUSIONS

In this thesis a concept and architecture for a Real-Virtuality system (R-V system) has been proposed. The R-V system is intended to help machine designers and machine builders to work with their machine in real-time although the machine itself has not necessarily been built. This new approach based on control signal emulation has been implemented and tested with a prototype environment. Three test cases have been established to demonstrate, validate and verify the proposed architecture. A video sequence is included to demonstrate the R-V system prototype in operation. Results from three test cases include the following findings:

- the proposed architecture has been shown to support offline programming of industrial controllers. Test case I demonstrates how the programmer can visualise in 3-D how the machine will respond. The use of the R-V system provides a useful tool for offline programming, verification and validation of a machine system. It also shows the seamless transfer from the R-V environment to the real environment.

- the R-V system is able to operate in real-time. There was no noticeable difference between the response of the virtual machine and the real machine, namely they are both able to operate in the same time frame.

- the R-V system can be connected to multiple clients which makes it possible to build test scenarios in the Real-Virtually system.

8.1 CONTRIBUTION TO KNOWLEDGE

Contributions to the knowledge base on machine system design made from this research study include:

(i) This research study has contributed a new paradigm in the use of 3-D virtual environments as a platform for industrial controller verification and validation. It offers a new concept for bridging the virtual and real world in the context of machine control system design. The migration from the virtual world to the real world is totally seamless.

(ii) The research study also provides a solution for real time simulation at the mid-level in the machine. Mid-level simulation which involves the simulation of systems with a combination of continuous and discrete operations has not previously been explored widely.

(iii) This new approach has resulted in the formulation of a technique for emulation of quadrature encoder signals. The technique is novel and can be used not only for the R-V system environments but also for servomotor control testing and verification. This is because the encoder emulator is programmable; it can help to provide the desired signals for the control feedback element.

(iv) The research study has also shown that the R-V system can not only be used for machine control system development but can also execute test scenarios. This opens new horizons for the application of virtual environments. The system can increase the levels of confidence for the use of virtual environments for example in system training and commissioning. Earlier research has shown the benefits of virtual commissioning and training but these studies have not resulted in their widespread use for such purposes by industry. This may be the result of a lack of confidence in virtual commissioning and training due to the fact

that the simulation does not represent the real time operation of the actual system. The R-V approach may well eliminate this hesitation since the research demonstrates that there is no difference at all between the real system and the operation of the virtual environment. With its real-time capability, the R-V system can also be used for operator training without having to have access to the real system. R-V systems can also be used in education which will see more students having access to "real machines" but at the same time less money and space are required.

8.2 FUTURE DIRECTIONS

(i) The present implementation of the RTEE only provides emulators for two types of signal; discrete signals and continuous digital signals (encoders). A complete RTEE should also include emulation for continuous analogue signals. A study could also be conducted on how to emulate absolute encoders signal (if this is seen as an important extension of the system capabilities).

(ii) In the demonstrator, the test scenario application is fairly basic. A more demanding test scenario could be devised to further test the R-V system. Furthermore, at present the R-V system is only based on a single machine representation. Applying the R-V system on a larger scale and a more complex system would allow a more elaborate evaluation to be conducted.

(iii) Since the R-V system has been proven to be able to mimic a real machine in real time, further studies could be done on possibilities for incorporating virtual reality hardware into the environment. Such an approach would allow users to have a more realistic experience of the virtual machine. Furthermore, some simulation software such as IGRIP provides virtual

reality capabilities.

(iv) In R-V system, the controller program has been shown to operate correctly with the associated signal connections. The R-V system could be developed further with the ability to provide system wiring and connection diagrams, and eventually automatic system documentation.

(v) Developing and real time simulation system for a fast and complex manufacturing environment such as packaging and inspection system will be very challenging. This is because there are some constraints that need to be addressed. The simulation software for example, has very limited refresh rate (IGRIP maximum refresh rate is 50 Hz). This might cause a problem when simulating a very fast operation.

(vi) Real-time simulation should not only focus on machines or manufacturing systems, a complete study should also include real time human behaviour and responses. Integration between the two will definitely enhance the realisation of true real time virtual machine/manufacturing evaluations.

REFERENCES

ADOLFSSON, J., OLOFSGÅRD, P., MOORE, P.R. and PU, J. (1998), 3-D graphical simulation for agile modular machine systems, *Proceedings of Mechatronics 98*, Skövde, Sweden, pages 867-872.

AUINGER, F., VORDERWINKLER, M. and BUCHTELA, G. (1999), Interface driven domain-independent modeling architecture for "soft-commissioning" and "reality in the loop", *Proceedings of the 1999 Winter Simulation Conference*, pages 798-805.

BANKS, J. (2000), *Getting started with AutoMod*, AutoSimulations, Inc., Utah, USA.

BANKS, J. (1999), Introduction to simulation, *Proceedings of the 1999 Winter Simulation Conference*, pages 7-13.

BANKS, J. (1998), *Handbook of simulation: principles, methodology, advances, applications, and practice*, John Wiley and Sons, New York, USA.

BANKS, J., CARSON II, J.S., NELSON, B.L. and NICOL, D.M. (2000), *Discrete-event system simulation*, 3rd Ed. , Prentice-Hall, Upper Saddle River, New Jersey, USA.

BARACOS P., MURERE, G., RABBATH, C.A. and JIN, W. (2001), Enabling PC-based HIL simulation for automotive application, *Proceedings of the IEEE International Electric Machines and Drives Conference (IEMDC '01)*, Cambridge, Massachusetts, USA

BARNES, M. (1996), Virtual reality and simulation, *Proceedings of the 28th Conference on Winter Simulation*, Colorado, California, pages101-110.

BATESON, R.N. (1999), *Introduction to control system technology*, 6th Ed., Prentice Hall, Upper Saddle River, New Jersey, USA.

BENNETT, S. (1994), *Real-time computer control: an introduction*, 2nd Ed., Prentice Hall, New York, USA

BERNHARDT, R., SCHRECK, G. and WILLNOW, C. (1995), Realistic robot simulation, *Computing and Control Engineering Journal*, 6(4), pages 174-176.

BERNHARDT, R., SCHRECK, G. and WILLNOW, C. (2001), Development of virtual robot controllers and future trends, *Proceedings of the 6th IFAC Symposium on "Cost-oriented Automation"*, 8-9 October, Berlin, Germany.

BIRLA, S., FAULKNER, D., MICHALOSKI, J., SORENSON, S., WEINERT. G. and YEN, J. (2001), Reconfigurable machine controllers using the OMAC API, *Proceedings of the CIRP 1st International Conference on Reconfigurable Manufacturing*, Ann Arbor, Michigan, USA.

BLUEMEL, E., HINTZE, A., SCHULZ, T., SCHUMANN, M. and STUERING, S. (2003), Virtual environments for the training of maintenance and service tasks, *Proceedings of the 2003 Winter Simulation Conference*, pages 2001-2007.

BONFATTI, F., MONARI, P.D. and SAMPIERI, U. (1999), *IEC 1131-3 programming methodology*, 1st ed., CJ International, Fontaine, France.

BONISSONE, P.P, BADAMI, V., CHIANG, K. H., KHEDKAR, P.S. AND MARCELLE, K. W. (1995), Industrial applications of fuzzy logic at General Electric, *Proceedings of the IEEE 833*, pages 450-465.

CARO, D. (2006), Industrial Ethernet du jour, *Manufacturing Automation Magazine*, November/December Issue, http://www.automationmag.com [Accessed: 30 June 2007]

CELL CONTROL (2004), *Cell Control brochure*, http://www.delmia.com/solutions/pdf /cell_control.pdf. [Accessed: 24 October 2004]

CHARLES, M. and LEONG, S. (2001), The expanding role of simulation in future manufacturing, *Proceedings of the 2001 Winter Simulation Conf.*, pages 1478- 1486.

CHICK, S., SÁNCHEZ, P. J., FERRIN, D. and MORRICE, D. J. (2003), The future of the simulation industry, Proceedings of the 2003 Winter Simulation Conference, pages 2033-2043.

CHENG, F. S, (2000), A methodology for developing robotic workcell simulation models, *Proceedings of the 2000 Winter Simulation Conference, pages 1265-1271*

CHO, M.S. and KIM, T.G. (2001), Real time simulation framework for RT-DEVS models, *Transaction of Society for Modeling and Simulation International*, Vol. 18, No. 4, pages 203-215.

CHONG, S.K. (2002a), *Component-based runtime support framework for agile manufacturing machinery*, Ph.D Thesis, De Montfort University, UK.

CHONG, S.K., Pu, J., MOORE, P.R., WONG, C.B., CHEN, X. and NG, A.H.C. (2002b), Component-based runtime support framework for agile modular manufacturing machinery, *Proceedings of Mechatronics 2002*, Enschede, Netherland.

CHOUINARD, J. (2000), What open control was meant to be, *Control Solutions*, October Issue.

CHUEN, N.H. (2003), *An intergrated design, simulation and programming environment for modular manufacturing machine systems*, Phd. Thesis, De Montfort University, UK.

CLARK, G. M. (1996), Introduction to manufacturing applications, *Proceedings of the 1996 Winter Simulation Conference*, Coronado, pages 85-92.

DANIELSSON, F. (2002), *Off-line programming, verification and optimisation of industrial control system*, Ph.D Thesis, De Montfort University, UK.

DANIELSSON F. and MOORE, P. (2003), Validation, off-Line programming and optimisation of industrial control logic, *Mechatronics*, Vol. 13. Issue 6, pages 571–585.

DE VIN, L. J., OSCARSSON, J., NG, A., JÄGSTAM, M. and KARLSSON, T. (2004), Manufacturing simulation: good practice, pitfalls, and advanced applications, *IMC21 Conference*, Limerick, Ireland, pages 156-163.

DILTS, D.M., N.P. BOYD, and H.H. WHORMS. (1991), The evolution of control architectures for automated manufacturing systems, *Journal of Manufacturing Systems*, Vol. 10, No. 1, pages 79-93.

DJIEV, S. (2004), *Industrial networks for communication and control, Lecture Notes, Faculty of Automation*, Technical University – Sofia, January, http://anp.vmei.acad.bg/djiev/PDF%20files/NCPCS_Lecture1.pdf [Accessed: 02 November 2004].

DOBSON, B. (2004), More Choice In controls with open systems, *Machinery Update*, November/December Issue, PPMA Ltd., Surrey, UK.

DYNASIM (2004), *Case study: real-time simulation of industrial robots with Dymola*, Dynasim, http://www.dynasim.se/www/CaseStudyDLR_robot.pdf. [Accessed: 4 May 2004]

EPC (2007), *Product descriptions*, Encoder Products Company (EPC), http://www.encoder.com/products.html [Accessed: 1 March 2007]

ERIKSSON, P. (1996). *Enhancements in virtual robotics*, PhD. thesis, De Montfort University, Leicester, UK.

FARAHMAND, K. (2000), Using simulation to support implementation of flexible manufacturing cell, *Proceedings of the 2000 Winter Simulation Conference*, pages 1272 -1281.

FLETCHER, J.D. (1996), Does the stuff work? Some findings from applications of technology to educational and training, *Proceedings of Conference on Teacher Education and the Use of Technology Based Learning Systems.*

FREEDMAN, S. (1999), An overview of fully integrated digital manufacturing technology, *Proceedings of the 1999 Winter Simulation Conference*, pages 281-285.

FRITZ, R., HORVATH, S., ORELLANA, C. and WOHLERS, J. (2002), Real-time control of a discrete-event simulation using an external controller: a feasibility study with application to mail distribution systems, *2002 IEEE Systems and Information Design Symposium*, University of Virginia, USA.

GELGELE, H.L. (2002), *Hybrid intelligent systems in manufacturing*, PhD. Thesis, Norwegian University of Science and Technology (NTNU), Norway.

GEORGIEW, K. (1995), Airbus flight control system under test, *dSPACE News*, Vol.4, No. 2.

GOMES DE SÀ, A. and ZACHMANN, G. (1999), Virtual reality as a tool for verification of assembly and maintenance processes, *Computer & Graphics*, Elsevier, Vol. 23, No. 3, pages 389-403.

GROOVER, M. P. (2001), *Automation, production systems, and computer-integrated manufacturing*, 2nd Ed., Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA.

GRUND, C. (1995), First HIL testbench for wheel slip control installed at Audi, *dSpace News*, Vol.4, No. 2.

HANISH, H. M. (1997), THIEME, J. L. and WIENHOLD, O. (1997), Modeling of PLC behaviour by means of timed net condition/event systems, *IEEE Symposium on Engineering Technologies and Factory Automation*, Carlifonia, USA, pages 391-396.

HANSELMANN, H. (1996), Hardware-in-the-loop simulation testing and its integration into a CACSD Toolset, *IEEE International Symposium on Computer-Aided System Design*, USA.

HARROLD, D. (2002), Controllers: heartbeat of production, *Control Engineering*, January Issue.

HASSAPIS, G. (2000), Soft-testing of industrial control systems programmed in IEC1131-3 languages, *ISA Transactions*, 39, pages 345-355.

HAYES, S. (2000), The advantages of PC-based control systems, *Technical Background Article*, Hayes Control Systems.

HITOMI, K. (1996), *Manufacturing systems engineering: a unified approach to manufacturing technology, production management, and industrial economics*, 2nd Edition, Taylor & Francis, London, UK.

HODGSON, J. and KATZ, M. (2000), Using a portable simulation structure with emulation for offline testing, *Auto Simulations Symposium 2000.*

HOLST, L. (2001), *Integrating discrete-event simulation into the manufacturing system development process*, PhD. Thesis, Lund University, Sweden.

HOWE, D. (2004), *The free on-line dictionary of computing*, Editor Denis Howe, http://www.foldoc.org/ [Accessed: 06 June 2004].

IEC (1993), *Programmable controller – part3: programming languages*, International Standard, IEC 1131-3:1993(E).

IRMA (2004), IRMA : *A worldwide IMS virtual reality project for the industrial enterprise*, http://www.project-irma.com/ [Accessed: 04 May 2004].

IWATA, K., ONOSATO, M., TERAMOTO, K. and OSAKI, S. (1997), Virtual manufacturing systems as advanced information infrastructure for integrating manufacturing resources and activities", *Annals of the CIRP*, Vol. 46/1.

JACK, H. (2004), *Dynamic system modeling and control*, Draft Ver. 2.6, Unpublished.

JAIN, S., CHOONG, N.F., AYE, K.M. and LUO, M. (2001), Virtual factory: an integrated approach to manufacturing systems modeling, *International Journal of Operations & Production Management*, Vol. 21 No. 5/6, 2001, pp. 594-608.

JOINES, J. A., BARTON, R. R., KANG, K. and FISHWICK, P. A. (2000), Simulation in the future, *Proceedings of the 2000 Winter Simulation Conference*, pages 1568-1576.

KAMEN, E.W. (1999), *Industrial controls manufacturing,* Academic Press, London, UK.

KANAI, S. and KISHINAMI, T. (1999), A virtual verification environment for the sequence control system using VRML and JAVA, *Proceeding of DETC '99/CIE.*

KATZ, R., MIN, B.K. and PASEK, Z. (2000), Open architecture control technology trends, *ERC/RMS Report 35.*

KINSELLA, J. (1998), Open automation: a perspective on connection, *Manufacturing Engineering*, July Issue.

KLINGSTAM, P. (1998), A methodology for development of flexible Shop-Floor Control Systems: Simulation Tools Support, *Proceedings of the CIRP International Seminar on Manufacturing Systems*, California, US.

KOLLA, S., MICHALOSKI, J. and RIPPEY, W. (2002), Evaluation of component-based reconfigurable machine controllers, *Proceedings of the World Automation Congress (WAC) 2002*, as part of the International Symposium on Robotics & Applications (ISORA), Orlando, Florida, USA.

KOREN, Y. (1998), Open-architecture controllers for manufacturing systems, *Open Architecture Control Systems – Summary of Global Activity*, ITIA Series, Vol. 2, pages 15-37.

LAW, A. M. and KELTON, W. D. (1991), *Simulation modeling and analysis*, 2nd Ed., McGraw-Hill, New York, USA.

LeBARON, T. and THOMPSON, K. (1998), Emulation Of A Material Delivery System, *Proceedings of the 1998 Winter Simulation Conference*, pages 1055–1060.

LEWIS, R.W. (1998), *Programming industrial control systems using IEC1131-3*, Vol.50 of Control Engineering Series, The Institution of Electrical Engineers, Stevenage, UK.

LINJAMA, M., VIRVALO, T., GUTAFSSON, J, LINTULA, J., AALTONEN, V. and KIVIKOSKI, M. (2000), Hardware-in-the-loop for servo system controller design, tuning and testing, *Microprocessor and Microsystem (24)*, Elsevier, pages 13-21.

LU, Z., CHAI, J. and WANG, X. (2003), An investigation into the use of hardware-in-the-loop simulation testing for brushless DC motor drive of hybrid electric vehicle, *Sixth International Conference on Electrical Machines and Systems (ICEMS 2003)*, Vol. 2, pages 588 – 591.

MANSOOR, S.P., JONES, D.I., BRADLEY, D.A, ARIS, F.C. AND JONES, G.R., (2003), Hardware-in-the-loop simulation of a pumped storage hydro station, *International Journal of Power and Energy Systems*, Vol. 23, No. 2.

MASKELL, B. (2001), Insight from industry – the age of agile manufacturing, *Supply Chain Management: An International Journal*, Vol. 6, No. 1, pages 5-11.

MATHIAS, D. and HELLMANN, R. (1999), Boeing implements HMI, *Manufacturing Engineering*, pages 78-82.

McHaney, R. (1989), Bridging the Gap: Transferring logic from a simulation into an actual system controller, *Proceedings of the 1989 Winter Simulation Conference*, pp. 848-858.

McGREGOR, I. (2000), Reduce your commissioning costs through emulation, *AutoFlash*, Published by AutoSimulation, a Division of Brooks Automation, Vol.13, No.9.

MCKAY, M.A. (2002), Combining PC control and HMI, *InTech Magazine*, ISA, Decem ber Issue.

McLEAN, C. and LEONG, S. (2001), The expanding role of simulation in future manufacturing, *Proceeding of the 2001 Winter Simulation Conference*, pages 1478-1486.

MECKLENBERG, K. (2001), Seamless integration of layout and simulation, *Proceedings of the 2001 Winter Simulation Conference*, pages 1487-1491.

MIN, B. K., HUANG, A., PASEK, Z. J., YIP-HOI, D., HUSTED, F. and MARKER, S. (2002), Integration of real-time control simulation to virtual manufacturing environment, Journal *of Advanced Manufacturing Systems*, Vol. 1, No. 1, pages 67-87.

MINTCHELL, G.A. (1998), PLCs adapt, but will they withstand assault of PCs?, *Control Engineering*, May Issue.

MINTCHELL, G.A. (1999), Graphic interface are programmers' friends, *Control Engineering International*, November/December Issue.

MODERN. P. J., STEDMON, A.W., D'CRUZ, M. and SHARPLES, G. J. (2003), the factory of the future? the integration of virtual reality for advanced industrial applications, *The 10th International Conference on Human-Computer Interaction*, Heraklion, Crete, Greece.

NATIONAL INSTRUMENT (2007), *Understanding Servo Tune* http://zone.ni.com/devzone/cda/tut/p/id/2923 [Accessed: 6 June 2007]

NELSON, N.J. (2000), Industrial control languages: Forth vs. IEC61131, *16th EuroForth Conference*, Prestbury, Cheshire, UK.

NEMATRON (2004), *PC-based control technology: an in-depth look*, http://www.nematron.com/Library/pdf/Nematron_PC-based_Control.pdf [Accessed: 20/12/2004]

NEVINS, J.L. and WHITNEY, D.E. (1989), *Concurrent design of products and processes*, McGraw-Hll, New York, USA.

NGM (1997), Next-generation manufacturing - a framework for action, *Vol.I - Summary Report*, NGM Project Office, Bethlehem, Pennsylvania, USA.

NWOKE, B.U. and NELSON, D.R. (1993), Overview of computer simulation in manufacturing, *Industrial Engineering*, July, pages 43-45.

OLOFSGÅRD, P., NG, A.H.C.,. MOORE, P.R, PU, J. and WONG, C.B. (2002), VIR-ENG integration platform, *Proceedings of Mechatronics 2002*, Enschede, Netherland.

OLSSON, G. and PIANI, G. (1992), C*omputer systems for automation and control*, Prentice Hall International, UK.

PAGE, E. H. (1994), *Simulation modeling methodology: principles and etiology of decision support*, PhD.Thesis, Virginia Polytechnic Institute and State University, USA.

PARKER MOTION & CONTROL (1995), Machine control, *Engineering Reference*, Parker Hannifin Corp., California, USA.

PEGDEN, C.D., SHANNON R.E. and SADOWSKI, R.P. (1995), *Introduction to simulation using SIMAN*, 2nd Ed., McGraw-Hill Inc., New York, USA.

PETTERSSON, L., ADOLFSSON, J., NG, AHC. and DE VIN, L.J. (2007), Cell Monitoring and Diagnostics Using Computer Aided Robotics, *Proceedings of 40th CIRP International Seminar on Manufacturing Systems*, Liverpool UK.

PFEIFFER, A, KÁDÁR, B., MONOSTORI L. (2003), Evaluating and improving production control systems by using emulation, *IASTED International Conference on Applied Simulation and Modelling (ASM 2003)*, Spain.

PIDD, M. (2004a), *Computer simulation in management science*, 5th Ed., John Wiley & Sons, Chichester, UK.

PIDD, M. (2004b), Simulation worldviews - so what?, *Proceedings of the 2004 Winter Simulation Conference*, pages 288-292.

POWERSOFT (2003), An introduction to IEC 61131, *Powersoft White Paper*, Powersoft Control Systems Ltd., Hampshire, UK.

PRASAD, B. (1996), *Concurrent engineering fundamentals: integrated product and process organization*, Volume I, Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA.

PROCTOR, F.M., DAMAZO, B., YANG, C. AND FRECHETTE, S. (1993), *Open architectures for machine control*, NISTIR 5307, National Institute of Standards and Technology, Gaithersburg, Maryland, USA.

PU, J. and MOORE, P. R. (1998), Towards Paradigm Shift in Machine Design and Control, *Proceeding of the 6th UK Mechatronics Forum International Conference*, Skövde, Sweden, pages 23-30.

RAFE, G.E., NEEDY, K.L., BIDANDA, B. and SCHMIDT, T.M. (2001), Delivering continuous manufacturing education and training via an internet-based distributed virtual laboratory, *Frontier in Education (FIE) 2001 Conference*, Reno, Nevada, USA

REALSIM (2004), *Realtime simulation*, DLR Institute of Robotics and Mechatronics, http://www.dlr.de/rm/en/DesktopDefault.aspx/tabid-223/351_read-2758/ [Accessed: 07 May 2004]

REED, D. (2003), *Applying the OSI seven layer network model to information security*, SANS GIAC GSEC Practical Assignment version 1.4b Option One.

REHN, G. D., LEMESSI, M., VANCE, J. M., and DOROZHKIN, D. V. (2004), Integrating operations simulation results with an immersive virtual reality environment, *Proceedings of the 2004 Winter Simulation Conference*, pages 1713-1719.

ROCHE, P. (1993), Hardware-in-the-loop simulation for electrical drive testing, University College Cork, Ireland, *dSpace News*, May 1993.

ROHRER, M. (2002), *Modeling, analysis and return on investment*, Michigan Simulation User Group Presentation, via http://www.m-sug.org/ [Accessed: 21 January 2004].

SAFARIC, R., PARKIN, R. M., CZARNECKI, C. and CALKIN, D. (2001), Virtual environment for telerobotics, *Integrated Computer-Aided Engineering*, 8(2): 95-104.

SCHEIBER, S. F. (2003), Emulation and control system assurance, *Control Engineering*, November Issue.

SCHIESS, C. (2001), Emulation: debug it in the lab-not on the floor, *2001 Winter Simulation Conference*, IEEE.

SCHLUDERMANN, H., KIRCHMAIR, T. and VORDERWINKLER, M. (2000), Soft-commissioning: hardware-in-the-loop-based verification of controller software, *Proceedings of the 2000 Winter Simulation Conference*, pages 893-899.

SCHWIENHORST, K. (2002), Why virtual, why environments? Implementing virtual reality concepts in computer assisted language learning, *Simulation and Gaming*, 33, pages 196-209.

SHACKLEFORD, W.P. and PROCTOR, F.M. (2000), Use of open source distribution for a machine tool controller, *Proceedings of the International Society for Optical Engineering (SPIE) Photonics East Conference*, Boston, Massachusetts, USA.

SHERIDAN, T. B. (1993), Space teleoperation through time delay: review and prognosis", *IEEE Transactions on Robotics and Automation*, vol. 9, no. 5,, pages 592-606.

SHERMAN, W. R. (1997), Experiences with virtual reality applications, *Proceedings of 24th Annual Conference on Computer Graphics and Interactive Techniques*, pages 473-476.

SIEGWARD, R., WANNAZ, C., GARCIA, P. and BLANK, R. (1998), Guiding mobile robots through the web, *Proceedings of the International Conference Intelligent Robots and Systems (IROS)*, Workshop on Web Robot.

SPERLING, W. and LUTZ, P. (1997), Designing applications for an OSACA control, *Proceedings of the International Mechanical Engineering Congress and Exposition*, Dallas USA, pages: 16-21.

STENERSON, J. (2003), Industrial Automation and Process Control, Prentice-Hall, Inc., Upper Saddle River, New Jersey, USA.

STONE, R.J. (1995), The reality of virtual reality, *World Class Design to Manufacture*, 2: 11-17.

STONE, R.J. (1998), Virtual reality: definition, technology and selected engineering applications overview", *Mechatronics Forum Paper*, May 1998.

SUH, N.P. (1990), *Principles of design*, Oxford University Press, UK.

SUNDBERG, M., NG, AHC., ADOLFSSON, J. and DE VIN, L.J. (2006), *Simulation Supported Service and Maintenance in Manufacturing, Proceedings IMC23*, Jordanstown, UK, pages 559-566.

TECNOMATRIX (2003), eM-PLC and STEP 7 Professional, *eMPower for Manufacturing Process Management*, Tecnomatix Technologies Ltd.

TECNOMATRIX (2004), *Tecnomatix realising your manufacturing vision*, www.tecnomatix.co.uk/downloads/products_eM-PLC.pdf [Accessed: 28 October 2004].

TEMELTAS, H., GOKASAN, M., BOGOSYAN, S. and KILIC, A. (2002), Hardware in the loop simulation of robot manipulators through Internet in mechatronics education, *2002 IEEE 28th Annual Conference of the Industrial Electronics Society (IECON 02)*, Vol. 4, pages 2617 – 2622.

THE AMERICAN HERITAGE (2000), *Dictionary of the English language*, 4th Ed., Houghton Mifflin Company, via Dictionary.com (www).

VACHTSEVANOS, G. (1998), *Hierarchical control*, Handbook of Fuzzy Computation, Editors in Chief E. Ruspini, P.Bonissone and W. Pedrycz, Institute of Physics Publishing, pages F 2.2:42-53.

VAN DOREN, V. (2003), Test your control system with simulation, *Control Engineering*, September Issue.

VEDAPUDI, S. (2001), Using MCM to do emulation of a car assembly line, *Brooks Automation Symposium '01*.

VERNADAT, F. (1996), *Enterprise modeling and integration*, 1st Ed., Chapman & Hall.

VERSTEEGT, C. and VERBRAECK, A. (2002), The extended use of simulation in evaluating real-time control systems of AGVs and automated material handling systems, *Proceedings of the 2002 Winter Simulation Conference*, pages 1659-1666.

VIR-ENG (2001), Integrated design, simulation and distributed control of agile modular manufacturing machinery (VIR-ENG) 25444, *VIR-ENG Final Report*, June.

WANG, B. (editor) (1997), *Integrated product, process and enterprise design. manufacturing systems engineering*, 1st Edition, Chapman & Hall.

WESTON, R. H. (1998), The importance of holistic model driven manufacturing system, *ImechE*.

WHITMAN, L. E., JORGENSEN, M., HATHIYARI, K. and MALZAHN, D. (2004), Virtual reality: its usefulness for ergonomic analysis, *Proceedings of the 2004 Winter Simulation Conference*, pages 1740-1745.

WILLS, L., KANNAN, S., HECK, B., VACHTSEVANOS, G., RESTREPO, C., SANDER, S., SCHRAGE, D. and PRASAD, J.V.R. (2000), An open software infrastructure for reconfigurable control systems, *Proceedings of the American Control Conference*, Chicago, Illinois, USA

WU, B. (1994), *Manufacturing systems design and analysis: context and techniques*, Chapman & Hall, London, UK.

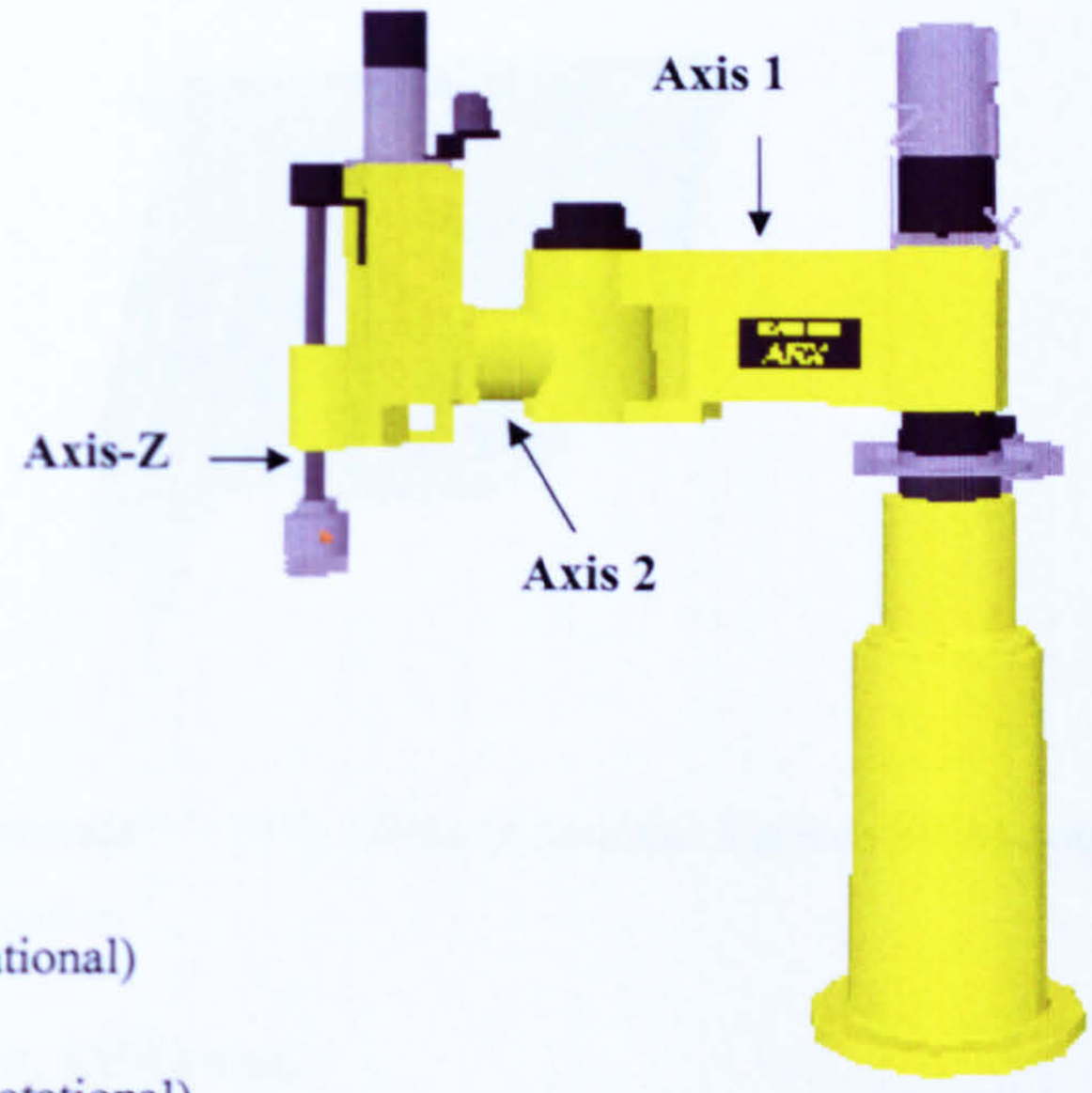
YIEN, T. J. and TSENG, M. M. (1997), *Manufacturing systems design: a review of state-of-the-art methodologies*, Chapter 13, pages 392-434, In Wang (1997).

ZUPANCIC, B. (1998), Extension software for real-time control system design and implementation with MATLAB-SIMULINK, *Simulation Practice and Theory*, Vol. 6 No. 8, pages 703-719.

ZWEGERS, A. (1998), On systems architecting: a study in shop floor control to determine architecting concepts and principles, PhD Thesis, Technische Universiteit Eindhoven, Netherland.

APPENDIX A – ROBOT AND CONTROLLER SPECIFICATIONS

SILVER REED ARX ROBOT FEATURES



Axis 1: First joint (rotational)

Axis 2: Second joint (rotational)

Axis Z: Third joint (translational)

Sensor 1: Reverse limit switch Axis 1

Sensor 2: Forward limit switch Axis 1

Sensor 3: Reverse limit switch Axis 2

Sensor 4: Forward limit switch Axis 2

Sensor 5: Upper limit switch Axis Z

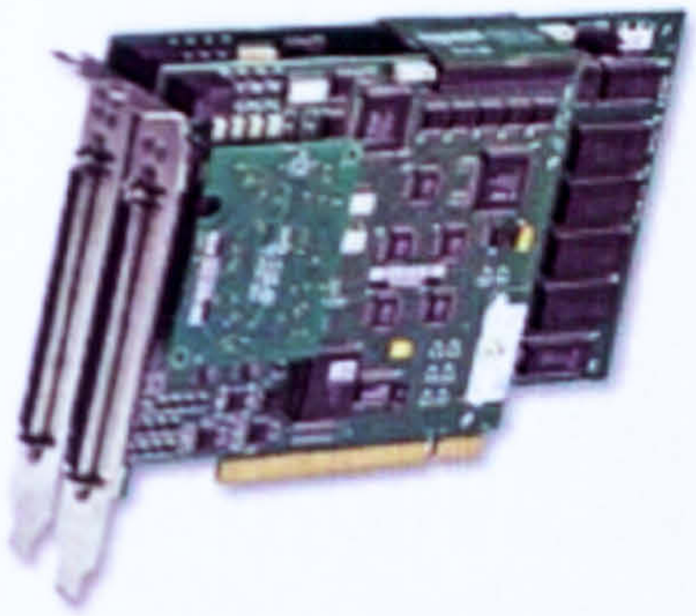
Sensor 6: Lower limit switch Axis Z

Sensor 7: Home switch Axis 1

Sensor 8: Home switch Axis 2

Sensor 9: Home switch Axis Z

NEXTMOVE CONTROLLER SPECIFICATIONS



4 x Servo-amplifier demands +/- 10V, 12-bit or optional 3 phase PWM outputs for direct motor communication

4 x incremental encoder, 8MHz max.

4 x Stepper direction and boost signals to control stepper motor, configurable as 3 phase PWM outputs for direct commutation of 3 phase AC motors

24 digital inputs, opto-isolated PNP or NPN, 12-24V

12 digital outputs, opto-isolated PNP or NPN, 12-24V, 350 mA max.

Four 12 bit analogue differential inputs, software switch-able to 8 single-ended inputs

16-bit ISA bus interface via 2 Kbyte dual ported RAM

512 Kbyte Static RAM

Control Area Network at 1 Mbit per second

12 MBaud serial communications for high speed data transfer

APPENDIX B – IGRIP OPCODE SPECIFICATION

IGRIP OPCODE SPECIFICATION

Opcode	Data
10.0	Joint value information (All)

Word	Data
1	Number of joints to be updated.
2	Joint 1 data
3	Joint 2 data
4	Joint 3 data

Opcode	Data
20.0	Joint value information (Single)

Word	Data
1	Joint Number to Update (1..dof)
2	Joint data

Opcode	Data
30.0	T6 information

Word	Data
1	Configuration
2-17	4x4 [row x column] data.

Opcode	Data
40.0	Part transform information (All)

Word	Data
1	Number of Parts to be updated (1..n)
2-17	First 4x4 [row x column] data.
18-33	Second 4x4 [row x column] data.
...	...

Opcode	Data
50.0	Part transform information (Single)

Word	Data
1	Which Part to update (1..n)
2-17	4x4 [row x column] data.

Opcode	Data
60.0	Eye point information

Word	Data
1	Length of User View name.
2-n	User View name (array of packed chars).

70.0 Device display mode information

Word	Data	
1	Display mode	index
	Invisible	0.0
	Wireframe	1.0
	Hidden Line	2.0
	Flat Shading	3.0
	Fancy Shading	4.0
	Smooth Shading	5.0
	Transparent	6.0
	All Polygons	7.0

80.0 Rotate world relative immediate

Word	Data
1	Axis
	X 0.0
	Y 1.0
	Z 2.0
2	Relative amount data.

90.0 Rotate world absolute immediate

Word	Data
1	Absolute X position (yaw) data.
2	Absolute Y position (pitch) data.
3	Absolute Z rotation (roll) data.

100.0 Translate world relative immediate

Word	Data
1	Axis
	X 0.0
	Y 1.0
	Z 2.0
2	Relative amount data.

110.0 Translate world absolute immediate

Word	Data
1	Absolute X position data.
2	Absolute Y position data.
3	Absolute Z position data.

120.0 Scale world relative immediate

Word	Data
1	Scale amount data.

130.0 CLI command

Word	Data
1	Length of CLI command
2-n	CLI text (array of packed chars)

131.0 Multiple CLI commands

Word	Data
1	Length of CLI command set including ~ (tildes)
2-n	CLI commands separated by ~ (array of packed characters)

140.0 Graphic Update 1 Buffer

Word	Data
no data required	

150.0 Graphic Update 2 Buffers

Word	Data
no data required	

APPENDIX C – AN INCREMENTAL ENCODER SIMULATOR

PCA

INCREMENTAL ENCODER SIMULATOR

Part No. **ENC-SIM01**

Release October 2006

This instrument is designed to simulate the function of most types of incremental encoders; it allows the engineer to fully test a control system that uses a rotary or linear shaft encoder, without the need for any mechanical movement. In addition the user can control the exact number of square wave signals applied to the output, this allows the engineer to control the input pulses generated down to just one pulse, so the user can monitor in slow motion the operation of a control system.

Access to the instrument is through a standard nine pin D type plug, the user simply connects the simulator to the terminals or plug of their control system. The power for the instrument is taken from the control system, it will operate on any voltage from 5 to 30 volts 100mA DC

PLUG WIRING

1 = O Volts	3 = A Output	6 = /A Output
2 = + 5 to 30 Volts	4 = B Output	7 = /B Output
	5 = O Output	8 = /O Output

EXPLANATION OF CONTOL PANEL

LCD Display Panel 2 X 16 Character 5.5mm x 3mm

RPM: Displays the speed of the output encoder, this is determined by the time taken for each "shaft" rotation. The maximum value for the RPM display is 15,000, if the simulation parameters drive the display faster, the value will display as arrow heads.

MODE: This states the current mode of operation for the programming switches in the panel below

PPR: Displays the number of pulses for one rotation of the simulated shaft encoder that has been programmed into the instrument or it displays the number of increment pulses between each zero pulse.

Count Progress: This is used when in MAN mode so that the user can see the number of signals that have been transmitted to the output.

Frequency: Displays the frequency of the output signals, the minimum value is 10 Hertz

OPERATOR PROGRAMMING BUTTONS

Mode Selection: The red LED at the top of each column shows the function which will be applied to the four programming buttons on the left side of the operator control panel. The two Select buttons below change the selection from one mode to the next.

PPR Mode: To setup the pulses of the encoder to be simulated, the results of the following setting process are shown in the PPR display.

Start: Resets the PPR display to 2, and allows the new selection process to begin.

Select: Increments the right digit under the PPR display by one, each time the button is pressed.

Move: Moves the digit just entered one space to the left, then the Select input is used again to increment the next digit in the number as described above.

Save & Run: Once the desired value is entered, press this button and the "Count Progress" number is resent to zero and the system is ready to run in either "man" or continuously in the "Rate" modes of operation

NOTE: The output signal pulse width is determined by the Hertz selected under the RATE mode setup.

NOTE 2: This button will also rest the counter at any time when the RPM selection *is operational.*

MAN Mode: This mode provides the facility to *manually* control the number of output signals sent. The progress of this entry is displayed on the "Count Progress" display.

"1": Indexes the output by 1 pulse with each press of the button.

"10" Indexes the output by 10 pulses with each press of the button.

"100" Indexes the output by 100 pulses with each press of the button.

"1000" Indexes the output by 1000 pulses with each press of the button.

NOTE: If there is no output selected (a green LED illuminated) and a pulse train button pressed, the last value selected will output immediately the output is turned on.

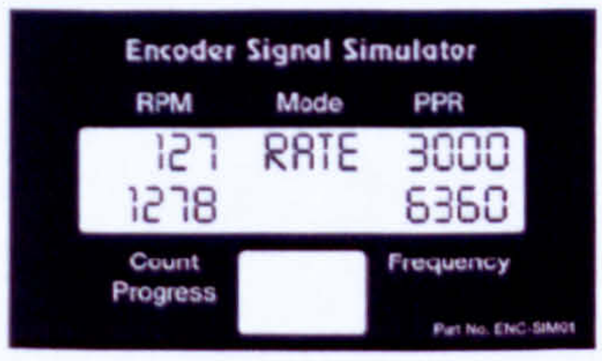
RATE: The results of this range setting are displayed on the "Frequency" display, for speeds between the ranges rotate the "Hertz Fine Adjust" knob.

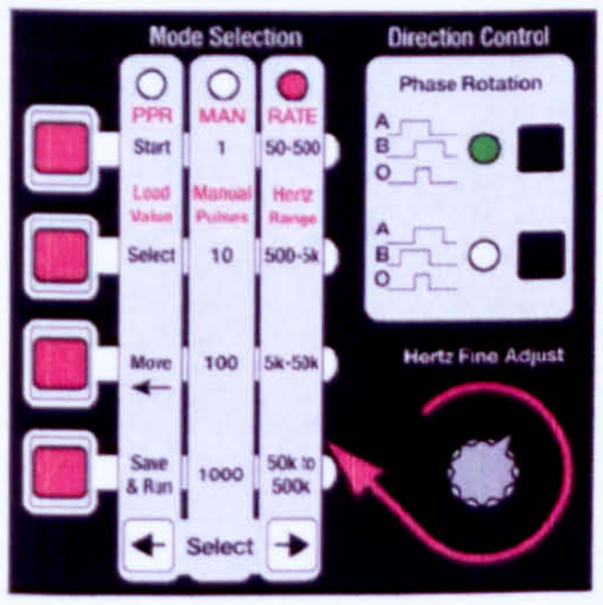
50 - 500: Fifty to five hundred hertz.
500 – 5k: Five hundred to five thousand hertz.
5k – 50k: Five thousand to fifty thousand hertz.
50k – 500k: Fifty to five hundred thousand hertz.

Direction Control: Selects the output wave relationship, this function in an encoder determines the direction of rotation as seen by the control system. The top button produces an output phase with A rising before B, the lower button selection will make B rise before A.

NOTE: No output signals are transmitted until a phase direction is selected, a green LED must be illuminated.

Housing Size: 95mm wide, 150mm long, 30mm high for the button section, and 40mm high at the display end.



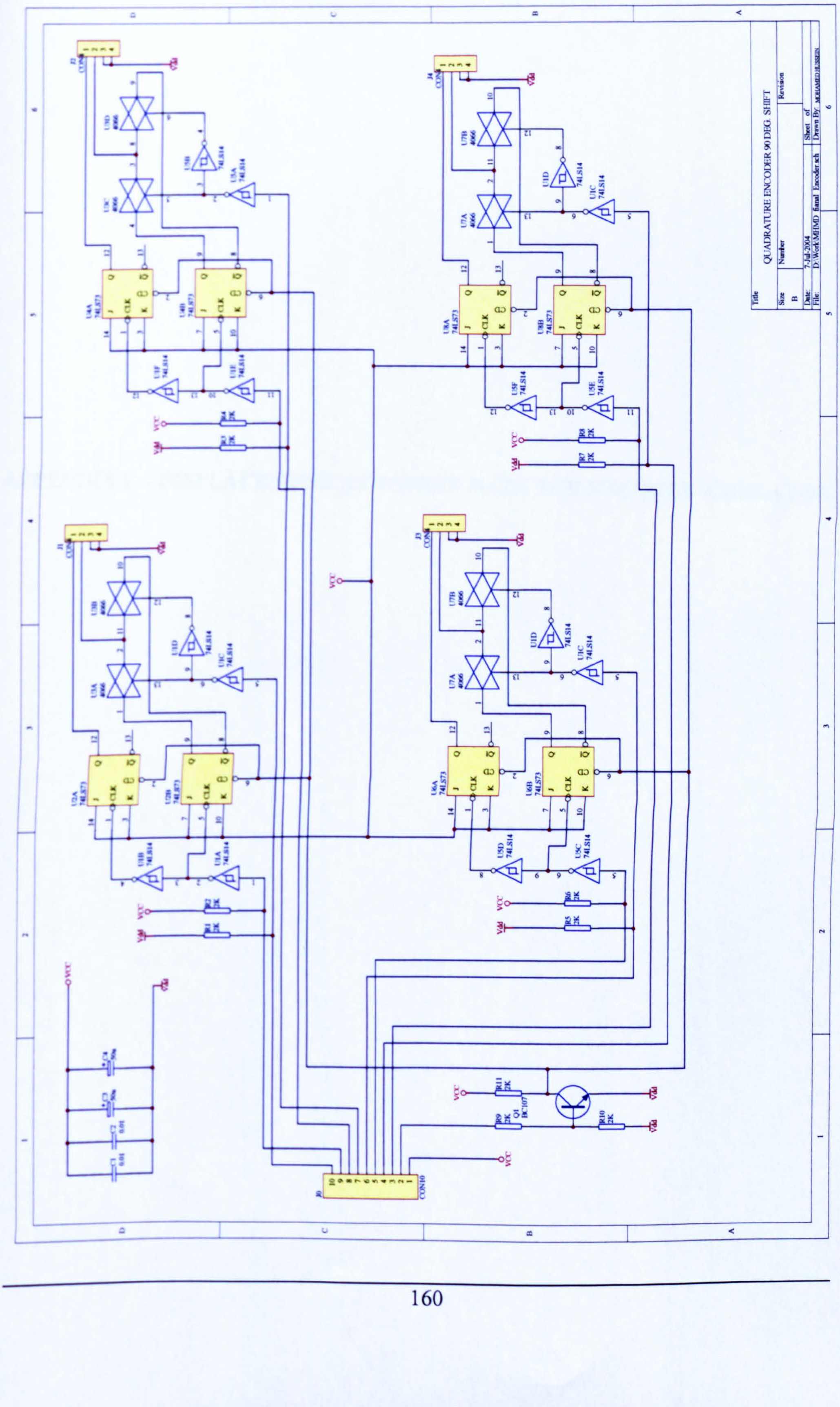


PLANT CONTROL AND AUTOMATION PTY LTD

02 9482 3733 1 800 226 345 Australia ABN 51-000-671-830 PO Box 121 Hornsby NSW 1630

02 9476 6822 Country Code 61 http://www.pca-aus.com.au *Unit 3 No 95 Hunter Street Hornsby NSW 2077*

APPENDIX D – SCHEMATIC DIAGRAM FOR QUADRATURE ENCODER EMULATOR


QUADRATURE ENCODER 90 DEG. SHIFT
Title
Size
B
Number
Revision
Date: 7-Jul-2004
Sheet of
File: D:\Work\MHMD final Encoder.sch
Drawn By: MOHAMED HUSSEIN
J4
J3
J2
J1
U7B 4066
U7A 4066
U3D 4066
U3C 4066
U3B 4066
U3A 4066
U1D 74LS14
U1C 74LS14
U1F 74LS14
U1E 74LS14
U1B 74LS14
U1A 74LS14
U8A 74LS73
U8B 74LS73
U6A 74LS73
U6B 74LS73
U5F 74LS14
U5E 74LS14
U5D 74LS14
U5C 74LS14
U4A 74LS73
U4B 74LS73
U2A 74LS73
U2B 74LS73
R8 2K
R7 2K
R6 2K
R5 2K
R4 2K
R3 2K
R2 2K
R1 2K
R11 2K
R9 2K
R10 2K
Q1 BC107
C4 50u
C3 50u
C2 0.01
C1 0.01
VCC
Vdd
J0
CON10

APPENDIX E – DISPLACEMENT AND SPEED DATA FOR ENCODER EMULATOR

		Axis 1		Axis 2		Axis 3	
	Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
	0.000	0.0	0.0	0.0	0.0	0.0	0.0
	0.014	0.0	0.0	0.0	0.0	0.0	0.0
	0.028	0.0	0.0	0.0	0.0	0.0	0.0
	0.043	0.0	0.0	0.0	0.0	0.0	0.0
	0.057	0.0	0.0	0.0	0.0	0.0	0.0
	0.071	0.0	0.0	0.0	0.0	0.0	0.0
	0.085	0.0	0.0	0.0	0.0	0.0	0.0
	0.099	0.0	0.0	0.0	0.0	0.0	0.0
	0.114	0.0	0.0	0.0	0.0	0.0	0.0
	0.128	0.0	0.0	0.0	0.0	0.0	0.0
	0.142	0.0	0.0	0.0	0.0	0.0	0.0
	0.156	0.0	0.0	0.0	0.0	0.0	0.0
	0.170	0.0	0.0	0.0	0.0	0.0	0.0
	0.185	0.0	0.0	0.0	0.0	0.0	0.0
	0.199	0.0	0.0	0.0	0.0	0.0	0.0
	0.213	0.0	0.0	0.0	0.0	0.0	0.0
	0.227	0.0	0.0	0.0	0.0	0.0	0.0
	0.241	0.0	0.0	0.0	0.0	0.0	0.0
	0.256	0.0	0.0	0.0	0.0	0.0	0.0
	0.270	0.0	0.0	0.0	0.0	0.0	0.0
	0.284	0.0	0.0	0.0	0.0	0.0	0.0
	0.298	0.0	0.0	0.0	0.0	0.0	0.0
	0.312	0.0	0.0	0.0	0.0	0.0	0.0
	0.327	0.0	0.0	0.0	0.0	0.0	0.0
	0.341	0.0	0.0	0.0	0.0	0.0	0.0
	0.355	0.0	0.0	0.0	0.0	0.0	0.0
	0.369	0.0	0.0	0.0	0.0	0.0	0.0
	0.383	0.0	0.0	0.0	0.0	0.0	0.0
	0.398	0.0	0.0	0.0	0.0	0.0	0.0
	0.412	0.0	0.0	0.0	0.0	0.0	0.0
	0.426	0.0	0.0	0.0	0.0	0.0	0.0
	0.440	0.0	0.0	0.0	0.0	0.0	0.0
	0.454	0.0	0.0	0.0	0.0	0.0	0.0
	0.469	0.0	0.0	0.0	0.0	0.0	0.0
	0.483	0.0	0.0	0.0	0.0	0.0	0.0
	0.497	0.0	0.0	0.0	0.0	0.0	0.0
	0.511	0.0	0.0	0.0	0.0	0.0	0.0
	0.525	0.0	0.0	0.0	0.0	0.0	0.0
	0.540	0.0	0.0	0.0	0.0	0.0	0.0
Axis 1 (Start)	**0.554**	**0.0**	**0.0**	0.0	0.0	0.0	0.0
	0.568	0.4	0.0	0.0	0.0	0.0	0.0
	0.582	0.6	19.5	0.0	0.0	0.0	0.0
	0.596	0.7	19.5	0.0	0.0	0.0	0.0
	0.611	0.9	19.7	0.0	0.0	0.0	0.0
	0.625	1.0	19.7	0.0	0.0	0.0	0.0
	0.639	1.4	19.7	0.0	0.0	0.0	0.0
	0.653	1.6	19.7	0.0	0.0	0.0	0.0
	0.667	2.0	19.7	0.0	0.0	0.0	0.0
	0.682	2.1	20.0	0.0	0.0	0.0	0.0
	0.696	2.2	19.7	0.0	0.0	0.0	0.0
	0.710	2.6	20.0	0.0	0.0	0.0	0.0
	0.724	2.8	20.0	0.0	0.0	0.0	0.0

		Axis 1		Axis 2		Axis 3	
	Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
	0.738	3.2	19.7	0.0	0.0	0.0	0.0
	0.753	3.4	20.0	0.0	0.0	0.0	0.0
	0.767	3.8	20.0	0.0	0.0	0.0	0.0
	0.781	4.0	20.0	0.0	0.0	0.0	0.0
	0.795	4.4	19.7	0.0	0.0	0.0	0.0
	0.809	4.6	20.0	0.0	0.0	0.0	0.0
	0.824	5.0	20.0	0.0	0.0	0.0	0.0
	0.838	5.4	20.0	0.0	0.0	0.0	0.0
	0.852	5.6	20.0	0.0	0.0	0.0	0.0
	0.866	5.9	19.7	0.0	0.0	0.0	0.0
	0.880	6.2	20.0	0.0	0.0	0.0	0.0
	0.895	6.4	19.7	0.0	0.0	0.0	0.0
	0.909	6.8	20.0	0.0	0.0	0.0	0.0
	0.923	7.0	20.0	0.0	0.0	0.0	0.0
	0.937	7.4	20.0	0.0	0.0	0.0	0.0
	0.951	7.5	20.0	0.0	0.0	0.0	0.0
	0.966	7.6	20.0	0.0	0.0	0.0	0.0
	0.980	8.0	20.0	0.0	0.0	0.0	0.0
	0.994	8.2	20.0	0.0	0.0	0.0	0.0
	1.008	8.6	20.0	0.0	16.7	0.0	0.0
Axis 2 (Start)	**1.022**	8.8	20.0	**0.0**	**19.5**	0.0	0.0
	1.037	9.2	20.0	0.7	19.5	0.0	0.0
	1.051	9.4	20.0	1.2	19.5	0.0	0.0
	1.065	9.8	19.7	1.3	19.5	0.0	0.0
	1.079	10.0	20.0	1.4	20.1	0.0	0.0
	1.093	10.4	20.0	1.8	20.1	0.0	0.0
	1.108	10.5	20.0	2.0	20.1	0.0	0.0
	1.122	10.6	20.0	2.4	20.1	0.0	0.0
	1.136	11.0	19.7	2.6	20.1	0.0	0.0
	1.150	11.4	20.0	2.9	19.5	0.0	0.0
	1.164	11.6	20.0	3.0	19.5	0.0	0.0
	1.179	11.8	20.0	3.4	20.1	0.0	0.0
	1.193	12.2	20.0	3.6	20.1	0.0	0.0
	1.207	12.6	20.0	3.8	20.1	0.0	0.0
	1.221	12.8	20.0	4.1	19.5	0.0	0.0
	1.235	13.0	20.0	4.3	19.5	0.0	0.0
	1.250	13.4	20.0	4.4	19.5	0.0	0.0
	1.264	13.5	20.0	4.8	19.5	0.0	0.0
	1.278	13.6	20.0	5.1	20.1	0.0	0.0
	1.292	14.1	19.7	5.4	20.1	0.0	0.0
	1.306	14.2	20.0	5.6	19.5	0.0	0.0
	1.321	14.6	20.0	5.9	19.5	0.0	0.0
	1.335	14.8	20.0	6.1	20.1	0.0	0.0
	1.349	15.3	20.0	6.2	20.1	0.0	0.0
	1.363	15.4	20.0	6.5	20.1	0.0	0.0
	1.377	15.8	20.0	6.7	19.5	0.0	0.0
	1.392	15.9	20.0	6.9	20.1	0.0	0.0
	1.406	16.1	20.0	7.2	19.5	0.0	0.0
	1.420	16.5	20.0	7.4	19.5	0.0	0.0
	1.434	16.8	20.0	7.8	20.1	0.0	0.0
	1.448	17.1	20.0	8.0	20.1	0.0	0.0
	1.463	17.3	20.0	8.3	20.1	0.0	0.0
	1.477	17.7	20.0	8.5	19.5	0.0	0.0

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
1.491	17.9	20.0	8.6	20.1	0.0	0.0
1.505	18.3	20.0	9.0	20.1	0.0	0.0
1.519	18.5	20.0	9.2	20.1	0.0	0.0
1.534	18.9	20.0	9.5	19.5	0.0	0.0
1.548	19.3	20.0	9.7	20.1	0.0	0.0
1.562	19.5	19.7	9.8	20.1	0.0	0.0
1.576	19.7	20.0	10.2	20.1	0.0	0.0
1.590	20.1	20.0	10.4	20.1	0.0	0.0
1.605	20.3	20.0	10.8	19.5	0.0	0.0
1.619	20.7	20.0	11.0	20.1	0.0	0.0
1.633	20.9	20.0	11.4	20.1	0.0	0.0
1.647	21.3	20.0	11.6	20.1	0.0	0.0
1.661	21.5	20.0	12.0	20.1	0.0	0.0
1.676	21.9	20.0	12.2	19.5	0.0	0.0
1.690	22.1	20.0	12.6	20.1	0.0	0.0
1.704	22.5	20.0	12.8	20.1	0.0	0.0
1.718	22.7	20.0	13.2	20.1	0.0	0.0
1.732	23.0	20.0	13.4	20.1	0.0	0.0
1.747	23.1	20.0	13.7	20.1	0.0	0.0
1.761	23.4	20.0	13.9	20.1	0.0	0.0
1.775	23.7	20.0	14.0	20.1	0.0	0.0
1.789	23.9	20.0	14.4	20.1	0.0	0.0
1.803	24.2	20.0	14.8	20.1	0.0	0.0
1.818	24.3	19.7	15.0	20.1	0.0	0.0
1.832	24.5	20.0	15.2	19.5	0.0	0.0
1.846	24.9	20.0	15.6	20.1	0.0	0.0
1.860	25.2	19.7	15.8	20.1	0.0	0.0
1.874	25.5	20.0	16.2	20.1	0.0	0.0
1.889	25.7	20.0	16.6	19.5	0.0	0.0
1.903	26.1	20.0	16.8	19.5	0.0	0.0
1.917	26.3	20.0	17.0	20.1	0.0	0.0
1.931	26.7	20.0	17.4	20.1	0.0	0.0
1.945	28.6	20.0	17.6	20.1	0.0	0.0
1.960	28.9	20.0	17.9	20.1	0.0	0.0
1.974	29.1	20.0	18.1	20.1	0.0	0.0
1.988	29.4	19.7	18.2	19.5	0.0	0.0
2.002	29.8	20.0	18.6	20.1	0.0	0.0
2.016	30.1	20.0	18.8	20.1	0.0	0.0
2.031	30.4	20.0	19.2	20.1	0.0	0.0
2.045	30.7	20.0	19.4	20.1	0.0	0.0
2.059	30.9	20.0	19.8	20.1	0.0	0.0
2.073	31.3	20.0	20.0	20.1	0.0	0.0
2.087	31.6	20.0	20.4	20.1	0.0	0.0
2.102	32.0	20.0	20.6	20.1	0.0	0.0
2.116	32.1	20.0	21.0	19.5	0.0	0.0
2.130	32.5	20.0	21.2	19.5	0.0	0.0
2.144	32.8	20.0	21.6	20.1	0.0	0.0
2.158	33.1	20.0	21.8	20.1	0.0	0.0
2.173	34.5	20.0	22.1	20.1	0.0	0.0
2.187	34.8	20.0	22.3	20.1	0.0	0.0
2.201	35.1	20.0	22.4	20.1	0.0	0.0
2.215	35.3	20.0	22.8	20.1	0.0	0.0
2.229	35.7	20.0	23.0	20.1	0.0	0.0

		Axis 1		Axis 2		Axis 3	
	Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
	2.244	35.9	20.0	23.4	20.1	0.0	0.0
	2.258	36.2	20.0	23.8	19.5	0.0	0.0
	2.272	36.5	20.0	24.0	19.5	0.0	0.0
	2.286	36.9	20.0	24.2	20.1	0.0	0.0
	2.300	37.1	20.0	24.6	20.1	0.0	0.0
	2.315	37.5	20.0	24.8	20.1	0.0	0.0
	2.329	37.7	20.0	25.2	20.1	0.0	0.0
	2.343	38.1	19.7	25.6	20.1	0.0	0.0
	2.357	38.5	20.0	25.8	20.1	0.0	0.0
	2.371	38.7	20.0	26.0	20.1	0.0	0.0
	2.386	38.9	20.0	26.4	20.1	0.0	0.0
	2.400	39.3	20.0	26.5	20.1	0.0	0.0
	2.414	39.5	20.0	26.6	20.1	0.0	0.0
	2.428	39.9	20.0	27.0	19.5	0.0	0.0
	2.442	40.1	20.0	27.2	20.1	0.0	0.0
	2.457	40.5	20.0	27.6	20.1	0.0	0.0
	2.471	40.7	20.0	27.8	20.1	0.0	0.0
	2.485	41.1	20.0	28.2	20.1	0.0	0.0
	2.499	41.3	20.0	28.4	19.5	0.0	0.0
	2.513	41.7	20.0	28.9	19.5	0.0	0.0
	2.528	41.9	20.0	29.1	20.1	0.0	0.0
	2.542	42.2	20.0	29.4	20.1	0.0	0.0
	2.556	42.3	20.0	29.8	20.1	0.0	0.0
	2.570	42.5	20.0	30.0	19.5	0.0	0.0
	2.584	42.9	20.0	30.3	20.1	0.0	0.0
	2.599	43.1	20.0	30.7	20.1	0.0	0.0
	2.613	43.4	20.0	30.8	20.1	0.0	0.0
Axis 3 (Start)	**2.627**	43.5	20.0	31.3	20.1	**0.0**	**0.0**
	2.641	43.9	20.3	31.6	20.1	0.4	9.2
	2.655	44.1	20.0	31.9	20.1	0.5	9.9
	2.670	44.3	20.0	32.0	20.1	0.7	9.9
	2.684	44.6	20.0	32.4	19.5	0.8	9.9
	2.698	44.7	20.0	32.5	20.1	1.0	9.9
	2.712	44.9	20.0	32.7	20.1	1.1	9.9
	2.726	45.3	20.0	33.1	19.5	1.3	10.0
	2.741	45.5	20.0	33.3	20.1	1.4	9.9
	2.755	45.9	20.0	33.7	20.1	1.6	9.9
	2.769	46.3	20.0	33.9	20.1	1.7	9.9
	2.783	46.5	20.0	34.3	20.1	1.8	10.0
	2.797	46.7	20.0	34.5	19.5	1.9	9.9
	2.812	47.1	20.0	34.8	20.1	2.0	10.0
	2.826	47.3	20.0	34.9	20.1	2.2	9.9
	2.840	47.7	19.7	35.1	20.1	2.3	9.9
	2.854	47.9	20.0	35.5	20.1	2.5	9.9
	2.868	48.3	19.7	35.7	20.1	2.6	9.9
	2.883	48.5	20.0	36.0	20.1	2.8	9.9
	2.897	48.8	19.7	36.1	19.5	2.9	10.0
	2.911	48.9	20.0	36.3	20.1	3.1	10.0
	2.925	49.3	19.7	36.7	20.1	3.2	9.9
	2.939	49.5	20.0	36.9	20.1	3.4	9.9
	2.954	49.7	20.0	37.3	19.5	3.5	9.9
	2.968	50.1	20.0	37.5	20.1	3.7	10.0
	2.982	50.3	20.0	37.9	20.1	3.8	10.0

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
2.996	50.7	20.0	38.2	20.1	4.0	9.9
3.010	51.1	20.0	38.5	20.1	4.1	9.9
3.025	51.3	20.0	38.7	19.5	4.3	9.9
3.039	51.5	20.0	39.1	20.1	4.4	10.0
3.053	51.8	20.0	39.3	20.1	4.5	10.0
3.067	51.9	20.0	39.7	20.1	4.6	9.9
3.081	52.1	20.0	39.9	20.1	4.8	10.0
3.096	52.5	20.0	40.3	19.5	4.9	10.0
3.110	52.7	20.0	40.5	20.1	5.0	9.9
3.124	53.1	19.7	40.9	20.1	5.2	9.9
3.138	53.3	20.0	41.1	20.1	5.3	9.9
3.152	53.7	20.0	41.5	20.1	5.5	9.9
3.167	54.1	20.0	41.7	19.5	5.7	10.0
3.181	54.3	20.0	42.1	20.1	5.8	10.0
3.195	54.5	20.0	42.3	20.1	5.9	9.9
3.209	54.9	20.0	42.7	20.1	6.1	9.9
3.223	55.1	19.7	42.9	20.1	6.2	10.0
3.238	55.5	20.0	43.3	20.1	6.4	9.9
3.252	55.7	20.0	43.5	20.1	6.5	9.9
3.266	56.1	20.0	43.9	20.1	6.6	9.9
3.280	56.3	20.0	44.1	20.1	6.7	9.9
3.294	56.7	20.0	44.4	20.1	6.8	9.9
3.309	56.9	20.0	44.5	20.1	7.0	9.9
3.323	57.2	20.0	44.8	20.1	7.2	10.0
3.337	57.3	20.0	45.1	20.1	7.3	9.9
3.351	57.5	20.0	45.3	20.1	7.4	9.9
3.365	57.9	20.0	45.7	20.1	7.6	10.0
3.380	58.1	20.0	45.9	19.5	7.7	9.9
3.394	58.5	20.0	46.3	20.1	7.9	9.9
3.408	58.7	20.0	46.5	20.1	8.1	10.0
3.422	59.1	20.0	46.9	20.1	8.2	9.9
3.436	59.3	20.0	47.1	20.1	8.3	9.9
3.451	59.7	20.0	47.5	19.5	8.5	10.0
3.465	59.9	20.0	47.8	20.1	8.6	10.0
3.479	60.3	20.0	48.1	20.1	8.7	10.0
3.493	60.4	20.0	48.3	20.1	8.8	9.9
3.507	60.5	20.0	48.6	20.1	8.9	9.9
3.522	60.9	20.0	48.7	20.1	9.1	9.9
3.536	61.1	20.0	48.9	19.5	9.2	9.9
3.550	61.5	20.0	49.3	20.1	9.4	10.0
3.564	61.7	20.0	49.5	20.1	9.5	9.9
3.578	62.1	20.0	49.9	20.1	9.7	9.9
3.593	62.3	20.0	50.1	20.1	9.8	9.9
3.607	62.7	19.7	50.5	19.5	10.0	9.9
3.621	62.9	20.0	50.9	20.1	10.1	9.9
3.635	63.3	20.0	51.1	20.1	10.3	10.0
3.649	63.5	20.0	51.3	20.1	10.4	9.9
3.664	63.9	20.0	51.6	20.1	10.6	9.9
3.678	64.3	20.0	51.7	20.1	10.7	9.9
3.692	64.5	20.0	51.9	19.5	10.8	9.9
3.706	64.7	20.0	52.3	20.1	10.9	9.9
3.720	65.1	20.0	52.5	20.1	11.1	10.0
3.735	65.4	20.0	52.9	20.1	11.2	9.9

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
3.749	65.7	20.0	53.1	20.1	11.3	10.0
3.763	65.9	20.0	53.5	20.1	11.5	9.9
3.777	66.3	20.0	53.7	20.1	11.6	9.9
3.791	66.5	20.0	54.1	20.1	11.8	9.9
3.806	66.9	20.0	54.3	20.1	11.9	9.9
3.820	67.3	20.0	54.7	19.5	12.1	9.9
3.834	67.5	20.0	54.9	20.1	12.2	10.0
3.848	67.7	20.0	55.3	20.1	12.4	9.9
3.862	68.1	20.0	55.5	20.1	12.6	10.0
3.877	68.3	20.0	55.9	20.1	12.7	10.0
3.891	68.7	20.0	56.1	20.1	12.8	9.9
3.905	68.7	20.0	56.5	20.1	13.0	9.9
3.919	69.1	20.0	56.8	20.1	13.0	9.9
3.933	69.3	20.0	57.1	20.1	13.1	10.0
3.948	69.5	20.0	57.3	20.1	13.3	9.9
3.962	69.9	20.0	57.7	19.5	13.5	9.9
3.976	70.0	20.0	57.9	20.1	13.6	10.0
3.990	70.1	20.0	58.3	20.1	13.7	9.9
4.004	70.5	20.0	58.5	20.1	13.9	9.9
4.019	70.7	20.0	58.9	20.1	14.0	9.9
4.033	71.1	20.0	59.3	19.5	14.2	9.9
4.047	71.3	20.0	59.5	20.1	14.3	9.9
4.061	71.7	20.0	59.7	20.1	14.5	9.9
4.075	71.7	20.0	60.0	19.5	14.6	10.0
4.090	71.9	20.0	60.1	20.1	14.8	9.9
4.104	72.3	20.0	60.3	20.1	15.0	9.9
4.118	72.5	20.0	60.7	20.1	15.1	10.0
4.132	72.9	20.0	60.9	19.5	15.2	9.9
4.146	73.1	20.0	61.2	20.1	15.4	9.9
4.161	73.5	20.0	61.3	20.1	15.5	9.9
4.175	73.9	20.0	61.5	20.1	15.7	10.0
4.189	74.1	20.0	61.8	20.1	15.7	9.9
4.203	74.3	20.0	61.9	20.1	15.9	10.0
4.217	74.7	20.0	62.1	19.5	16.0	10.0
4.232	74.9	20.0	62.5	20.1	16.2	9.9
4.246	75.3	20.0	62.7	20.1	16.3	9.9
4.260	75.7	20.0	63.0	19.5	16.5	9.9
4.274	75.9	20.0	63.1	20.1	16.6	9.9
4.288	76.1	19.7	63.3	20.1	16.8	9.9
4.303	76.5	20.0	63.7	19.5	16.9	9.9
4.317	76.7	20.0	64.1	20.1	17.0	10.0
4.331	77.1	20.0	64.3	20.1	17.2	9.9
4.345	77.3	20.0	64.5	20.1	17.2	10.0
4.359	77.7	20.0	64.9	20.1	17.4	10.0
4.374	77.9	20.0	65.1	19.5	17.5	10.0
4.388	78.3	20.0	65.4	20.1	17.6	9.9
4.402	78.4	20.0	65.5	20.1	17.8	9.9
4.416	78.5	20.0	65.9	20.1	17.9	9.9
4.430	78.9	20.0	66.1	20.1	18.1	9.9
4.445	79.1	20.0	66.3	20.1	18.2	9.9
4.459	79.5	20.0	66.7	19.5	18.4	10.0
4.473	79.7	20.0	66.9	20.1	18.5	10.0
4.487	80.1	20.0	67.3	20.1	18.7	9.9

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
4.501	80.3	20.0	67.5	20.1	18.9	9.9
4.516	80.7	20.0	67.9	19.5	19.0	10.0
4.530	80.9	20.0	68.1	19.5	19.1	9.9
4.544	81.3	19.7	68.5	20.1	19.2	9.9
4.558	81.5	20.0	68.7	20.1	19.3	9.9
4.572	81.9	20.0	69.1	20.1	19.4	10.0
4.587	82.1	20.0	69.3	20.1	19.6	9.9
4.601	82.5	19.7	69.6	20.1	19.7	9.9
4.615	82.6	20.0	69.7	19.5	19.9	9.9
4.629	82.7	20.0	69.9	20.1	20.0	9.9
4.643	83.1	20.0	70.3	20.1	20.2	9.9
4.658	83.5	20.0	70.7	20.1	20.4	9.9
4.672	83.7	20.0	70.9	19.5	20.5	9.9
4.686	83.9	20.0	71.1	20.1	20.6	9.9
4.700	84.3	20.0	71.5	20.1	20.8	10.0
4.714	84.5	20.0	71.7	20.1	20.9	9.9
4.729	84.9	20.0	72.1	20.1	21.1	9.9
4.743	85.3	20.3	72.5	20.1	21.2	9.9
4.757	85.5	20.0	72.7	20.1	21.3	9.9
4.771	85.7	20.0	72.9	20.1	21.4	9.9
4.785	86.1	19.7	73.3	20.1	21.5	9.9
4.800	86.3	20.0	73.5	20.1	21.7	10.0
4.814	86.7	20.0	73.8	20.1	21.8	10.0
4.828	86.8	20.0	73.9	20.1	22.0	9.9
4.842	86.9	20.0	74.1	20.1	22.1	9.9
4.856	87.3	20.0	74.5	20.1	22.3	10.0
4.871	87.5	20.0	74.7	20.1	22.4	9.9
4.885	87.9	20.0	75.1	20.1	22.6	9.9
4.899	88.0	20.0	75.3	19.5	22.7	10.0
4.913	88.1	20.0	75.7	20.1	22.9	9.9
4.927	88.5	20.0	75.9	20.1	23.0	9.9
4.942	88.7	20.0	76.3	20.1	23.2	10.0
4.956	89.1	20.0	76.5	20.1	23.4	10.0
4.970	89.3	20.0	76.9	20.1	23.5	9.9
4.984	89.7	20.0	77.3	20.1	23.6	9.9
4.998	90.1	20.0	77.5	20.1	23.8	9.9
5.013	90.3	20.0	77.7	20.1	23.9	9.9
5.027	90.5	20.0	78.0	20.1	24.1	9.9
5.041	90.9	20.0	78.2	20.1	24.2	10.0
5.055	91.0	19.7	78.3	20.1	24.4	9.9
5.069	91.1	20.0	78.7	20.1	24.6	9.9
5.084	91.5	20.0	78.9	20.1	24.7	10.0
5.098	91.9	20.0	79.3	20.1	24.8	9.9
5.112	92.1	20.0	79.8	20.1	25.0	9.9
5.126	92.3	20.0	79.9	20.1	25.1	10.0
5.140	92.7	20.0	80.3	20.1	25.3	9.9
5.155	92.9	20.0	80.5	20.1	25.4	9.9
5.169	93.3	20.0	80.7	20.1	25.6	10.0
5.183	93.5	20.0	81.1	19.5	25.7	9.9
5.197	93.9	20.0	81.3	20.1	25.9	9.9
5.211	94.1	20.0	81.7	20.1	26.0	9.9
5.226	94.5	20.0	82.0	20.1	26.2	10.0
5.240	94.7	20.0	82.3	20.1	26.2	9.9

		Axis 1		Axis 2		Axis 3	
	Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
	5.254	95.1	20.0	82.5	20.1	26.3	9.9
	5.268	95.3	20.0	82.9	20.1	26.5	9.9
	5.282	95.7	20.0	83.1	20.1	26.6	10.0
	5.297	95.9	20.0	83.5	20.1	26.8	9.9
	5.311	96.3	20.0	83.7	20.1	26.9	9.9
	5.325	96.4	20.0	84.1	19.5	27.1	10.0
	5.339	96.7	20.3	84.3	20.1	27.3	9.9
	5.353	96.9	20.0	84.7	20.1	27.4	9.9
	5.368	97.1	20.0	85.1	20.1	27.5	9.9
	5.382	97.5	20.0	85.3	19.5	27.7	9.9
	5.396	97.7	20.0	85.5	19.5	27.7	9.9
	5.410	98.1	20.0	85.9	20.1	27.8	9.9
	5.424	98.2	20.0	86.1	20.1	28.0	9.9
	5.439	98.3	20.0	86.4	19.5	28.1	9.9
	5.453	98.7	20.0	86.5	20.1	28.3	9.9
	5.467	98.9	20.0	86.9	20.1	28.4	9.9
	5.481	99.3	20.0	87.1	19.5	28.6	10.0
	5.495	99.7	20.0	87.3	20.1	28.7	9.9
	5.510	97.9	10.6	87.7	20.1	28.9	9.9
	5.524	99.9	1.1	87.8	20.1	29.0	10.0
	5.538	99.9	0.6	87.9	20.1	29.2	9.9
	5.552	99.9	0.3	88.3	20.1	29.3	9.9
Axis 1 (End)	**5.566**	**100.0**	**0.3**	88.7	20.1	29.5	10.0
	5.581	100.0	0.3	88.9	19.5	29.6	9.9
	5.595	100.0	0.3	89.1	20.1	29.8	9.9
	5.609	100.0	0.3	89.5	20.1	29.9	9.9
	5.623	100.0	0.3	89.6	20.1	30.1	10.0
	5.637	100.0	0.3	89.7	20.1	30.2	9.9
	5.652	100.0	0.0	90.1	20.1	30.4	9.9
	5.666	100.0	0.0	90.3	19.5	30.4	9.9
	5.680	100.0	0.3	90.7	20.1	30.6	9.9
	5.694	100.0	0.3	90.9	20.1	30.7	9.9
	5.708	100.0	0.3	91.3	20.1	30.8	9.9
	5.723	100.0	0.0	91.5	20.1	31.0	9.9
	5.737	100.0	0.3	91.9	20.1	31.1	9.9
	5.751	100.0	0.0	92.1	20.1	31.3	9.9
	5.765	100.0	0.0	92.5	19.5	31.5	9.9
	5.779	100.0	0.0	92.6	20.1	31.6	9.9
	5.794	100.0	0.3	92.7	20.1	31.7	9.9
	5.808	100.0	0.3	93.1	19.5	31.9	9.9
	5.822	100.0	0.0	93.5	20.1	31.9	10.0
	5.836	100.0	0.0	93.7	20.1	32.0	10.0
	5.850	100.0	0.0	93.9	20.1	32.2	9.9
	5.865	100.0	0.0	94.3	20.1	32.3	9.9
	5.879	100.0	0.0	94.5	19.5	32.5	9.9
	5.893	100.0	0.0	94.9	20.1	32.6	9.9
	5.907	100.0	0.0	95.0	20.1	32.8	10.0
	5.921	100.0	0.0	95.3	20.1	33.0	9.9
	5.936	100.0	0.0	95.5	20.1	33.1	9.9
	5.950	100.0	0.0	95.7	20.1	33.2	9.9
	5.964	100.0	0.0	96.1	20.1	33.4	9.9
	5.978	100.0	0.0	96.3	20.1	33.5	9.9
	5.992	100.0	0.0	96.7	20.1	33.7	9.9

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
6.007	100.0	0.0	96.9	20.1	33.8	10.0
6.021	100.0	0.0	97.3	19.5	34.0	9.9
6.035	100.0	0.0	97.5	20.1	34.1	9.9
6.049	100.0	0.0	97.9	20.1	34.3	10.0
6.063	100.0	0.0	98.3	20.1	34.4	9.9
6.078	100.0	0.0	98.5	20.1	34.6	9.9
6.092	100.0	0.0	98.7	19.5	34.6	10.0
6.106	100.0	0.0	99.1	20.1	34.7	10.0
6.120	100.0	0.0	99.3	20.1	34.9	9.9
6.134	100.0	0.0	99.7	20.1	35.0	9.9
6.149	100.0	0.0	99.9	20.1	35.2	10.0
6.163	100.0	0.0	100.3	19.5	35.4	9.9
6.177	100.0	0.0	100.5	20.1	35.5	9.9
6.191	100.0	0.0	100.9	20.1	35.6	9.9
6.205	100.0	0.0	101.1	20.1	35.8	9.9
6.220	100.0	0.0	101.5	19.5	35.9	9.9
6.234	100.0	0.0	101.7	20.1	36.1	9.9
6.248	100.0	0.0	102.1	20.1	36.3	9.9
6.262	100.0	0.0	102.3	20.1	36.4	9.9
6.276	100.0	0.0	102.7	20.1	36.5	9.9
6.291	100.0	0.0	103.0	20.1	36.7	9.9
6.305	100.0	0.0	103.3	20.1	36.7	9.9
6.319	100.0	0.0	103.5	20.1	36.8	9.9
6.333	100.0	0.0	103.9	20.1	37.0	9.9
6.347	100.0	0.0	104.1	20.1	37.1	9.9
6.362	100.0	0.0	104.5	19.5	37.3	9.9
6.376	100.0	0.0	104.7	20.1	37.4	9.9
6.390	100.0	0.0	105.1	20.1	37.6	10.0
6.404	100.0	0.0	105.3	20.1	37.7	10.0
6.418	100.0	0.0	105.7	20.1	37.9	9.9
6.433	100.0	0.0	105.9	19.5	38.0	9.9
6.447	100.0	0.0	106.3	20.1	38.2	10.0
6.461	100.0	0.0	106.5	20.1	38.3	9.9
6.475	100.0	0.0	106.9	20.1	38.5	9.9
6.489	100.0	0.0	107.3	20.1	38.7	9.9
6.504	100.0	0.0	107.5	20.1	38.8	9.9
6.518	100.0	0.0	107.7	20.1	38.9	9.9
6.532	100.0	0.0	108.1	20.1	39.1	10.0
6.546	100.0	0.0	108.3	20.1	39.2	10.0
6.560	100.0	0.0	108.7	20.1	39.4	9.9
6.575	100.0	0.0	108.8	20.1	39.5	9.9
6.589	100.0	0.0	109.2	20.1	39.7	9.9
6.603	100.0	0.0	109.3	20.1	39.8	9.9
6.617	100.0	0.0	109.5	20.1	40.0	9.9
6.631	100.0	0.0	109.9	20.1	40.1	10.0
6.646	100.0	0.0	110.1	20.1	40.3	9.9
6.660	100.0	0.0	110.5	20.1	40.4	9.9
6.674	100.0	0.0	110.6	19.5	40.6	10.0
6.688	100.0	0.0	110.9	20.1	40.7	9.9
6.702	100.0	0.0	111.1	20.1	40.9	10.0
6.717	100.0	0.0	111.3	20.1	40.9	9.9
6.731	100.0	0.0	111.7	20.1	41.1	9.9
6.745	100.0	0.0	111.8	20.1	41.2	9.9

		Axis 1		Axis 2		Axis 3	
	Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
	6.759	100.0	0.0	111.9	20.1	41.3	9.9
	6.773	100.0	0.0	112.3	20.1	41.5	9.9
	6.788	100.0	0.0	112.5	20.1	41.6	10.0
	6.802	100.0	0.0	112.9	20.1	41.8	9.9
	6.816	100.0	0.0	113.1	19.5	42.0	10.0
	6.830	100.0	0.0	113.5	20.1	42.1	9.9
	6.844	100.0	0.0	113.8	20.1	42.2	9.9
	6.859	100.0	0.0	114.1	20.1	42.4	9.9
	6.873	100.0	0.0	114.3	20.1	42.5	10.0
	6.887	100.0	0.0	114.7	19.5	42.7	9.9
	6.901	100.0	0.0	114.9	20.1	42.8	9.9
	6.915	100.0	0.0	115.3	20.1	43.0	9.9
	6.930	100.0	0.0	115.7	20.1	43.0	9.9
	6.944	100.0	0.0	116.0	20.1	43.1	9.9
	6.958	100.0	0.0	116.1	20.1	43.3	9.9
	6.972	100.0	0.0	116.6	19.5	43.4	9.9
	6.986	100.0	0.0	116.8	20.1	43.6	9.9
	7.001	100.0	0.0	117.1	20.1	43.7	9.9
	7.015	100.0	0.0	117.2	20.1	43.9	9.9
	7.029	100.0	0.0	117.5	20.1	44.0	10.0
	7.043	100.0	0.0	117.8	20.1	44.2	9.9
	7.057	100.0	0.0	118.0	19.5	44.4	10.0
	7.072	100.0	0.0	118.4	20.1	44.5	10.0
	7.086	100.0	0.0	118.6	20.1	44.6	9.9
	7.100	100.0	0.0	119.0	20.1	44.8	9.9
	7.114	100.0	0.0	119.2	20.1	44.9	10.0
	7.128	100.0	0.0	119.6	19.5	45.1	9.9
	7.143	100.0	0.0	119.9	14.9	45.1	9.9
Axis 2 (End)	**7.157**	100.0	0.0	**120.0**	**7.5**	45.3	10.0
	7.171	100.0	0.0	120.0	4.6	45.4	10.0
	7.185	100.0	0.0	120.0	0.6	45.5	10.0
	7.199	100.0	0.0	120.0	0.6	45.7	9.9
	7.214	100.0	0.0	120.0	0.0	45.8	10.0
	7.228	100.0	0.0	120.0	0.6	45.8	9.9
	7.242	100.0	0.0	120.0	0.6	46.0	10.0
	7.256	100.0	0.0	120.0	0.6	46.1	9.9
	7.270	100.0	0.0	120.0	0.0	46.3	10.0
	7.285	100.0	0.0	120.0	0.6	46.4	9.9
	7.299	100.0	0.0	120.0	0.0	46.6	9.9
	7.313	100.0	0.0	120.0	0.0	46.8	10.0
	7.327	100.0	0.0	120.0	0.0	46.9	10.0
	7.341	100.0	0.0	120.0	0.6	47.0	9.9
	7.356	100.0	0.0	120.0	0.6	47.2	10.0
	7.370	100.0	0.0	120.0	0.0	47.2	9.9
	7.384	100.0	0.0	120.0	0.0	47.3	9.9
	7.398	100.0	0.0	120.0	0.0	47.5	9.9
	7.412	100.0	0.0	120.0	0.0	47.7	10.0
	7.427	100.0	0.0	120.0	0.0	47.8	9.9
	7.441	100.0	0.0	120.0	0.0	47.9	10.0
	7.455	100.0	0.0	120.0	0.0	48.1	9.9
	7.469	100.0	0.0	120.0	0.0	48.2	10.0
	7.483	100.0	0.0	120.0	0.0	48.4	9.9
	7.498	100.0	0.0	120.0	0.0	48.5	9.9

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
7.512	100.0	0.0	120.0	0.0	48.7	10.0
7.526	100.0	0.0	120.0	0.0	48.8	9.9
7.540	100.0	0.0	120.0	0.0	49.0	9.9
7.554	100.0	0.0	120.0	0.0	49.2	9.9
7.569	100.0	0.0	120.0	0.0	49.3	9.9
7.583	100.0	0.0	120.0	0.0	49.4	9.9
7.597	100.0	0.0	120.0	0.0	49.6	9.9
7.611	100.0	0.0	120.0	0.0	49.7	10.0
7.625	100.0	0.0	120.0	0.0	49.9	10.0
7.640	100.0	0.0	120.0	0.0	50.0	9.9
7.654	100.0	0.0	120.0	0.0	50.1	9.9
7.668	100.0	0.0	120.0	0.0	50.2	10.0
7.682	100.0	0.0	120.0	0.0	50.3	9.9
7.696	100.0	0.0	120.0	0.0	50.5	9.9
7.711	100.0	0.0	120.0	0.0	50.6	10.0
7.725	100.0	0.0	120.0	0.0	50.8	9.9
7.739	100.0	0.0	120.0	0.0	50.9	9.9
7.753	100.0	0.0	120.0	0.0	51.1	10.0
7.767	100.0	0.0	120.0	0.0	51.2	9.9
7.782	100.0	0.0	120.0	0.0	51.4	9.9
7.796	100.0	0.0	120.0	0.0	51.5	9.9
7.810	100.0	0.0	120.0	0.0	51.7	9.9
7.824	100.0	0.0	120.0	0.0	51.8	9.9
7.838	100.0	0.0	120.0	0.0	52.0	9.9
7.853	100.0	0.0	120.0	0.0	52.1	10.0
7.867	100.0	0.0	120.0	0.0	52.3	9.9
7.881	100.0	0.0	120.0	0.0	52.5	9.9
7.895	100.0	0.0	120.0	0.0	52.6	10.0
7.909	100.0	0.0	120.0	0.0	52.7	9.9
7.924	100.0	0.0	120.0	0.0	52.9	9.9
7.938	100.0	0.0	120.0	0.0	53.0	9.9
7.952	100.0	0.0	120.0	0.0	53.2	9.9
7.966	100.0	0.0	120.0	0.0	53.3	9.9
7.980	100.0	0.0	120.0	0.0	53.5	9.9
7.995	100.0	0.0	120.0	0.0	53.6	10.0
8.009	100.0	0.0	120.0	0.0	53.8	9.9
8.023	100.0	0.0	120.0	0.0	53.9	9.9
8.037	100.0	0.0	120.0	0.0	54.1	9.9
8.051	100.0	0.0	120.0	0.0	54.2	9.9
8.066	100.0	0.0	120.0	0.0	54.2	9.9
8.080	100.0	0.0	120.0	0.0	54.4	9.9
8.094	100.0	0.0	120.0	0.0	54.5	10.0
8.108	100.0	0.0	120.0	0.0	54.7	9.9
8.122	100.0	0.0	120.0	0.0	54.9	9.9
8.137	100.0	0.0	120.0	0.0	55.0	10.0
8.151	100.0	0.0	120.0	0.0	55.1	10.0
8.165	100.0	0.0	120.0	0.0	55.3	9.9
8.179	100.0	0.0	120.0	0.0	55.4	9.9
8.193	100.0	0.0	120.0	0.0	55.6	10.0
8.208	100.0	0.0	120.0	0.0	55.7	9.9
8.222	100.0	0.0	120.0	0.0	55.9	9.9
8.236	100.0	0.0	120.0	0.0	56.0	10.0
8.250	100.0	0.0	120.0	0.0	56.2	9.9

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
8.264	100.0	0.0	120.0	0.0	56.5	10.0
8.279	100.0	0.0	120.0	0.0	56.5	10.0
8.293	100.0	0.0	120.0	0.0	56.6	9.9
8.307	100.0	0.0	120.0	0.0	56.8	9.9
8.321	100.0	0.0	120.0	0.0	56.9	9.9
8.335	100.0	0.0	120.0	0.0	57.1	9.9
8.350	100.0	0.0	120.0	0.0	57.2	9.9
8.364	100.0	0.0	120.0	0.0	57.4	9.9
8.378	100.0	0.0	120.0	0.0	57.5	10.0
8.392	100.0	0.0	120.0	0.0	57.7	10.0
8.406	100.0	0.0	120.0	0.0	57.8	9.9
8.421	100.0	0.0	120.0	0.0	57.8	9.9
8.435	100.0	0.0	120.0	0.0	58.0	9.9
8.449	100.0	0.0	120.0	0.0	58.1	9.9
8.463	100.0	0.0	120.0	0.0	58.3	9.9
8.477	100.0	0.0	120.0	0.0	58.4	10.0
8.492	100.0	0.0	120.0	0.0	58.4	10.0
8.506	100.0	0.0	120.0	0.0	58.6	9.9
8.520	100.0	0.0	120.0	0.0	58.9	10.0
8.534	100.0	0.0	120.0	0.0	58.9	10.0
8.548	100.0	0.0	120.0	0.0	59.0	9.9
8.563	100.0	0.0	120.0	0.0	59.2	9.9
8.577	100.0	0.0	120.0	0.0	59.3	9.9
8.591	100.0	0.0	120.0	0.0	59.5	9.9
8.605	100.0	0.0	120.0	0.0	59.8	10.0
8.619	100.0	0.0	120.0	0.0	59.8	9.9
8.634	100.0	0.0	120.0	0.0	59.9	10.0
8.648	100.0	0.0	120.0	0.0	60.2	9.9
8.662	100.0	0.0	120.0	0.0	60.2	9.9
8.676	100.0	0.0	120.0	0.0	60.4	10.0
8.690	100.0	0.0	120.0	0.0	60.5	9.9
8.705			120.0	0.0	60.7	9.9
8.719			120.0	0.0	60.8	10.0
8.733			120.0	0.0	61.0	9.9
8.747			120.0	0.0	61.3	10.0
8.761			120.0	0.0	61.3	10.0
8.776			120.0	0.0	61.4	9.9
8.790			120.0	0.0	61.7	9.9
8.804			120.0	0.0	61.7	9.9
8.818			120.0	0.0	61.9	10.0
8.832			120.0	0.0	62.0	9.9
8.847			120.0	0.0	62.2	9.9
8.861			120.0	0.0	62.2	10.0
8.875			120.0	0.0	62.4	9.9
8.889			120.0	0.0	62.5	9.9
8.903			120.0	0.0	62.6	9.9
8.918			120.0	0.0	62.7	9.9
8.932			120.0	0.0	62.9	9.9
8.946			120.0	0.0	63.1	9.9
8.960			120.0	0.0	63.2	9.9
8.974			120.0	0.0	63.3	10.0
8.989			120.0	0.0	63.5	9.9
9.003			120.0	0.0	63.6	9.9

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
9.017			120.0	0.0	63.8	10.0
9.031			120.0	0.0	63.9	9.9
9.045			120.0	0.0	64.1	9.9
9.060			120.0	0.0	64.2	10.0
9.074					64.4	9.9
9.088					64.5	9.9
9.102					64.7	10.0
9.116					64.8	10.0
9.131					65.0	9.9
9.145					65.1	9.9
9.159					65.3	10.0
9.173					65.5	9.9
9.187					65.6	9.9
9.202					65.7	10.0
9.216					65.9	9.9
9.230					66.0	9.9
9.244					66.2	10.0
9.258					66.3	10.0
9.273					66.5	9.9
9.287					66.6	9.9
9.301					66.7	9.9
9.315					66.8	9.9
9.329					66.9	9.9
9.344					67.1	9.9
9.358					67.2	10.0
9.372					67.4	9.9
9.386					67.5	9.9
9.400					67.7	10.0
9.415					67.9	9.9
9.429					68.0	9.9
9.443					68.1	9.9
9.457					68.3	9.9
9.471					68.4	9.9
9.486					68.6	9.9
9.500					68.7	10.0
9.514					68.9	9.9
9.528					69.0	9.9
9.542					69.2	10.0
9.557					69.3	9.9
9.571					69.5	9.9
9.585					69.6	10.0
9.599					69.8	9.9
9.613					69.9	9.9
9.628					70.1	10.0
9.642					70.3	9.9
9.656					70.4	9.9
9.670					70.5	9.9
9.684					70.7	9.9
9.699					70.8	9.9
9.713					70.9	9.9
9.727					71.0	10.0
9.741					71.2	9.9
9.755					71.3	9.9

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
9.770					71.4	9.9
9.784					71.6	10.0
9.798					71.7	9.9
9.812					71.9	9.9
9.826					72.0	9.9
9.841					72.2	9.9
9.855					72.3	9.9
9.869					72.5	9.9
9.883					72.7	9.9
9.897					72.8	9.9
9.912					72.9	9.9
9.926					73.0	9.9
9.940					73.1	9.9
9.954					73.2	9.9
9.968					73.4	9.9
9.983					73.6	9.9
9.997					73.7	9.9
10.011					73.8	9.9
10.025					74.0	10.0
10.039					74.1	10.0
10.054					74.3	9.9
10.068					74.4	9.9
10.082					74.6	9.9
10.096					74.7	9.9
10.110					74.9	9.9
Axis 3 (End)	**10.125**				**75.0**	**8.0**
10.139					75.0	0.6
10.153					75.0	0.0
10.167					75.0	0.6
10.181					75.0	0.6
10.196					75.0	0.6
10.210					75.0	0.0
10.224					75.0	0.6
10.238					75.0	0.0
10.252					75.0	0.0
10.267					75.0	0.0
10.281					75.0	0.0
10.295					75.0	0.0
10.309					75.0	0.0
10.323					75.0	0.0
10.338					75.0	0.0
10.352					75.0	0.0
10.366					75.0	0.0
10.380					75.0	0.0
10.394					75.0	0.0
10.409					75.0	0.0
10.423					75.0	0.0
10.437					75.0	0.0
10.451					75.0	0.0
10.465					75.0	0.0
10.480					75.0	0.0
10.494					75.0	0.0
10.508					75.0	0.0

	Axis 1		Axis 2		Axis 3	
Time (sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (deg.)	Speed (deg./sec.)	Disp. (mm)	Speed (mm/sec.)
10.522					75.0	0.0
10.536					75.0	0.0
10.551					75.0	0.0
10.565					75.0	0.0
10.579					75.0	0.0
10.593					75.0	0.0
10.607					75.0	0.0
10.622					75.0	0.0
10.636					75.0	0.0
10.650					75.0	0.0
10.664					75.0	0.0
10.678					75.0	0.0
10.693					75.0	0.0
10.707					75.0	0.0
10.721					75.0	0.0
10.735					75.0	0.0
10.749					75.0	0.0
10.764					75.0	0.0
10.778					75.0	0.0

APPENDIX F – CALCULATION OF R-V ENCODERS ERRORS

Calculation of R-V Encoder Errors

Using the following kinematics parameters

Axis	SPEED	ACCELERATION	DECELERATION	DISTANCE
Axis 1	20	1000	1000	100
Axis 2	20	500	500	120
Axis 3	10	1000	1000	75

and equation [6.6], $t_{total} = \frac{\theta}{\omega_{max}} + \frac{\omega_{max}}{\alpha}$ the respective theoretical value of t_{total} can be calculated;

$$[t_{total}]_{theo1} = \frac{100}{20} + \frac{20}{1000} = 5.020 \text{ sec.}$$

$$[t_{total}]_{theo2} = \frac{120}{20} + \frac{20}{500} = 6.040 \text{ sec.}$$

$$[t_{total}]_{theo3} = \frac{75}{10} + \frac{10}{1000} = 7.510 \text{ sec.}$$

The experimental value of t_{total} can be calculated easily from the experimental data (see Appendix F) by deducting the start time and end time of the motion. Hence,

$$[t_{total}]_{actual1} = 5.566 - 0.554 = 5.012 \text{ sec.}$$

$$[t_{total}]_{actual2} = 7.157 - 1.022 = 6.135 \text{ sec.}$$

$$[t_{total}]_{actual3} = 10.125 - 2.627 = 7.498 \text{ sec.}$$

The difference between the theoretical and actual; t_{total} and the respective percentage of error are calculated as follows:

$$[t_{total}]_{diff1} = |5.020 - 5.012| = 0.008 \text{ sec.}$$

$$[t_{total}]_{diff2} = |6.040 - 6.135| = 0.095 \text{ sec.}$$

$$[t_{total}]_{diff3} = |7.510 - 7.498| = 0.012 \text{ sec.}$$

$$[\%\ error]_1 = (0.008/5.020) \times 100\% = \mathbf{0.159\ \%}$$

$$[\%\ error]_2 = (0.095/6.040) \times 100\% = \mathbf{1.573\ \%}$$

$$[\%\ error]_3 = (0.012/7.510) \times 100\% = \mathbf{0.160\ \%}$$

This CD contains the following materials	
RVdata.pdf	This file contains the R-V velocity and positional data from test case II.
REALdata.pdf	This file contains the real velocity and positional data from test case II
[folder: *real time RV*] *real time RV.mpg* *real time RV.rm* *real time RV.wmv*	The three files in the folder show videos that demonstrate the real time ability of the RV system. All the three videos are the same but in different file format.
[folder: *switch RV to Real*] *switch RV to Real.mpg* *switch RV to Real.rm* *switch RV to Real.wmv*	The three files in the folder show videos that demonstrate the machine controller program developed in R-V environment being used onto the real system. All the three videos are the same but in different file format.