REAL-TIME SIMULATION OF ADVANCED MANUFACTURING SYSTEMS

Ari Edinburg

REAL-TIME SIMULATION OF ADVANCED MANUFACTURING SYSTEMS

Ari Edinburg

A project report submitted to the Faculty of Engineering, University of the Witwatersrand, Johannesburg, in partial fulfilment of the requirements for the degree of Master of Science in Engineering.

Johannesburg, 1991

DECLARATION

I declare that this project report is my own, unaided work. It is being submitted for the Degree of Master of Science in Engineering, the University of the Witwatersrand, Johannesburg. It has not been submitted before for any degree or examination in any other University.

(Signature of candidate)

5th of August 1991

ABSTRACT

In order to analyse and optimize the design and operation of Advanced Manufacturing Systems (AMS) various simulation techniques are currently in use. However, these methods are generally inflexible and are used off-line. In this project report, a modular modelling approach is adopted to investigate the requirements of shop floor device models which operate in real-time. The models, when connected together, simulate the operation of the AMS. A suitable technique for the development of these models is proposed. A simulation architecture to support the models is discussed. It is intended that the models are connected on-line to the AMS control system via a Local Area Network (LAN). Advantages and applications of this approach to AMS simulation are discussed.

ACKNOWLEDGEMENTS

The author is indebted to:

The CSIR Foundation for Research Development for assistance received.

CONTENTS

TABLE OF FIGURES

TABLE OF TABLES

1 INTRODUCTION

New developments in production and manufacture have led to programmable and flexible automated manufacturing systems. Such systems contain automated material handling and storage systems, intelligent processing centres such as computer numerically controlled (CNC) machines and robots along with a control system to co-ordinate and integrate the whole operation.

The flexibility of these systems is apparent in their ability to take new action to cope with market demand for new products. This requires coping with the effects of gearing up to produce new products as well as the ability to manage machine breakdowns, dynamic shop floor scheduling and production monitoring. This type of advanced manufacturing system (AMS) includes a class of systems denoted by a variety of synonyms such as Flexible Manufacturing Systems (FMS), Computer Integrated Manufacture (CIM), Variable Mission Manufacture (VMM) etc, with flexibility and adaptability being the common element (Bloch, 1986).

Manara, Giuffre and Cavagnaro (1982) note that the complexity, novelty and structural differences among the controlled processes of advanced manufacturing systems, has led to the development of multifarious systems on a one by one basis, creating unique samples whose software is poorly maintainable and has to be redeveloped at great expense when changing to new applications or when modifying existing ones.

Further, Brandolese and Garetti (1983) note that this way of proceeding from one prototype to another is extremely costly, offering no accumulation of experience, scale economy, increased dependability and effectiveness of well tested and optimized modules. Bloch, (1986) states that one of the major problems involved in the control of these systems is the need for general, flexible software, capable of being tailored to a wide variety of applications.

The task of developing an advanced manufacturing system computer control architecture, the shop floor layout and the necessary software for off-the-shelf real-time shop floor control is complex. Ideas or philosophies developed towards these aims must be evaluated and verified. One method of achieving this is to evaluate proposed systems with a complete manufacturing system. However, the costs involved and the disruptions to existing systems would be prohibitive.

In an attempt to overcome these problems, the use of a real-time, modular simulation as a development and analysis aid is proposed. The use of real-time models simulating the advanced manufacturing system shop floor equipment would provide a powerful tool, not only for control system and software evaluation, but also for evaluation of existing systems. The use of modular simulation provides advanced manufacturing system building blocks, which allow the construction and simulation of almost any system configuration.

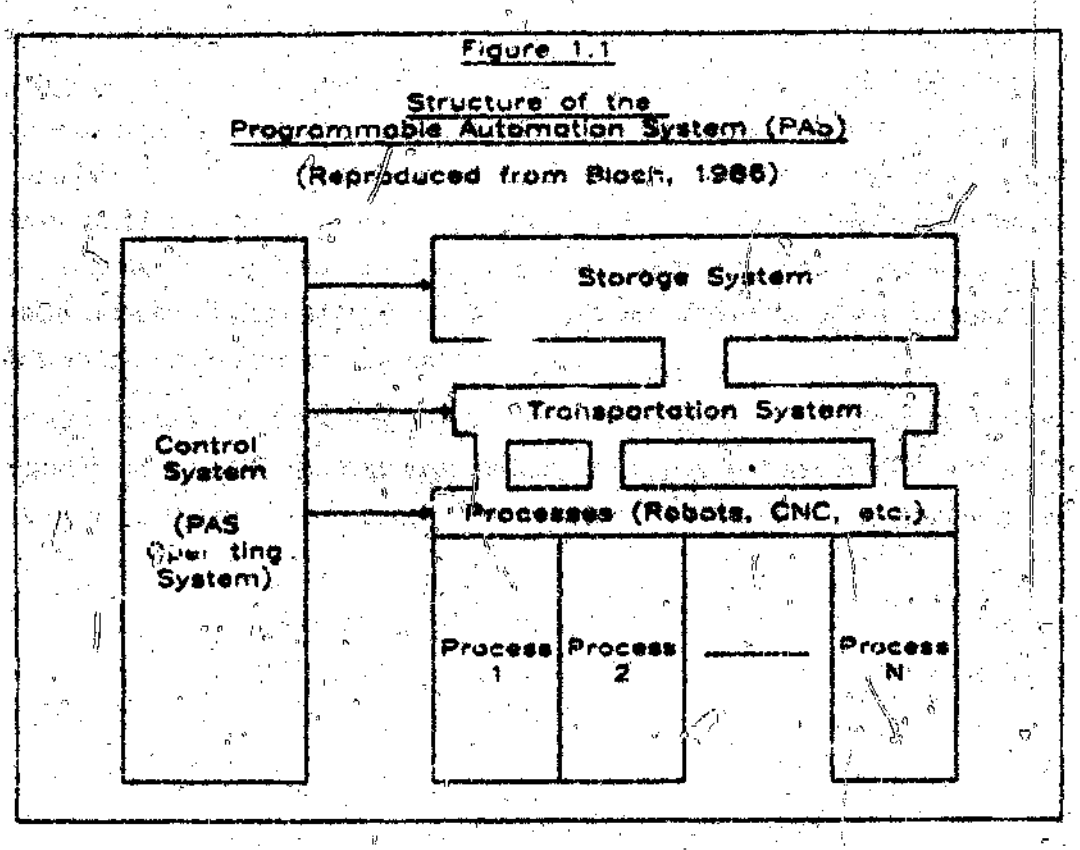
1.1 The Research Objective.

One system, which falls under the category of advanced manufacturing simulation systems, is the programmable automation system (PAS) proposed by Bloch (1986). The PAS system structure is illustrated in Figure 1.1.

Figure 1.1

Structure of the Programmable Automation System (PAS)

(Reproduced from Bloch, 1986)



The storage and transportation system comprises of shop floor equipment units or devices such as automated storage and retrieval systems (AS/RS), robots,

automated guided vehicle systems (AGVS) etc. The processes are extremely broadly based, comprising whatever types of equipment are needed to process the material in the PAS in question.

The storage and transportation systems and the processes constitute the physical system, which is controlled by the PAS control system or more appropriately, the PAS operating system. Bloch (1986) notes that the control system is the element that holds the key to the real success of the PAS and determines the limits and flexibility of the system.

This project report is concerned with the development of a real-time advanced manufacturing simulation system which will form the basis of an investigational tool used in evaluating the requirements and operational aspects of the PAS control system and the physical system.

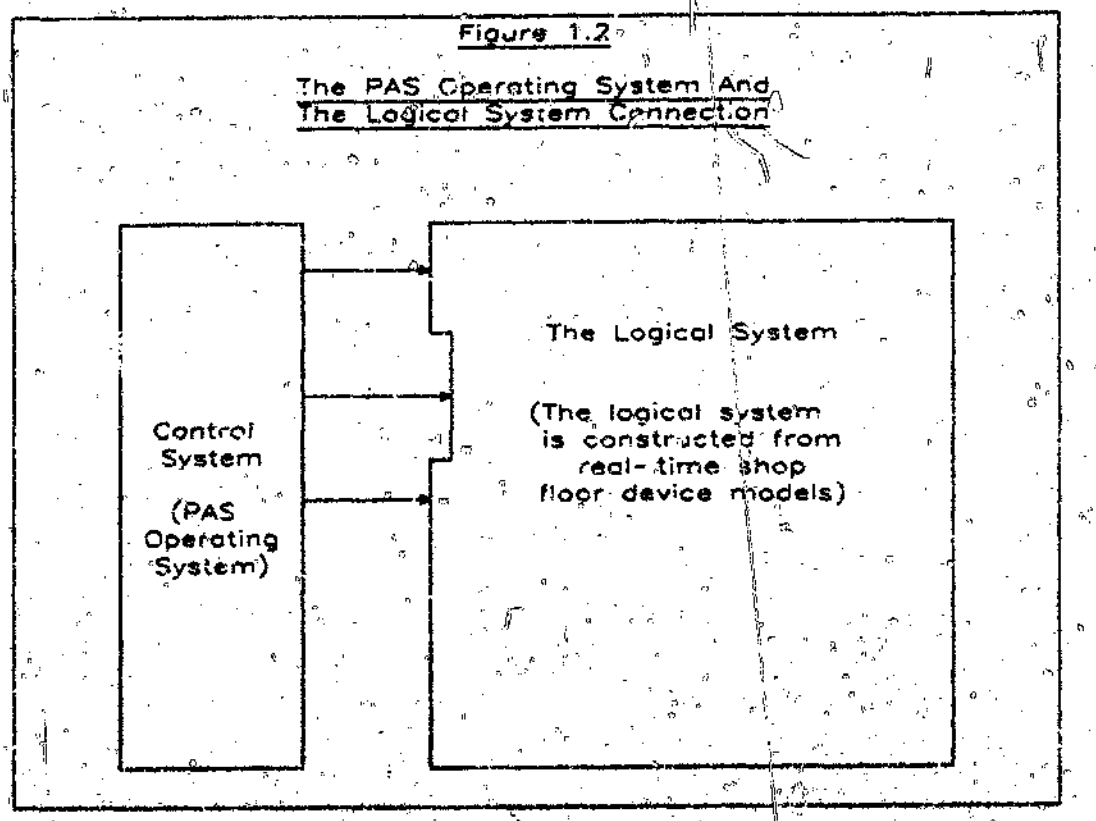
The objective of this project report is to develop a real-time modelling methodology whereby each equipment type or device on the shop floor can be modelled. Based on these models, a simulation system can be constructed which completely replaces the physical system in the PAS, thereby forming a logical advanced manufacturing system as shown in Figure 1.2. The simulation therefore replaces the physical system by an equivalent logical system. The PAS operating system should not be able to distinguish between the operation of the physical system and the logical system running in real-time.

In developing the modelling methodology, each shop floor device or system, namely, the AGVS, the conveyor system, the robot and the AS/RS and their associated controllers are considered in depth. The role of CNC machines are discussed.

In summary, the objective is to propose a method of model construction, description and implementation that will constitute the advanced manufacturing system modelling philosophy of the project report.

Figure 1.2

The PAS Operating System And The Logical System Connection



1.2 The Advanced Manufacturing System Modelling Approach.

1.2.1 The PAS/Physical System Architecture.

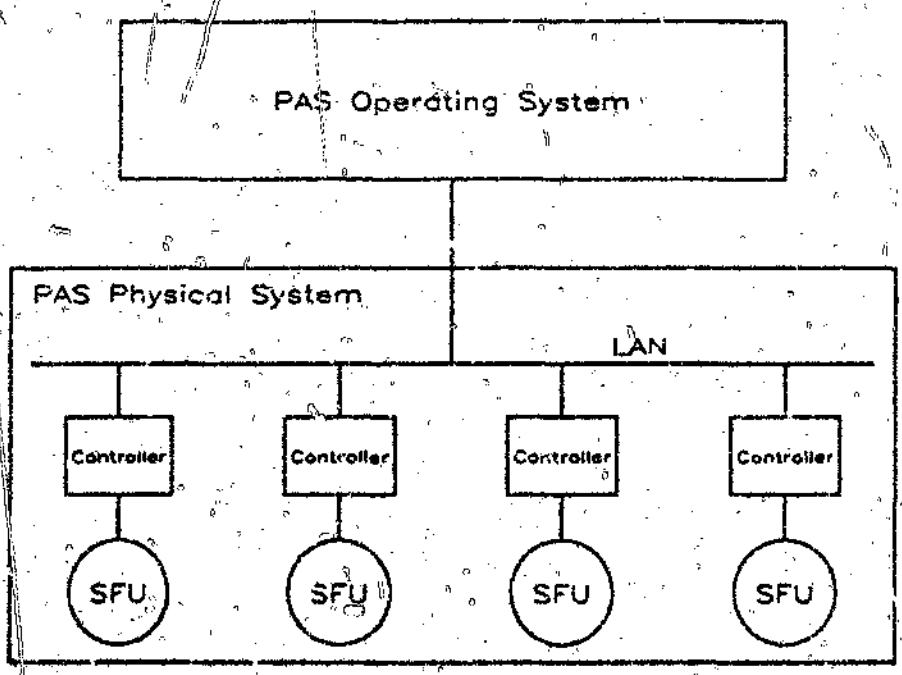
The PAS operating system communicates via a local area network (LAN) with the individual shop floor devices which constitute the physical system, as shown in Figure 1.3 The shop floor controllers consist of programmable logic controllers (PLC), personal computers (PC), robot controllers and CNC machine controllers. The shop floor equipment units or systems consist of the AS/RS, the AGVS, the conveyor system, robots and CNC machines.

The PAS operating system exercises control by sending primitives or commands, which constitute the smallest atomic action that a device can be called upon to perform, to the device controllers. The PAS operating system monitors device operation by status messages compiled by the device controllers.

Figure 1.3

The Programmable Automation System Architecture

(SFU = shop floor equipment unit)



1.2.2 Modelling The Shop Floor Devices.

From the PAS architecture described, it can be seen that the system is modular and a definite, describable, interface exists between the shop floor equipment controllers and the PAS operating system. Bearing in mind that the simulation system architecture should mirror the architecture of the physical system, we intend to model the physical PAS based on the inputs and outputs of each shop floor device.

That is, the interface between each shop floor device the control system must be maintained in the respective device models. The intention is to individually model the shop floor devices (equipment units and controllers) in real-time and these models may be connected onto the LAN in place of shop floor devices. The PAS operating system would not be able to differentiate between the devices and their models. A collection of such models can thus model the PAS physical system and form what is denoted as a logical system. This modelling

approach allows the models to be highly modular and retain the structure of the physical system in the logical system. To achieve this the following is proposed:

(1) The logical system must accurately reproduce the interface between the PAS operating system and physical system. Thus the models must be able to communicate with the PAS operating system. The logical system will be required to execute the operations associated with primitives sent by the PAS operating system. Also information from the logical process must be maintained so that the PAS operating system can read this information for process monitoring and error detection purposes.

(2) The models of the shop floor devices, which constitute the logical system, must be able to interface with each other since they process a logical representation of material. For example: In the physical system, assume that a robot is to deposit a part on a conveyor. In the logical system this would be simulated by some form of message passing between the robot model and the conveyor model. Therefore, a definite interface must be constructed between the models which allows their interconnection and allows the simulation of material transfer.

(3) The last aspect to achieve this approach is the actual modelling of the shop floor devices. These include robots, the AGVS, conveyors, the AS/RS and CNC machines.

In summary, the development of the modelling approach consists of three areas of investigation.

1) The PAS operating system and logical system interface.

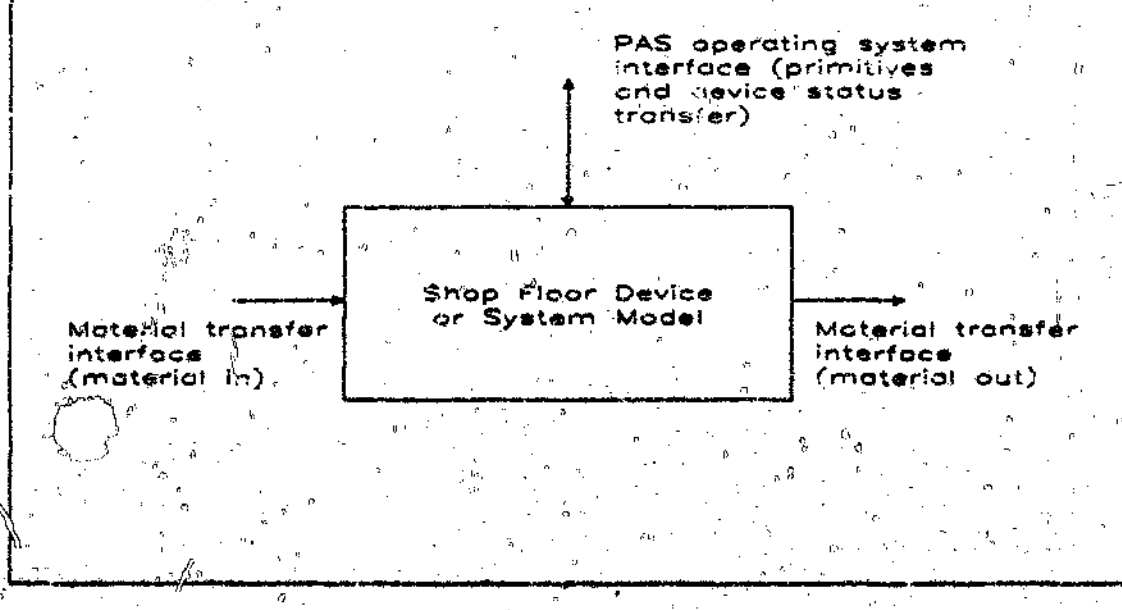
2) Interfacing shop floor devices or systems for the simulation of the transfer of material between them.

3) Modelling the shop floor devices to allow real-time simulation, specific types of devices and the various configuration possibilities.

The conceptual model for each shop floor device or system is presented in Figure 1.4.

Figure 1.4

Generalised Shop Floor Device Or System Model



1.3 An Architecture For Simulation.

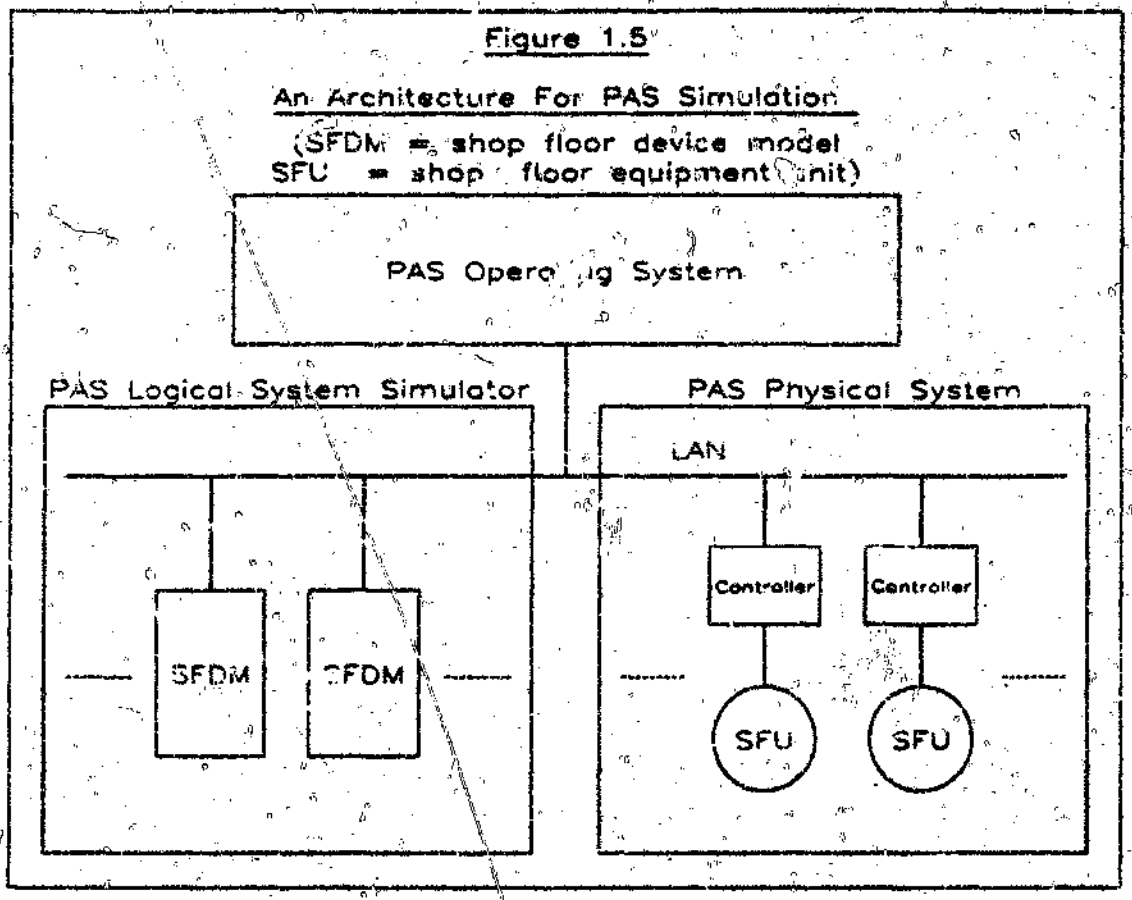
The proposed architecture for simulation of the physical system is shown in Figure 1.5. The simulator is connected on-line to the PAS operating system or control system. Each shop floor device model can communicate with the PAS operating system for the transfer of commands and information. The models can communicate with each other, simulating the transfer of material between them.

Although the concepts behind the real-time models requires that the simulator always be connected on-line to the PAS operating system, it may be connected on-line or off-line with respect to the physical system. Due to the approach adopted in the previous sections the following applications in simulation of the PAS physical system are feasible:

Figure 1.5

An Architecture For PAS Simulation

(SFDM = shop floor device model
SFU = shop floor equipment unit)



Applications of simulation fall into four distinct categories: On and off line real-time simulation. On and off line faster-than-real-time simulation.

Real-time simulation refers to the case where the outputs of the models occur in the same sequence as the outputs of the shop floor devices. Also, the time that the models take to provide an output based on an input will be similar to the time taken by the equivalent shop floor devices.

Faster than real-time simulation refers to the case where the outputs of the models are in the same sequence as the outputs of the shop floor devices, but are generated before those of the physical devices. This allows the prediction of the future behaviour of the device outputs ahead of time in the real system.

Off-line simulation refers to the case where the simulator has no knowledge of message transmissions between the PAS control system and the physical shop floor devices. For off-line simulation, we assume that a separate area of the PAS operating system is devoted to simulation control and applications.

On-line simulation is possible when the simulation system can receive messages from the PAS control system and the actual shop floor devices.

We envisage that the simulation system, as proposed, will find use in the following application areas:

Off-Line Simulation

(1) Conventional simulation applications.
(2) Use of simulation for PAS operating system design.
(3) Operator training.

On-Line Simulation

(1) System testing.
(2) Look ahead capability.
(3) Fault detection.

In all the above applications it is assumed that adequate real-time equipment models of the PAS are available. We now discuss each application in turn.

1.3.1 Off-Line Simulation

(1) Conventional Simulation Applications.
The usual reason for carrying out a simulation is to obtain results from which one can infer aspects of the behaviour of the real system.

(2) Use of Simulation for PAS Operating System Design. PAS operating system design requires simplified models of the physical system. It is often much easier to carry out experiments on a simulation in order to identify a suitable simplified model than it is to perform such experiments on the real system. It is also customary to use simulation to check the performance of the proposed designs.

(3) Operator Training.
After the PAS shop floor devices have been commissioned and assuming the models are accurate enough, the simulator may be used to train operators without affecting the operation of the actual PAS physical system.

1.3.2 On-Line Simulation

(1) System Testing.
This application allows the operating system to be completely tested before the shop floor devices are installed. This can be used to verify the selection of off-the-shelf, real-time control software for the particular AMS.

(2) Look Ahead Capability.
In this application we use the capability of a computer simulation to predict the outputs faster than they occur in the physical system. This ability can be used to advantage in the control of the devices on the shop floor.

As an indication of the possibilities here, consider the case of the effect of adopting a certain scheduling decision. This decision can be simulated to determine its effect on the operation of the physical system. The outcome of the simulation can be used to decide whether to use the particular decision or not. At a more complex level, feedback from the simulator can be used to automatically verify that the operating system has made the correct decision.

(3) Fault Detection.
In this application the simulator runs in real-time and it is on-line with the physical system. The shop floor device responses and their timing can be monitored and compared with the models and if they do not agree to some specified tolerance, fault recovery action can be initiated.

The above-mentioned applications give an indication of the intended use of a simulation system based on the modelling approach adopted in this project report. The applications presented are based on the use of multi-computer distributed processing systems for real-time control in the process control field. The applications in process control are discussed by De Almeida and MacLeod (1985).

1.4 The Modular Modelling Approach.

The modelling approach used in this project report is based on a modular modelling approach using an object-orientated system decomposition methodology based on abstraction and information hiding. We incorporate aspects of distributed simulation in the approach so that a real-time simulation algorithm can be formulated and the resulting modular structure can be maintained to the implementation stage.

1.4.1 Abstraction And Information Hiding.

A major approach to problem solving is abstraction, which involves model building. This is because abstraction is the extraction of essential properties while discarding irrelevant detail. Because people manage complexity by means of abstraction, abstraction provides a view of the essential components and processes that define a system (Ford and Wiener, 1985).

Information hiding makes details which should not effect other parts of a system, inaccessible. Information hiding suppresses how an object or operation is implemented thus focussing attention on the higher abstraction (Booch, 1983).

Modules incorporate the concepts of abstraction and information hiding. This is because by encapsulating our abstractions by modules, we can understand the essential operation of each module. By specifying a module's interface, we define parts of the module which may effect other parts of the system. The essential features of the module are maintained in its interface since we can identify module operation from its inputs and outputs. In this way, our abstractions are incorporated in each module.

By encapsulating our abstractions of a real-world system into modules and specifying module interfaces, we can define a system structure, where the modules and inter-relationships allow us to understand total system operation on a module by module basis. We can further structure our conceptual system by forming lower level abstractions. In this way we can model the real-world system at one specific level of detail at a time.

The above constitutes a description of the concepts behind the modular modelling approach. The modular approach consists of decomposing the real-world system into modules using abstraction and information hiding. The module interfaces are then specified which identifies how each module interacts with other modules in the system.

1.4.2 Advantages Of A Modular Modelling Approach.

Further reasons for adopting a modular modelling approach are enumerated:

1) The real-world system may be decomposed into a module structure which mirrors the structure of the real-world system. This allows modules to be interconnected as would the functional units of the real-world system. The flexibility of module interconnection, coupled with the ability of the simulation system to measure variables which are not so readily measured in the real-world, can be used to identify inflexible real-world interconnections.

2) A simulation system composed of modules with well defined or standard interfaces provides a flexible structure for simulation. Modules may be interconnected at will, to form the desired logical system simulation.

3) System maintenance is effected by the simple addition or removal of modules. Also implementation aspects of modules may be easily changed to incorporate new concepts or more accurate implementations with respect to the role of the module in the simulation system as long as the module interfaces are maintained.

From the above, one should note that the definition of interfaces is one of the most important aspects of module description since this identifies the flexibility of the module interconnections.

1.4.3 A System Decomposition Approach.

In this project report, the object-orientated development approach, based on abstraction and information hiding, is adopted as a method for system decomposition. Object-orientated design may be identified by the following steps (Booch, 1986):

1) Define an informal strategy for the problem solution. Since we are using object-orientated design as a modelling tool, the informal strategy consists of a description of the system which is being modelled.

2) Identify objects (nouns) used in the informal strategy.

3) Establish the visibility of each object in relation to other objects.

4) Establish the interface to each object.

5) Implement each object at a software level.

Each module in the system denotes an object from the problem space. An object has two parts: An outside view and an inside view. The outside view captures the abstract behaviour of the object or module, while the inside view indicates how this behaviour is implemented.

In this project report we are concerned with the outside view of each module (the module's interface or definition) with respect to the PAS operating system. The inside view (the module's implementation) is considered with respect to implementing module behaviour in terms of module inputs and outputs in real-time.

1.4.4 Concepts Of Distributed Simulation.

Concepts of distributed simulation as proposed by Misra (1984) are used in the modelling approach since it allows us to maintain the parallelism inherent in the real-world system and mirror the modularity of the real-world system, not only at a conceptual level but at an implementation level as well.

Distributed simulation utilizes an architecture which is similar to that of a general real-time distributed computer control system (DCCS). Advantages inherent in a DCCS such as fault tolerance, increased reliability, modularity and increased performance (De Almeida and MacCleod, 1985) will apply to the simulation system.

The use of a distributed architecture requires that a number of distributed real-time programming problems must be overcome. These problems apply to

both the PAS and the simulation system and include problems such as synchronization, flow control and the design of suitable naming and addressing schemes (MacCleod, 1986).

The concepts of distributed simulation in terms of the PAS simulation using Misra's terminology, are presented.

Each shop floor device constitutes a physical process (PP) and represents some component of the system to be simulated. Each physical process operates autonomously except to interact with other physical processes in the system The interaction is the transfer of material or the transmission of commands and information to and from the PAS operating system.

In the distributed simulation system, the physical processes are called logical processes (LP). Logical processes form the logical system and interact with each other and the PAS operating system by message passing. The messages model the interactions of the physical processes. Logical processes can be further decomposed into a number of logical processes (identified by the modular modelling approach).

The logical system simulates the operation of the physical system in real-time, if it is possible for the logical system, based on the input messages received, to predict the times and exact sequence of message transmissions which model the physical system operation. For example, a message from one LP to another LP at time T models the transfer of material between the corresponding PPs at time T.

Note that a LP and a PP may both be identical from a PAS operating system point of view, but behave differently since an LP is a model of a PP.

.5 Chapter Summary.

This chapter has introduced the architecture of the Programmable Automation System (PAS) which is the system to be simulated in real time. Such a simulation will allow investigation of the requirements and operational aspects of the PAS.

The real-time simulation of the PAS is to be based on a distributed simulation architecture. This yields many advantages inherent in a distributed computer control system and provides for various applications of real-time and faster than real-time simulation.

A modular modelling approach is proposed for modelling the constituents of the PAS. The modelling approach is based on an object-orientated system development utilizing concepts of distributed simulation (as proposed by Misra, 1984) for simulation implementation.

1.6 The Contents Of The Project Report.

The structure of the project report is based on the modular modelling approach. First the model external view and then its internal view are discussed.

Chapter 2 presents an introduction to advanced manufacturing systems and then an overview of current AMS simulation approaches. Based on the literature covered, we highlight the reasons that current methods are not adequate and then discuss aspects of simulation which would be desirable in an AMS simulation.

Chapter 3 defines the logical system building blocks. These are the models available to the user to build up his logical system. Building block models are presented for the shop floor devices under discussion. The reasons for choosing the particular building blocks are discussed.

Chapters 4 and 5 discuss the building block model interfaces. Chapter 4 presents a material transfer interface which models material transfer between models. In this chapter the term "material" is defined and a common material transfer interface is proposed. The interface operation is defined in terms of a material transfer protocol and a structure for protocol implementation is suggested.

Chapter 5 describes the building block model interfaces required so that the PAS operating system cannot differentiate between the physical system and logical system. This is achieved in terms of control and automatic identification interfaces. A signal interface is also discussed which models the shop floor equipment and its controller interconnection where applicable.

Chapter 6 discusses the requirements for building block model implementation. This is in terms of logical models, mathematical models and a real-time simulation algorithm. Presenting model operating information to the user during a simulation run is also discussed. The implementation structure, the logical and mathematical models for each shop floor device model cannot be generalised. Therefore, for each shop floor device model, they are discussed in the appendices associated with this chapter.

Chapter 7 maps an existing distributed simulation system computing architecture to the case of the AMS simulation system. The architecture is discussed with respect to connecting the logical system to the PAS operating system, logical system construction and the animated display of logical system operation.

Chapter 8 discusses work completed towards implementing a demonstration system for evaluating model operation and hence validating the models and the concepts behind them. Selection of a hardware system, a software development environment and a kernel for demonstration model implementation purposes is discussed.

Chapter 9 concludes the project report. In this chapter the modelling methodology is discussed and the requirements for distributed simulation presented. Areas for future work are enumerated and the concluding statements are made.

1.7 Terminology Of The Project Report.

1.7.1 Abbreviations

The following are the abbreviations used in this project report:

AGV	automated guided vehicle
AGVS	automated guided vehicle system
AIS	automatic identification system
AMS	advanced manufacturing system
AS/RS	automated storage and retrieval system
CIM	computer integrated manufacture
CNC	computer numerically controlled
CPU	central processing unit

DCCS	distributed computer control system
FMS	flexible manufacturing system
I/O	input and output
LAN	local area network
LP	logical process
MS-DOS	Microsoft disk operating system
PAS	programmable automation system
PC	personal computer
P/D	pickup and deposit
PLC	programmable logic controller
PP	physical process
PU	processing unit
S/R	storage and retrieval
UWTec	University of the Witwatersrand Technology Centre
VMM	variable mission manufacture

1.7.2 Definition of Terms

The following terms used in the project report are defined as follows:

Model	Models define the simulation system user view, ie: entities that make up the logical system and are visible to the system user. Each model is the "computer program" that resembles the operation of the physical device.
Module	A module describes the entities which are not visible to the user and define the internal view of models. A model may be made up of one or more modules which together implement the model. A module is dependant of the level of abstraction.
Shuttle	This is a mechanism for moving workpieces between work cells in the AMS.

2 OVERVIEW OF ADVANCED MANUFACTURING SYSTEM SIMULATION

This chapter introduces the concepts of AMS and its constituents. An overview of simulation techniques currently in use is presented. Based on the shortcomings of the current techniques in use, the more useful aspects, which should be incorporated in any AMS simulation, are discussed.

2.1 Introduction To Advanced Manufacturing Systems.

Advanced Manufacturing Systems (AMS) are flexible systems which are capable of performing many different operations in different order, with practically no set-up time between the different operations. The AMS is an integrated system of tools, materials handling devices and support equipment all under computer control. The individual machines have automatic tool changers to enable them to perform sequences of operations and also to switch to different tasks without incurring set-up delays (Browne and Rathmill, 1983). The purpose of this is to achieve an efficiency comparable to that of automated mass production (transfer line) with a high degree of flexibility comparable to that of a job shop. The general benefits of AMS are as follows (Ranky, 1983):

Greater productivity by means of a greater output and a lower unit cost on a 45-85% smaller floor space.

More uniform and consistent products.
Increased reliability due to feedback.
Lead times reduced by up to 50-75%.
High utilization of machine tools.

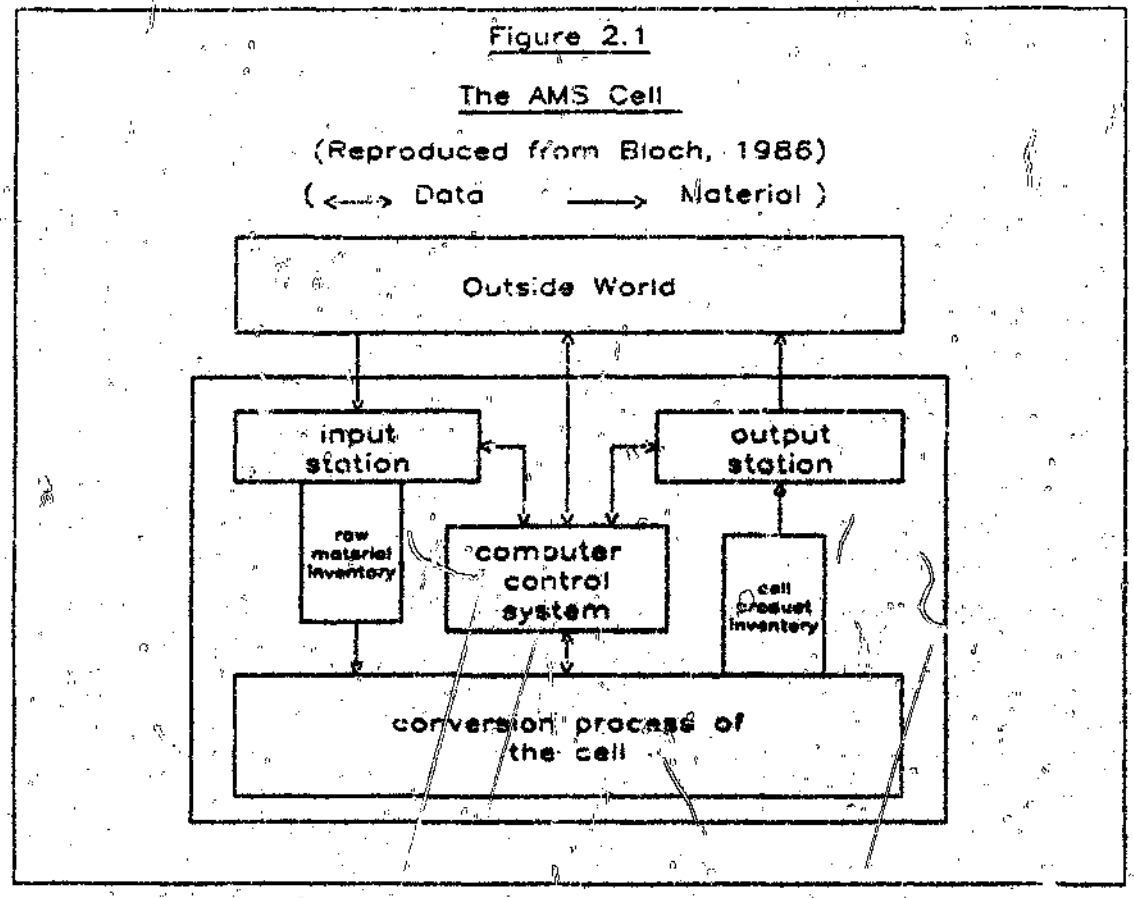
AMS can be thought of as a distributed management information system linking together intelligent sub-systems of machining, welding, painting, flame cutting, sheet metal manufacturing, inspection, assembly etc. along with material handling, identification and storage systems (Ranky, 1983) and a computer control system to co-ordinate and integrate the complete operation.

An AMS consists of a number of cells. These cells are complex for a number of reasons. They can have different physical characteristics to perform the required conversion on workpieces or sub-assemblies. They can have a variety of inter-connections with different material handling systems (robots, automatic guided vehicles, conveyors etc.) and they have to communicate with data processing networks for successful integration with the rest of the system (Ranky, 1983). A simplified illustration of an AMS cell is shown in Figure 2.1.

Figure 2.1

The AMS Cell

(Reproduced from Bloch, 1986)



The illustration shows a cell with its input and output indicating its basic functions. These are to perform the conversion process, to allow buffer storage and/or holding of workpieces, to provide the physical links to the materials handling system in the outside world and to provide the data communication links to the control system. The physical links allow the transport of parts, tools, pallets etc.

The data processing links enable communications with the databases containing part programs, inspection programs, robot programs, machining data, real-time AMS control data etc. They also enable the feedback of data to the upper level of the control hierarchy, thus providing the facility for analysis of performance for real-time fault recovery (Ranky, 1983).

Part and tool transportation inside the cell can be performed by a variety of equipment such as pallet changing devices, automated pallet storage devices and part changing devices along with tool changing manipulators. These devices are necessary for the stand-alone and often unmanned operation of the cell.

The main conversion process itself can be machining (Eg: turning, milling, boring, grinding, cutting, punching etc.) chip removal (part washing and swarf retrieval), welding, painting, automated assembly, inspection, packaging, storage etc. The product of the cell usually leaves through a buffer store.

The input and output buffer stores, as well as the materials handling system, can be the same transport service assuming that its capacity is large enough.

The output of the cell is the product of that particular module of the AMS. The output will consist of a finished or semi-finished part. The part will also be associated with data in a computer readable format. This information will be used to determine subsequent processing for the part and will be distributed over the communication network as required (Ranky, 1983). The significant components of an AMS are discussed. These include:

The automated storage and retrieval system (AS/RS).
The material handling system.
Robots.
The automatic identification system.

2.1.1 Automated Storage And Retrieval Systems.

AS/RS systems are vital to the automated factory and provide on-line buffers between production equipment handling small parts and unit loads of practically any size and shape. An AS/RS consists of a number of Storage and Retrieval (S/R) machines moving in aisles between rows of shelves containing pallets of raw materials, in-process items, or finished goods. The automation

of this process allows the maintenance of an inventory which is automatically updated as items enter or leave storage. The key here lies in the automation of the storage and retrieval process since information pertaining to the goods and inventory are made instantly available to the rest of the system through computer networks. Hence delays and lags normally associated with manual systems do not arise (Bloch, 1985).

2.1.2 Material Handling System

The material handling system includes robots, conveyors, part feeders, palletizers etc. However, one of the most important is the automatic guided vehicle (AGV). The AGV has the ability to deliver materials over flexible routes and to interface automatically with production machinery. AGVs are used to deliver work in progress to production machines, to maintain small quantities of buffer stock on the shop floor or to carry finished goods from inspection to shipping (Bloch, 1985).

In future, AGVs which can move completely under the guidance of computers will be used exclusively in AMS. The AGVs will move without the guidance of buried cables or reflective strips painted on the shop floor. This will transform future automated manufacturing systems into a collection of flexible cells making each machine on the shop floor accessible at will. Thus, a single, large, and highly flexible AMS will be created (Barash, 1986).

2.1.3 Robots

Although robots are usually part of the manufacturing cell, their development is one of the prime reasons for the existence of AMS. It is therefore, necessary to discuss their contribution.

In the cell, robots are used as process machines and more generally as universal transfer devices which load machine tools, palletize products and pick up parts off conveyors. With their flexibility and ability to duplicate the motions of the human hand, industrial robots are ideal for the integrated operation of the AMS as a general material handling device.

Robots operate according to a program and are able to handle parts and products of almost any shape and configuration. Robots can also precisely duplicate their motion and this repeatability allows goods of an extremely consistent and high quality to be produced (Bloch, 1985).

2.1.4 Automatic Identification Systems

Automatic identification systems are intelligent sensors on the computer network that runs the automated factory. Without such systems factory automation would be virtually impossible to achieve. The system provides information which allows maintenance of accurate production and inventory records and thus the ability to determine part routing from one workstation to the next. By closing this loop between information flow and material flow better decisions affecting the optimum performance of the complete system can be made (Bloch, 1985).

So far the elements necessary for the operation of truly flexible automated production or manufacturing system have been discussed. One will notice that computers, which form an integral part of the AMS operation, orchestrate the complete production process. Figure 2.2 illustrates the concept of the integrated information and material flow in unmanned factories. The reader is referred to Ranky (1983) for more detail.

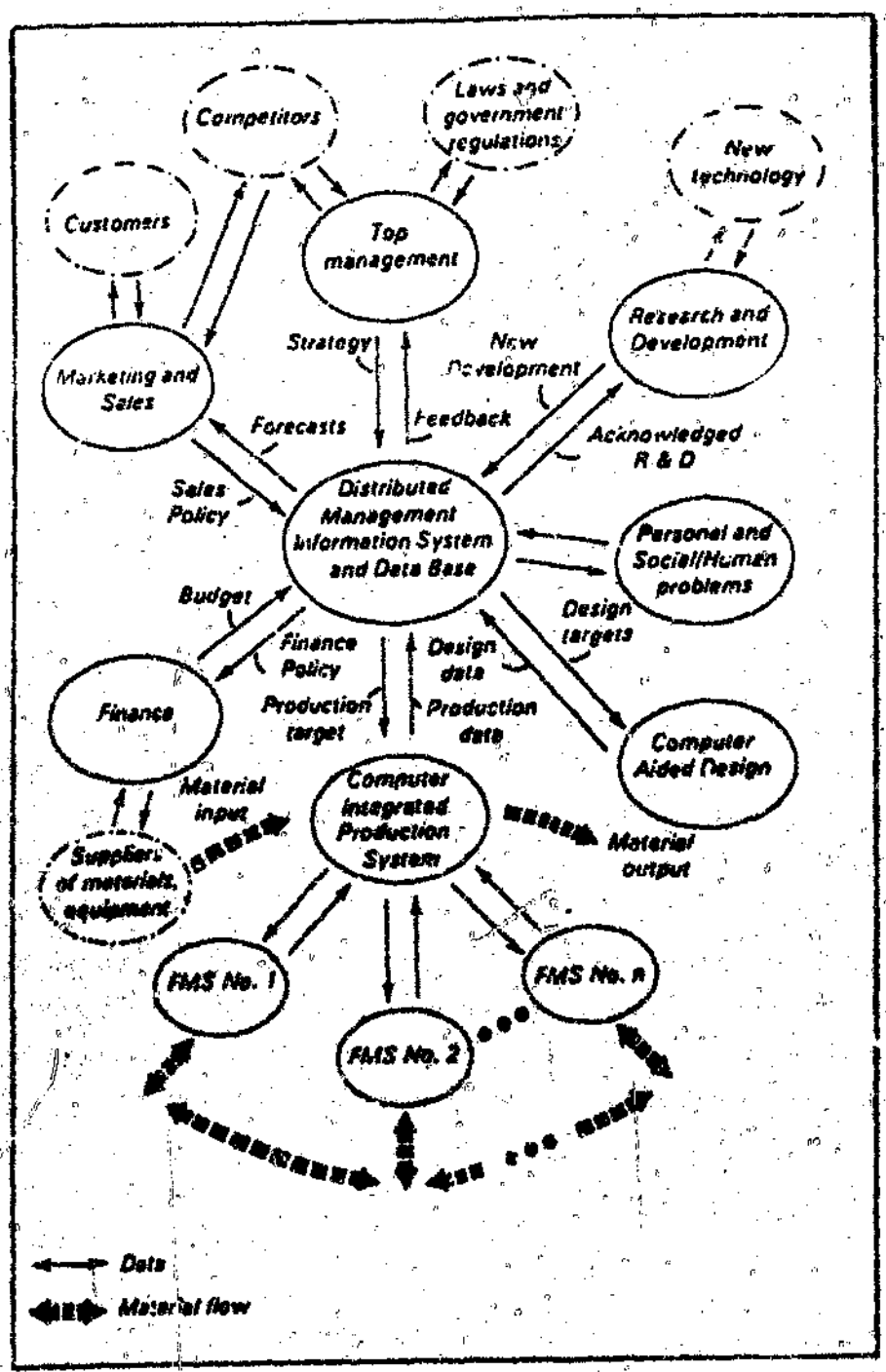


Figure 2.2

Integrated Information Flow and Material Flow in Unmanned Factories (Reproduced From Ranky, 1983)

2.2 The Need For AMS Simulation.

Kay and Walmsley (1982) note that the appearance of increasingly automated systems within the batch production area has created design and operation problems that cannot be solved by traditional production engineering methods. In particular AMS have three characteristics that highlight the need for new design and analysis tools:

High capital investment.
High system complexity.
High risks due to the nature of the technology.

The introduction of new technology and its subsequent operation can be a daunting exercise for any company. With the sums of money involved it is important to get it right. This applies to the number of machines up to the control system and not least, to the time it takes to install, commission and operate.

Typically a single machine might cost £80,000-£400,000. A large system may cost £500,000 to £10 million (Johns, 1985). It follows that mistakes in design and operation could be extremely costly. The risk of course, increases with the complexity of the project, especially where extensive integrated controls are involved in an automated plant.

Planned improvements in manufacturing efficiency are usually piecemeal, arrived at improving floor-to-floor times of individual operations and savings in labour costs to the possible detriment of work in progress, warehousing costs, overall control effort etc. Now, more than ever, there is a need to analyse the system as a whole instead of its individual elements. To assist this task, computer based design and analysis tools are available to aid both development and understanding of AMS (Kay and Walmsley, 1982).

Rahnejat (1986) notes that analytical methods have been developed to study the system dynamics and calculate outputs. Because of the complex nature of manufacturing processes, suitable analytic models cannot be easily constructed to deal with the general characteristics of such systems. In most cases a number of assumptions are usually made. The major assumptions are:

Infinite capacity at every waiting queue.
Zero defective rate.
Perfect reliability of system resources.
Normal or Poisson arrival of parts.
Fixed transfer times and exponential processing durations.

The consequence of such assumptions is to obtain resource utilization rates, estimates of production rates and queuing times which correspond to steady state working characteristics. Therefore, the use of these methods is more suitable in prediction of performance in systems with limited transient nature, such as continuous flow systems or fully automatic cells with long mean time between failure characteristics.

Further, Rahnejat (1986) notes that, in general, the use of analytic techniques is restricted by the following:

(1) modelling constraints in solving real and complex problems which involve a number of parameters that give rise to transient behaviour (such as on-line scheduling) and this changes the system state space.

(2) computational constraints in the available memory needed to store all the possible system states and configurations as well as long run time requirements which are necessary for executing such sizeable code.

Therefore, only simple automated systems can be realistically modelled and assessed by existing analytic methods. While scheduling, network optimisation and applied queueing approaches offer some solution to specific manufacturing problems, by and large they lack modelling accuracy because of their underlying assumptions (Rahnejat, 1986).

Early AMS had, at first, a very low utilization factor, a fraction of the design values. Rules for scheduling the batches for different part types that were known to work in conventional job shops proved unreliable when applied to AMS. Since events happened much faster in AMS than in a conventional job shop, there was no time for human intervention. In other words, the speed of information processing and decision making as well as the memory needed for operating an AMS are far beyond the human ability (Barash, 1986).

This is especially the case now as AMS become more complex. The new generation of systems are designed to process large numbers of part types. The menu of a system recently commissioned in a U.S. aerospace company reaches 500 (Barash, 1986). Moreover today's AMS not only move workpieces but also replacement tools in cassettes or transportable magazines. The greatly increased traffic and more frequent "last minute" changes in production orders make these systems highly complex (Barash, 1986).

University researchers who became interested in AMS introduced computer simulation as a possible management and design tool (Barash, 1986). Simulation has generally been applied in problem areas where the variables and their relationships are clearly understood and can be clearly stated but where no efficient analytical means of problem solving exist. The major advantages of simulation and this applies in particular in the analysis of AMS, is that it permits controlled experimentation with little or no disturbance to the actual system. A well designed and implemented simulation may provide insights into the workings of a complex system (Browne and Rathmill, 1983).

Simulation, as well as providing the optimum method of comparing alternative system approaches during the design and operation stages, also enable increased confidence to be developed at all levels of the AMS project.

A dynamic (animated) graphical simulation model, especially in colour, is more readily acceptable than sets of figures. It seems to be a real system rather than an abstract concept. Hence the use of such simulation systems gives the following benefits (Mills, 1983):

(1) selling the concept to management. The objectives are to show the thoroughness of the planning undertaken and to evaluate the operation of the AMS.

(2) The use of the simulation system to investigate any potential bottleneck areas as well as for developing different control and scheduling strategies.

(3) The personnel who will operate the system can be given hands on experience using the simulation system. This can reduce time spent commissioning the system and the time spent introducing the operators to the system.

(4) Monitoring the operation of the AMS and optimizing its performance.

Welch (1983) defines computer simulation as the process of designing and constructing a mathematical, logical model of a real-world system and experimenting with this model on a computer. A model is an abstraction or description of the real-world system in terms which may be represented on a computer. Thus, in order to develop a model, the system being examined must be represented by a set of key indicators or variables. A distinct set of these variables will define a system state. The manipulation of these variable values will simulate the movement of the system from one state to another.

Being an abstract mathematical-logical representation of a real system gives computer simulation the following advantages (Welch, 1983):

(1) Versatility: computer modelling is versatile and may be used to represent a variety of real-world systems.

(2) Flexibility: computer models are flexible and may easily be altered to represent different situations or accommodate updated information.

(3) Cost effective: Experiments using computer simulation enable the performance of a system to be readily investigated without building the physical system.

(4) Non-disruptive: simulation experiments permit a system to be designed or updated and analysed without disrupting the existing system.

(5) Exhaustive: simulation experiments may be performed under every conceivable set of system conditions, parameters or operating conditions.

In order to achieve these benefits, the physical system must be accurately represented in the simulation model (Welch, 1983).

In summary, there is no real alternative to the use of simulation which can demonstrate the high degree of confidence required by the level of investment in AMS. Conventional planning techniques have been proved unable to cope with the complex and dynamic nature of AMS. Additionally they cannot easily deal with the total systems approach needed to enable these complex amalgams to be successfully handled (Mills, 1983).

Because of the complexities of the system, analytical models can be constructed only with unrealistic simplifying assumptions which, are at best, only able to analyse the system at steady state. But in actual practice a system can hardly reach steady

TABLE 3.1 Table of Conveyor System Building Block Models	
Model Name	**Description of Model Operation**
Conveyor Transport Model	This model is used to transport material from one point to another, in both directions of conveyor motion.
Trap Model	This model traps material ie: stops the material motion and is used buffering or sorting applications.
Trap and Clamp Model	This model traps material and then clamps it in a logically defined position and orientation. Material is said to be located if it is trapped and clamped in such a model. This model is used as an interface to all building block models which require material to be located precisely before transfer.
Diverging Branch Model	This model is used to actively divert material in one of two possible directions.

3.5 The Robot System As A Logical System Building Block.

The robot system is a special case of a logical system building block. The robot system building block consists of a robot model and a robot workcell model as shown in Figure 3.5.

The robot system always includes a robot model and may or may not include the workcell model. The robot workcell model is needed for modelling robot operation as a material processing device compared with robot operation as purely a material handling device. This is the case for assembly and fabrication processes such as spray painting etc. These models are discussed in more detail in Appendix H. Since the robot controller is dedicated to robot control it is included in the robot model.

state because part types with different processing requirements are constantly released to the system at short and irregular intervals governed by the ever changing manufacturing plan. In such a situation it is more pertinent to analyse the transient behaviour of the system, and digital simulation becomes the only viable tool to perform the task (Cheng, 1985).

2.3 Current Simulation Techniques Used In Practice.

In this section physical and digital simulation techniques used in AMS simulation are presented.

2.3.1 Physical Simulation.

A relatively new method of AMS analysis is physical simulation. A physical simulation is an operational scale model of the actual AMS. When controlled by a computer, a physical simulator can be used to obtain operational data from the system. This data may then be used as an aid in the design, installation and operation of the actual system (Diesch and Malstrom, 1985).

Physical simulators have been developed and applied for a number of purposes including:

(1) demonstrating a complex, realistic, operating facility.

(2) analysing, experimenting with and evaluating the system design.

(3) training by providing "hands on" experience.

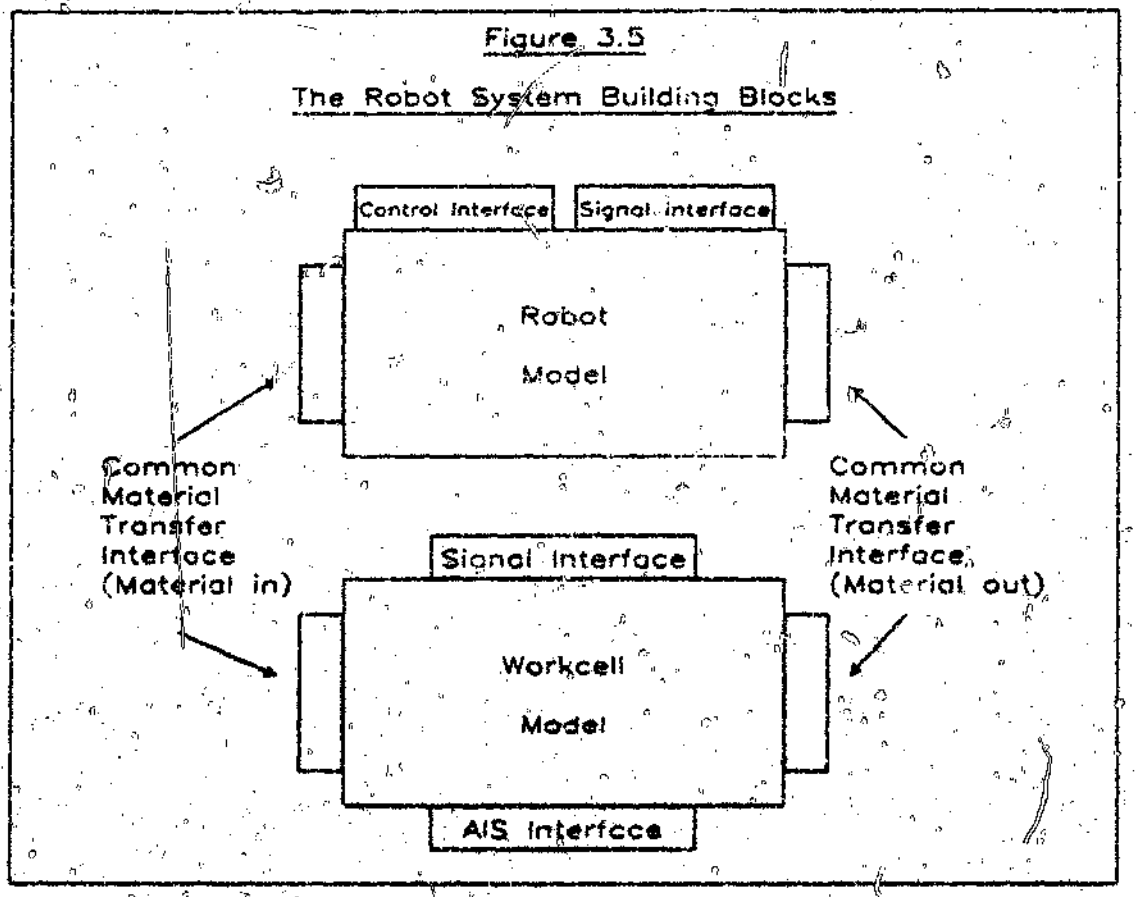
(4) developing and testing the control and information system software.

Computers, either mini or micro are interfaced with models to send and receive control signals and commands and handle information about the system operation. Computer control applied to the system may be simpler than in an actual facility, however, the principles of control sensing, logic and actuation are the same. Therefore an in-depth analysis into the control aspects of AMS may be undertaken (Nof, Meier and Deisenroth, 1980).

Physical simulators can complement but not replace analytic techniques or digital simulation. Its particular strength is in analysing general physical aspects of a production facility, ie: layout, dimension, sequences of operations etc. In

Figure 3.5

The Robot System Building Blocks



3.5.1 Robot System Interfaces.

The robot model includes a signal and control interface. The signal interface models the robot digital inputs and outputs. This simulates the robot's use of digital inputs to direct branching and sequence control. The digital output signals serve to implement control functions for activities of peripheral devices and other pieces of equipment which operate in conjunction with the robot, Eg: conveyors, part feeders, machine cycles, etc.

Programmable logic controllers are often connected to the robot digital inputs and outputs to augment the usually limited capabilities of robot controllers in this respect (Thomas, 1983b). The signal interface allows connection of the PLC (see 3.6) to the robot digital inputs and outputs and also to sensors and actuators within the workcell model or surrounding shop floor models such as conveyors, etc.

developing a physical simulator, engineers can try many alternative concepts and eliminate the least desirable ones. An operating model can prove or disprove certain assumptions and can lead to better design alternatives.

2.3.1.1 Control Software Development

Physical simulators can play an important role as aids in the development and testing of all control software for an AMS. There are two aspects to this application. First by developing and debugging control programs on a simulator, the danger of damaging a sophisticated system can be minimized. Just as important in this respect, is the fact that control programs can be developed and tested without having to wait until the actual facility is completed. Consequently, production operations can start much earlier and with fewer problems.

The objective of these simulators is to provide close contact with the operating system. Experimentation with the physical modules provides an excellent opportunity to exercise creative facility design ideas and computer control development. Furthermore, unlike abstract models the physical simulation retains the relevant physical properties of the analysed system, and can, therefore, be used to prepare and test actual control software before the actual system itself is installed (Nof, Meier and Deisenroth, 1980).

Compared to digital simulation, the major advantage of physical simulators is that they are visual, active and comprised of actual hardware.

The major disadvantages of physical simulators is their cost which may be in excess of $10,000 not including the cost of design and construction (Diesch and Malstrom, 1985). Further, physical simulators are relatively inflexible after construction (Feltner and Wiener, 1985).

2.3.2 Digital Simulation.

There is a large amount of literature available on the application of digital simulation in AMS design and evaluation. It is difficult to provide an in-depth overview of all these applications since many apply to specific AMS implementations. Also due to the variety of simulation languages used, it would

The control interface models robot communication with a host computer which forms part of the PAS operating system. This facilitates uploading and downloading of robot programs between the host computer and the robot controller. Commands representing functions such as those found on teach pendants can be sent to the robot controller from the host computer. Further, outside sensory computers can transmit information for robot motion control via this interface, as is the case for current vision system applications (Thomas, 1983a).

Although we have not discussed the case of interfaces with tactile, voice and optical sensors, these sensors can be included in the robot system with the addition of an appropriate model and the necessary interfaces.

The robot workcell model includes an AIS interface for sensors placed within the workcell. The workcell may include jigs and fixtures and the necessary tooling for robot operation as a processing tool (see Appendix H).

3.6 The CNC Machine Building Block Model.

Machining stations in AMS incorporate CNC machines. These stations include equipment to perform the following tasks:

Automated part and tool changing.
Tool and workpiece holding in magazines.
Milling, Tapping, Boring, Reaming, Drilling, Turning, Grinding, etc.

A simulation of a machining station is performed using the building block models identified in this chapter along with CNC machine building block models. A CNC machine building block model is proposed as illustrated in Figure 3.6. A major requirement is the ability of the CNC controller to communicate with other intelligent devices over LANs to achieve integrated manufacturing operations and management decision support systems via the LAN (Morris, 1984). This communication requirement is modelled by the control interface and allows the downloading/uploading of machine programs as well as the transfer of commands, status and diagnostic information.

require a detailed knowledge of each language to understand the modeling approaches adopted. This is because the majority of models are described in terms of these languages. These are, therefore, not general models, but are dependent on the features of the simulation language in which they are written. We refer the reader to a number of papers.

The paper by Lenz (1983) describes MAST, which is a package for designing computerized metalworking factories. The Integrated Manufacturing Modelling System presented by Engelke, et-al (1985) provides an especially useful man-simulation interface. Rathmill, Greenwood and Houshmand (1983) describe the application of digital simulation in the design of the Scamp system in the U.K. This paper describes a one-off model which will require extensive modifications to analyse system changes. The more recent, currently available AMS simulation packages incorporating graphic animation are discussed by Grant and Wiener (1986).

Browne and Rathmull (1983) note that simulation models built to date have been built on large minicomputers using specialist simulation languages Eg: GASP IV, Q-GERT, SLAM, GPSS etc. In short they are cumbersome to use from a user and computer point of view. They are not really user friendly and in general only the model builder can use the model effectively. Models designed to date seem to be useful primarily in providing quantitive predictions of system performance parameters such as throughput time, number of machine tools and size of work in progress buffers. However, these characteristics change if the physical layout changes and thus a complete re-modelling effort is necessary to accommodate the alternative layout (Barash, 1986).

2.4 Problems With Current AMS Simulation Techniques.

In order to evaluate the general requirements for a simulation system, uses and problems found in current AMS simulations are presented.

(1) AMS systems are becoming more complex and the effort involved in simulating a system to any level of detail is increasing rapidly. Larger sections of code must be written and to handle complexity, advanced software engineering methods are necessary. The problem is that, based on available literature, no such methods are in widespread use.

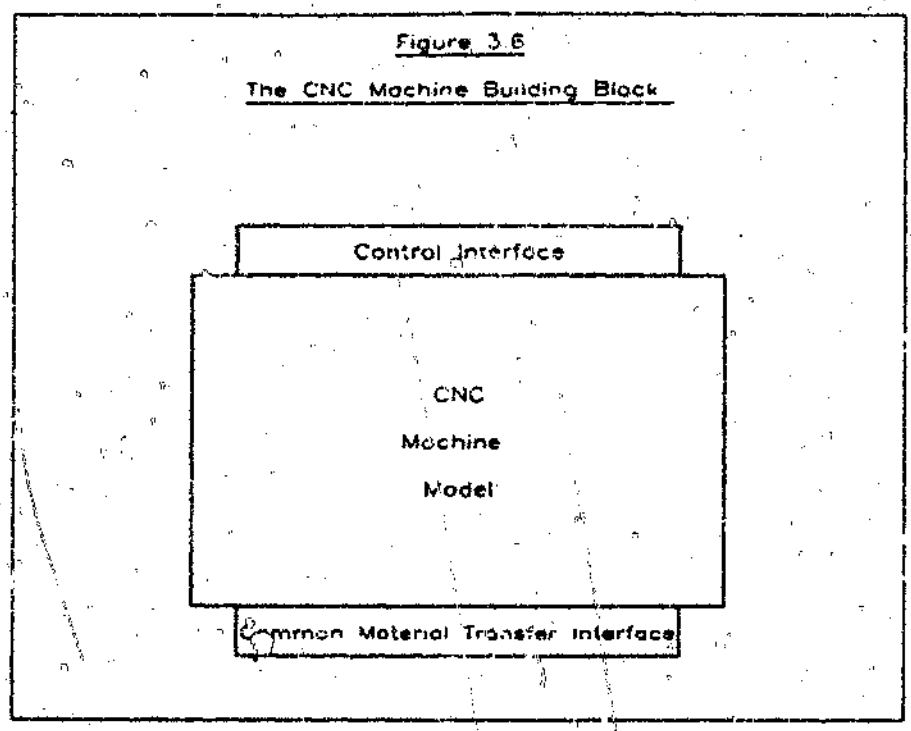


Figure 3.6

The CNC Machine Building Block

The level of simulation detail used for CNC machines ranges from the capability to simulate the execution of part programs (for test purposes) to machining or manufacturing time distributions for a machining station. This report does not cover the CNC machine in enough detail to determine the simulation model requirements.

3.7 The Equipment Controller Building Block Model.

We define a controller model which is responsible for translating PAS operating system commands or primitives into a sequence of signals which must be sent or monitored for the low level shop floor device control. This model represents a computing device which generates and monitors signals. The real-world PAS physical system analogy could be a programmable controller (PLC) or a personal computer (PC) with the proper interface. The model is shown in Figure 3.7. The controller model is completely defined in terms of the control and signal interfaces. The controller model is used to control conveyor system building block models, the AS/RS and to interface with sensors and actuators on the shop floor.

(2) Current AMS simulations are inflexible. Many simulation applications apply to specific problems in a given AMS. Analytic tools are used to evaluate an initial design and then, based on this, a detailed or "emulation" model is studied. Kay and Walmsley (1982) note that "emulation" models are inevitably customised to individual systems. General simulation languages are used and the problem is described in terms of the code. Problems arise when the system must be updated to accommodate new technology or the addition of new part types. This means that the initial modelling effort cannot be used since the necessary changes cannot be accommodated easily. Thus the modelling effort must be repeated at great effort and cost.

(3) Real-Time Simulation: Current simulations do not run in real time although there are simulation languages available with a concurrent base language.

(4) Differing Levels of Simulation Detail: Three levels of simulation are used in AMS. These are:

a) Emulation models for robotic cell layout and detailed simulation.

b) Analytic or simulation techniques for providing an aggregate view of system operation.

c) Dynamic simulation which shows the general operation of each unit in the manufacturing cell, ie: movement of parts and transporters and whether machine tools are occupied or not.

There are benefits obtained by using an emulation model since these are used for the detailed design of robotic manufacturing cells including robot program generation. In this case it is necessary to know how each unit in the cell interacts. In the case of dynamic models, a view of each unit in the AMS can be obtained along with a view of the system operation and hence its overall performance.

The reader is referred to Mills (1983) for a discussion of the various levels of simulation modelling and to Rathmill, Greenwood and Houshmand (1983) for a description of the application of the various simulation modelling levels in the design of the U.K. 600 Group's SCAMP system.

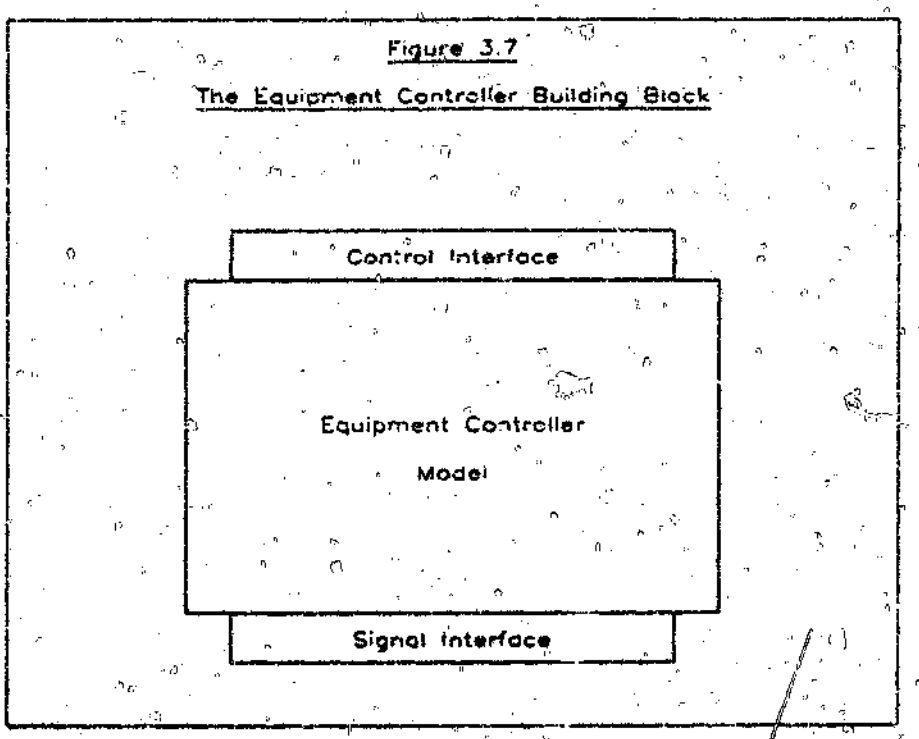


Figure 3.7

The Equipment Controller Building Block

3.8 Simulating the AMS Using Building Block Models.

This section briefly describes how a simulation of an AMS will be performed using the building blocks defined in this chapter.

In order to simulate an AMS, an initial plan or existing layout of the shop floor equipment will be required. Data pertaining to the characteristics of the shop floor devices will also be necessary, for example, conveyor speed, S/R machine speed, etc.

The user will use the interactive simulation system editor and will enter in the data describing the AMS layout as well as the characteristics needed to define the operational behaviour of the equipment in the AMS. The user will also specify the material transfer interfaces, that is, the equipment unit interfaces for the simulation of material transfer. This will allow the flow of material between models to be specified and hence the flow of material over the AMS layout to be defined. The AMS conveyor system will be defined using the building block models listed

(5) The use of queuing techniques and probability theory of part generation and processing time: These approaches aggregate the operation of the AMS and the results apply to steady state operation only. As any plant manager will state, no plant ever reaches steady state since new orders are processed. w goals must be met and scheduling is on-line, real-time (Feltner and W er, 1985). The results of these methods are therefore in doubt. Also the arrival and processing times of parts are not stochastic but deterministic and based on demand. The techniques of probability theory, however, are not invalidated by this statement but can be used for simulating breakdowns etc. which can be modelled stochastically and can be treated on an aggregate basis.

(6) Many simulation languages are used for AMS simulation. Apart from requiring a modelling expert to build models, their construction is time consuming and unless the models are well designed they can be inflexible. Mills (1983) notes that one of the major problems of these languages is that the control and decision rules for the operation of the AMS are included in the model. In many cases the designer must then use these built-in facilities rather than using control rules which apply to the specific AMS. This aspect is very important because one should, in model building, differentiate between the system control and the behaviour of the shop floor equipment.

(7) Bloch (1986) notes that current simulation methodologies incorporate fixed control strategies but do not, at the same time allow for the automation of control functions which include process planning, scheduling and coping with contingencies. In the analysis of AMS design and operation the simulation should allow for the evaluation of control aspects for a given physical configuration and vice versa. This separation will allow evaluation of the optimum hardware configuration as well as allow the generation of optimized, modular, off-the-shelf software for system control.

(8) Current AMS optimization by simulation requires an iterative process of simulating a given system, evaluating the results, changing the simulation model and repeating the process. This is a time consuming method and relies heavily on the experience of the AMS designers.

in Section 3.1. Each conveyor section will be characterised in terms of conveyor length, speed and the points at which material transfers may be made. Sensors and actuators may be simulated using the equipment controller model.

In the case of the AGVS, the user will enter in the number of AGVs, their operational characteristics such as speed, material transfer mechanisms and paths of vehicle travel. The positions on the shop floor at which material transfer is possible will also be defined.

In the case of the AS/RS, the user will enter in the position of the storage rack and its physical dimensions. The speed of the S/R machine will also be defined along with a material transfer mechanism. The points at which material transfer occur will also be defined (pickup and deliver points).

A robot will be simulated using the robot system building block model. Parameters describing the motion of the robot end-effector will be required along with the coordinates of all the points on the shop floor within the robots envelope at which material transfer takes place.

A machining station will be simulated using the building block models defined in this chapter along with a CNC machine building block model. Generally, a machining station will include a CNC machine, a robot for workpiece loading/unloading and tool changing. A simple conveyor system will be used for workpiece or tool magazine buffering and input/output to and from the machining station. An AGV may be used to transfer workpieces and tool magazines to and from the machining station.

Once all the data for the completed AMS has been entered and verified by the simulation system, the simulation may proceed. The user will select the simulation executive from where the simulation will be initiated. The simulation will be set up for one of the real-time or faster than real-time simulation applications as defined in Section 1.4. The simulation will execute accordingly with graphical results being displayed.

3.9 Chapter Summary.

This chapter has introduced the logical system building block models which can be interconnected by the user to form a simulation of the PAS physical system (ie:

2.5 Requirements For AMS Simulation.

The above section shows the shortcomings of current simulation methods. Based on the points raised, a number of features that should be included in any simulation system are enumerated.

(1) Evaluation of Control Software. This is the ability to design, test and optimize control software before system commissioning. This includes the complete manufacturing computer system hierarchy. This aids in the integration of the entire manufacturing process. Lenz (1983) notes that a simulation that aids manufacturing system design and control must be able to simulate systems and let the designer study alternative hardware and software configurations.

(2) Physical System Configuration. The simulator should allow the easy insertion and editing of the configuration of different shop floor machinery layouts and its optimization.

(3) The simulation should be at such a level of detail that it is possible to understand or gain insight into the operation of at each level in the system hierarchy. This will facilitate the design and construction of modular AMSs which can easily be adapted to the requirements of changing technology and the need to add new part types into the system menu.

(4) Another useful aspect of a simulator would be the capability to use expert systems to analyse the stages and outcomes of simulation runs. Using its knowledge base the system could suggest the path to AMS optimization.

(5) The use of feedback for real-time monitoring and control of the AMS. This will allow alternative scheduling rules to be evaluated before the system is subjected to them. Also if the models represent the physical system sufficiently accurately, then the simulator will contain all the information that the physical system should contain. Thus this information can be compared to facilitate on-line fault analysis.

(6) The simulation should allow a customer order to be fully checked, thus allowing a quick quotation and examination of all the requirements necessary to accept the order. This can maximize the efficiency of management aspects and smooth out the production run.

to form the logical system). The control and AIS interfaces support model communication with the PAS operating system via a LAN. The signal interfaces allow controller models to be connected to control other building block models and their associated peripheral devices on the shop floor where appropriate. The material transfer interface allows the transfer of logical material between the building block models, which comprise the logical system. Each building block model represents an entity which must be modelled to simulate the AMS operation in real-time or faster than real-time.

A short description of the steps required in simulating an AMS have been described. Although this description is a look into the future and extensive detail is lacking in the description, it provides insight into the ease of constructing an AMS simulation due to the modular model approach.

This completes the first stage of the modular modelling approach. The next stage is the specification of the model interfaces. This stage is discussed in Chapters 4 and 5. Chapter 6 discusses the requirements for constructing models so that they may simulate the physical system operation in real-time or faster than real-time.

(7) The simulator should also be able to examine the AMS operation under all operating circumstances (a valuable aspect of computer simulation) including coping with machine breakdown, communication failures and the loss of parts on the shop floor etc.

(8) One of the most important aspects of the simulator is the man-machine interface. This aspect of a simulator determines how the users will react and use the simulator thus determining its success as a design and operation aid in AMS analysis. Useful points to note are the ability to construct AMS systems using graphical symbols, animated colour displays of the system operation and the presentation of results in a simple and easy to understand manner. The user should be able to base his decisions on the information which is immediately available, rather than on printouts of statistical data.

This is noted by Mills (1987) who notes that simulation, particularly when using colour graphics, aids communication and helps build confidence in a project by allowing potential problems to be recognized and the appropriate corrective actions taken. This allows management to quantify the risks involved, the project team to understand the system constraints and the operating personnel to see the system's functional capabilities.

2.6 Chapter Summary.

The points stated in the previous section are deemed to constitute the essential features of any AMS simulation. In general, the objectives of such aspects are a better conceptualization of the various production problems which AMS are required to solve (both physical transformation and management aspects) and hence to increase the operating flexibility of the hardware and software modules of the system as a whole (Brandolese and Garetti, 1983).

To achieve these aims this project report has been written to lay the foundation for the development of a simulation system incorporating the essential features outlined here.

4 DEVELOPMENT OF A COMMON MATERIAL TRANSFER INTERFACE

The reason for adopting a modular simulation approach is the development of models which can be arbitrary interconnected or interfaced to other models. This is essential to provide the flexibility needed in simulating advanced manufacturing systems. This allows the simulation to cope with a variety of manufacturing system configurations and ease of changing the plant layout.

Since we are modelling an automated manufacturing system which consists of the interconnection of the manufacturing equipment under discussion, it is necessary to provide the same interface possibilities for each model, as exists in the manufacturing system itself. Each model must, therefore, have a material interface for the passage of material since, at physical level, this is how the shop floor machinery is interconnected. The interface must be common to all models so that arbitrary models can be interconnected as long as the interconnection is viable. This section describes the derivation of such an interface called the common material interface. The interface models the interconnection of the shop floor equipment under discussion.

We first discuss the model view of material transfer and then define material. Based on this we discuss the derivation of a common material transfer interface.

4.1 Model View Of Material Transfer.

The transfer of material between shop floor devices or equipment units is usually associated with some form of mechanical motion or operation, Eg: the robot gripper releases or grips material, material arrives at the extreme point of a conveyor or the shuttle mechanism of the AS/RS transfers a load at the extreme point of motion. At a physical level the devices are not aware of material transfer. They have no way of knowing whether material has been received or delivered. At an informational level, however, sensors are placed at strategically chosen positions so that it is possible to identify whether material has been transferred.

3 SIMULATION SYSTEM BUILDING BLOCKS

In this chapter the logical system building blocks are defined. The building blocks are defined in accordance with the shop floor devices or systems used in the AMS. The aim is to provide building blocks which can be interconnected according to the possible interconnections that exist in the AMS physical system. In this way the simulation system user can construct a logical system suited to his particular requirements and according to the shop floor layout.

Each building block constitutes a separate model or logical process (LP) which must be modelled. These models and their interfaces provide the external view of the logical system. That is, the view of the logical system that the user sees. In this chapter we identify the building block models for each shop floor device or system under discussion as a first step of the modular modelling approach. However, to facilitate our discussion, we briefly describe the general building block model interfaces. A short description of how the building blocks will be used in constructing a simulation of an AMS is presented.

3.1 Building Block Model Interfaces.

There are four general interfaces that define the building block models which are visible to the user. These are:

1) The Control Interfaces.
2) The Automatic Identification System (AIS) Interface.
3) The Common Material Transfer Interface.
4) The Signal Interface.

The control interface allows the interconnection of models with the PAS operating system for the transfer of commands or primitives, shop floor device status information and program uploading/downloading (see Section 5.1).

The AIS interface models the transfer of information from intelligent sensors within the shop floor device or system building block models to the PAS operating system (see Section 5.3).

The common material transfer interface models the transfer of material between building block models (see Chapter 4).

In this project report, models are constructed at a physical level where the simulation system user has the option of placing sensors as required. Since the models are constructed from a physical viewpoint, they are not aware of material transfer and this fact must be incorporated in the models. To achieve this each model has four functions available which define how material is transferred between models. These are called material transfer functions and are made up of the following:

1) Send_Material
2) Receive_Material
3) Get_Material
4) Material_Ready_For_Transfer

These functions define the procedure whereby material is transferred between models. The functions are called at a specific point of simulated motion or operation where material is transferred irrespective of material availability for transfer.

The function Material_Ready_For_Transfer makes material available in one model so that it may be transferred when the other model uses the function Get_Material (see Section 4.5.1).

In summary, the above functions define how models transfer material. In order to support this model view of material transfer an underlying material transfer interface is necessary. This interface models the physical interconnection for material transfer between shop floor devices and systems. The interface is described in terms of a material transfer protocol (see Section 4.4).

4.2 Model Material Inputs And Outputs.

As a first step in developing the common material interface it is necessary to define and model material used in the logical system. This section presents a definition of the material and then discusses the operations which the material is subjected to, to achieve the manufacturing system's objective. We then show that the given definition is suitable for representing the processing operation of each model and hence, are suitable to form the inputs and outputs of each model.

The signal interface models the interconnection of shop floor controller models such as programmable logic controllers (PLC) or personal computers (PC) (see Section 3.7) to building block models for their function control as well as the interface to sensors and actuators on the shop floor (see Section 5.2).

These interfaces define the building block models. We note that since each building block model is a logical process (see Section 1.5.5), the interfaces are all message passing interfaces and support communication over a LAN. The following shop floor devices are discussed with respect to presenting the AMS simulation building blocks:

1) Conveyor System.
2) Automated Guided Vehicle System.
3) Automated Storage and Retrieval System.
4) Robots.
5) CNC Machines.

3.2 Conveyor System Building Block Models.

The conveyor system configuration will be tailored towards the particular tasks the system is to perform in the AMS. The "unit" functions performed by the conveyor system will depend on the overall conveyor system operation required and its layout on the shop floor. In order to provide for all possible layouts and operations, it is necessary to provide modular building block conveyor models from which the user can form the desired conveyor system or systems. The reader is referred to Muller (1983) for a brief background of conveyor systems. We discuss the various conveyor system building block models.

3.2.1 Conveyor System Models.

We limit our discussion to the particular case of powered roller or chain type conveyor systems.

Flexibility in building up a conveyor system is provided by using modular conveyor building block models. These models are identified by decomposing the conveyor system operations into atomic functions which will then constitute a conveyor system building block model.

4.2.1 Material Definition.

In the simulation of the automated factory we define material to consist of parts, tools and pallets. The role of each is discussed.

Parts: A part is a unit of production from the raw material input to the finished product. Parts undergo assembly, fabrication and material handling operations. A part may be described in terms of the fabrication operations it has been subjected to and the collection of parts which have gone into its assembly. Parts may also be subjected to transport operations, palletizing and locating operations which all fall under the collective name of material handling operations.

Tools: A tool is a means of performing operations on parts. Tools may only be subjected to material handling operations.

Pallets: Pallets in the simulation are a general means for transporting parts, tools and other pallets. A pallet may therefore only be subjected to material handling operations. In this work a pallet is a collective name for items such as tote boxes, cartons etc. or container which is used to transport material from one location to another.

We have defined, in general terms, the material that the simulation system will be working with to achieve the production objectives. A more detailed description of the structures used to represent material in the simulation is discussed in Appendix A.

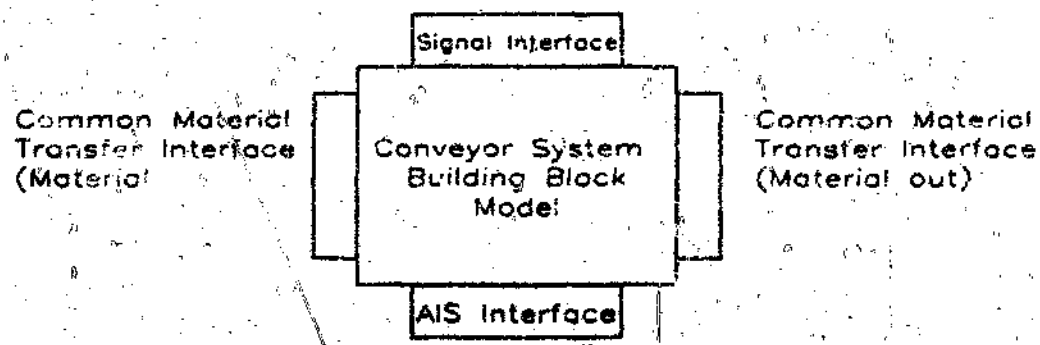
4.2.2 Representing Manufacturing Processing Operations On Material.

The automated manufacturing system performs many types of processing operations on material to achieve production objectives. These operations, called production processes, can be broken down into two major types of operations. These are:

1) material handling operations and
2) fabrication operations.

Figure 3.1

The Generalised Conveyor System Building Block Model



A summary of the operations of the conveyor system in its role of the material transporter, distributor and collector as well as part feeder, buffer, decoupling unit and interface to the transport and/or storage systems etc., identify the necessary building block models. These are summarised in Table 3.1. and are representative of some of the conveyor system functions used in advanced manufacturing systems.

The intention of the building block models described in Table 3.1, is to allow any combination of such models to simulate the operation of conveyor systems. Each model can be described as incorporating a common material transfer interface, an AIS interface and a signal interface for its function control. The generalised conveyor system building block model is illustrated in Figure 3.1.

The primary processing operations used in fabricating material are forming, machining, assembly and finishing. These operations are further enumerated in Table 4.1.

According to the material descriptions and representations (shown in Appendix A) it is possible to associate material with the processing operations it has been subjected to. For example, transport operations change the location of material on the shop floor. An assembled part is described by a list of its constituent parts and fabrication operations are associated with the part according to a list of operations it has undergone.

In concept, the list of parts which constitutes an assembled unit represents the bill of materials for the assembled unit. The list of fabrication operations represents a route sheet (Tompkins and White, 1984). It is therefore possible to determine where a part is, or what the part is, after it has undergone processing operations.

A point of note is that by superimposing a bill of materials and a route sheet, one obtains an operation process chart (Tompkins and White, 1984). This chart gives an overview of the flow within a facility. Assume that such a chart is constructed from the assembly and fabrication lists associated with a part after it has been "produced" by the simulation and timing information is also added to the chart. In this case a large amount of information may be gathered. This information would allow each part to be checked to ensure it has undergone the necessary processing operations in its production. The sequencing of each process may also be checked which is an indication of the flexibility of the manufacturing system and the ability of the control system to adapt to changing circumstances. Further analysis can yield the production rate, machining times and transport times, etc.

3.3 AGVS Building Block Models.

The discussion on the AGVS focuses on the particular case of inductively guided automated vehicles. The AGVS is decomposed into a number of building block models which facilitate the construction of the AGVS. A building block model is required to model a truck on the shop floor.

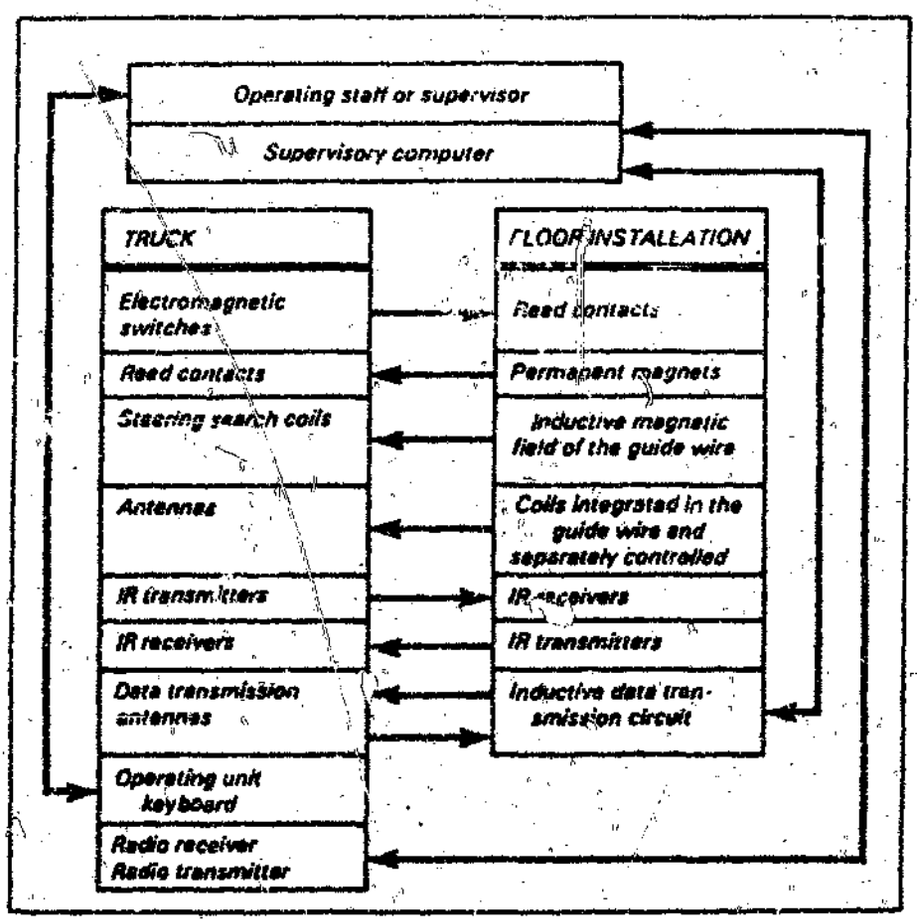


Figure 3.2

Data Transfer Between the AGVS Supervisory Computer, Trucks and Floor Installation.

(Reproduced From Muller, 19[illegible])

The AGVS will be made up of a number of truck models corresponding to the number of trucks actually used in the physical system. A further model is necessary which represents the AGVS floor installation and is accessible by each truck model. The floor installation model maintains a record of the paths of truck travel over

The primary processing operations used in fabricating material are forming, machining, assembly and finishing. These operations are further enumerated in Table 4.1.

According to the material descriptions and representations (shown in Appendix A) it is possible to associate material with the processing operations it has been subjected to. For example, transport operations change the location of material on the shop floor. An assembled part is described by a list of its constituent parts and fabrication operations are associated with the part according to a list of operations it has undergone.

In concept, the list of parts which constitutes an assembled unit represents the bill of materials for the assembled unit. The list of fabrication operations represents a route sheet (Tompkins and White, 1984). It is therefore possible to determine where a part is, or what the part is, after it has undergone processing operations.

A point of note is that by superimposing a bill of material and a route sheet, one obtains an operation process chart (Tompkins and White, 1984). This chart gives an overview of the flow within a facility. Assume that such a chart is constructed from the assembly and fabrication lists associated with a part after it has been "produced" by the simulation and timing information is also added to the chart. In this case a large amount of information may be gathered. This information would allow each part to be checked to ensure it has undergone the necessary processing operations in its production. The sequencing of each process may also be checked which is an indication of the flexibility of the manufacturing system and the ability of the control system to adapt to changing circumstances. Further analysis can yield the production rate, machining times and transport times, etc.

the shop floor and the placement of sensors and actuators within the shop floor installation. Data can be transferred between the floor installation and the AGVS supervisory computer (which forms part of the PAS operating system) and between the floor installation and the trucks.

The interaction of the floor installation with the trucks and supervisory computer is shown in Figure 3.2. Based on this, we provide each truck model with an interface to the floor installation model. The floor installation communicates with the supervisory computer and this is modelled by a control interface. Trucks receive commands or primitives and send status information to the PAS operating system via radio receivers and transmitters. This interface is modelled by a control interface. The AGV truck and floor installation models are illustrated in Figure 3.3. See Appendix G. for further discussion of the AGVS models.

Figure 3.3

The AGVS Building Blocks

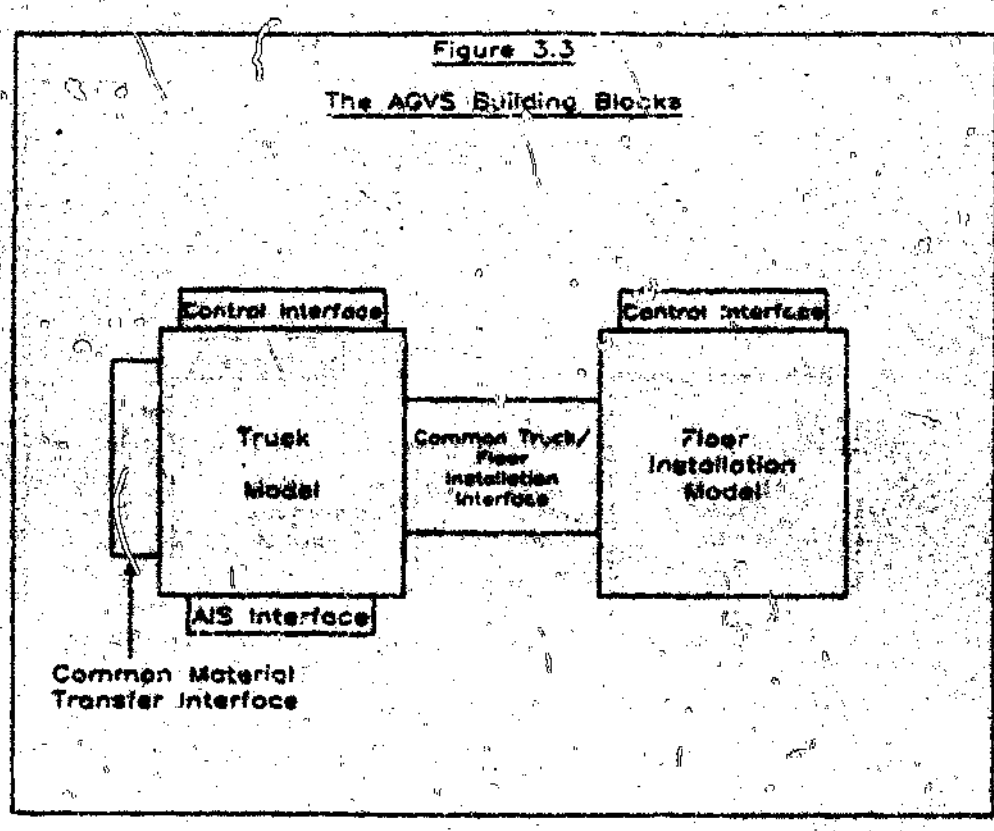


TABLE 4.1

Table of Manufacturing Processes

(Compiled from Eary and Johnson, 1962 and Carson, Bolz and Young, 1972)

Fabrication Operations:

Machining Processes :

Turning	Grinding
Shaping	Drilling
Milling	

Forming Processes :

Casting	Spinning
Embossing	Stamping
Forging	

Assembly Processes :

Welding	Insertion
Part	Riveting

Finishing Processes :

Cleaning	Buffing
Heat Treatment	Painting
Deburring	Polishing

Material Handling Operations :

Transportation	Locating
Palletizing	Load/Unload
Lifting	Operations

4.2.3 Summary.

This section has defined a model or representation of material which is processed by the simulated manufacturing system. Using this representation it is possible to use the material defined here as the general inputs and outputs of each equipment model. This is because, by comparison of the inputs and

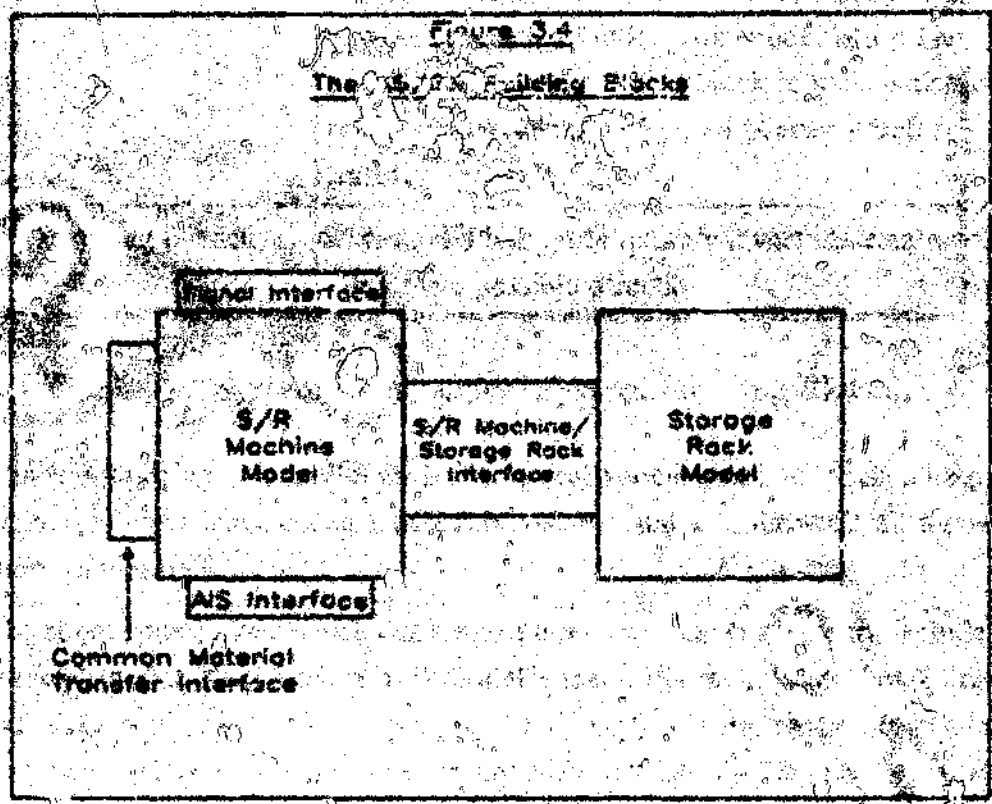
3.4 Automated Storage And Retrieval System Building Block Models.

The AS/RS is made up of two building block models. The first is the storage and retrieval (S/R) machine model which is responsible for fetching and depositing material at the pickup and deposit stations and for storing or retrieving material at the required location (slot) in the storage rack. The second is the storage rack structure model which models the physical layout and hence capacity of the AS/RS.

The AS/RS model is illustrated in Figure 3.4. The S/R machine is controlled by a PLC or PC model (see Section 3.7) via the signal interface. The S/R machine/storage rack structure interface is required to implement the model and cater for the motion of the S/R machine according to the physical structure of the storage rack. See Appendix F. for further discussion of the AS/RS models.

Figure 3.4

The AS/RS Building Blocks



corresponding outputs of a model, the operation of the model can be determined. We can therefore say that the model operation, in terms of material processing operations, is completely specified by its material inputs and the corresponding material outputs.

One further aspect of these representations is that a large amount of information may be obtained from the assembly and fabrication lists pertaining to the operation of the simulated system. This completes the first stage in the definition of a common material interface.

4.3 Material Interface Classification.

In the second step in the construction of a common interface, it is necessary to compile a knowledge base whereupon a solid foundation for the construction of interfaces may be based. In the case of automated manufacturing equipment, it is necessary to derive or deduce the nature of the interface and what type of interfaces exist and are used in practice. This can be determined from the knowledge base. To this end Appendix B includes a discussion of the various interfaces in automated manufacturing systems for material transfer. One should note that the terms used in Appendix B are appropriate for the physical system and not the simulation system.

Based on the discussion presented in Appendix B, an interface classification showing the various possible equipment interconnections has been compiled. This is shown in Table 4.2. The nature of the interfaces are also noted.

The nature of the interface is described by one of three possibilities:

1) active-active interface.
2) Active-passive interface.
3) Passive-active interface.

Material transfer is from the first to the second.

This characteristic shows which equipment unit is responsible for material transfer. For example, in an active-active interface, an AGV with a conveyor top will transfer material to a conveyor, if both conveyors are operating in the required direction of motion. An active-passive interface is illustrated when an AGV with a lift/lower

top deposits material on a conveyor. The conveyor is not in motion. A passive-active interface occurs when a robot picks up material off a conveyor which is not in motion.

TABLE 4.2

Material Interface Classification

Interface Partners	Nature of Interface
Conveyor Interfaces	
Conveyor to Conveyor	Active-Active
Conveyor to Robot	Passive-Active
Conveyor to AGV	Active-Active, Passive-Active
Conveyor to AS/RS	Active-Active, Passive-Active
AGV Interfaces	
AGV to Conveyor	All Types
AGV to Robot	Passive-Active
AGV to AS/RS	All Types
AS/RS Interfaces	
AS/RS to Conveyor	Active-Active, Active-Passive
AS/RS to AGV	Active-Active, Active-Passive
Robot Interfaces	
Robot to Conveyor	Active-Passive
Robot to AGV	Active-Passive
Robot to CNC Machine	Active-Passive, Active-Active
CNC Machine Interfaces	
CNC to Robot	Active-Active, Passive-Active

Material transfer is from the first mentioned to the second

Active: implies that motion is required in material transfer.
Passive: implies that no motion is required in material transfer.

For example, a Robot-Conveyor interface will not be Active-Active since it will be extremely difficult to program a robot to locate material on a moving conveyor during material transfer operations. There are other types of interfaces not shown in Table 4.2. These are cases where equipment such as an AGV, for example, will interface with an input/output station which is a solid physical structure which can accept a load from the AGV. This load will remain in position until it is needed by a robot, for example. These types of interfaces always constitute a passive entity in any material transfer.

In summary, the knowledge base provides the foundation for the third and last stage in developing a common material interface. This last stage is the development of a material transfer protocol and a conceptual structure for supporting the protocol.

4.4 Material Transfer Protocols.

The actual transfer of material in the physical system is associated with a mechanical motion or operation. This motion or operation is associated solely with the model. From the model interface point of view, this mechanical motion indicates the models intention of material transfer. At the extreme point of this motion, coupled with certain conditions being fulfilled, material transfer can occur. Some necessary conditions for load transfer are:

1) The existence of a load.
2) The equipment participating in the interface are working together to complete load transfer.
3) The interface mechanisms are compatible.

In modelling these aspects of the interface, which includes the nature of the interface, we can state the following:

1) From a modelling point of view the mechanical motion or operation needed for load transfer in the physical system is simulated by the model and indicates the models intention of load transfer.
2) The interface itself only becomes operational when the simulated mechanical motion causes the load to be at an extreme point of this motion. For example, in the simulation the lift/lower arm of the AGV has lowered the load onto a conveyor and is about to release the load. This is the point where the interface becomes operational and a function defined in Section 4.1 can be used to initiate the actual transfer.
3) The interface is solely responsible for load transfer between models. Load transfer between models can only occur when conditions, pertaining to both models which constitute the interface, have been fulfilled.

If transfer cannot be completed, the interface must take appropriate action based on what would occur in the physical system. The interface is thus concerned with transferring material once it has been released by the AGV, for example, and transferring it to the model the AGV is interfaced with

The discussion above indicates the necessity of condition evaluation for load transfers to be successful. These conditions exist on both sides of the model interface. This suggests that in order to effect load transfer, each model must know that, at a specified location, an interface exists to another model. The model however, is not aware of who the partner is or how the partner operates. Also, information must be supplied to the interface by the model (or the interface must obtain information from the model) to facilitate condition evaluation.

In summary, we note the need for communication between models, handled solely by the interface and the need for certain conditions to be satisfied for load transfer. If these conditions are not fulfilled, the interface takes appropriate action. The interface, therefore, supports message passing between models, condition evaluation and taking appropriate action if necessary. The message passing, condition evaluation and fault action taken are specified in terms of a material transfer protocol.

The material transfer protocol specifies how messages are passed, what the conditions to be evaluated are, what information is needed to evaluate the conditions and what action must be taken if the conditions fail. The protocol ensures that the models are unaware of this since the models are only aware of the fact that they are attempting to deliver a load or have received a load using the material transfer functions defined in Section 4.1.

4.4.1 Condition Evaluator And Condition Failure Action.

In this section we describe what conditions must be met for load transfer and the actions taken if these conditions are not met. We present simple, general examples of the conditions that must be evaluated. In general, condition evaluation and the action in case of condition failure is specific for the partners in the interface.

The nature of the interface affects the conditions that must be evaluated. For example, in the case of an active-passive interface, the model knows it must transfer a load (indicated by simulating the activation of a mechanical motion or process which culminates in load transfer). The model thus indicates its intention to transfer a load. However, at the extreme point of transfer, where

the interface becomes active due to the model calling a material transfer function, a number of conditions for successful material transfer must be fulfilled. Examples of these conditions are as follows:

1) The model which is to receive the load cannot accept the load if the location where the load is to be placed is already occupied.

2) If the largest dimension of the load makes it too big to allow handling by the receiving model's handling mechanism.

3) The point at where the load is to be placed is not a point where the receiving model can accept the load.

If any one of the above conditions are not met, load transfer cannot take place. In this case, appropriate action must be taken. In the physical system, it would be difficult to determine exactly what would happen to the load. To simplify matters, unless the outcome can be predicted, the material is assumed lost to the simulation system. Note that the model is not informed of the loss, since the model does not always have a means of detecting this occurrence. This is based on the assumption that the majority of material handling operations will not incorporate optical and tactile sensing.

In summary, the conditions that must be fulfilled and the action taken if they are not, are specific for each of the various interfaces and are also dependent on load transfer direction. Therefore, for each interface shown in Table 4.2, a protocol specification must exist.

4.4.2 Message Passing Aspects Of The Protocol.

Information from each partner in the interface is necessary to facilitate condition evaluation and failure action. In most cases this information is inherent in the material being transferred. Information may be required on the operating state of the simulated equipment or transfer mechanisms. It is also necessary to communicate to other models, the intent of material transfer. To this end, model handshaking by message passing, is proposed.

4.4.3 Message Interpretation.

We discuss message passing for each interface type:

4.4.3.1 Active-Active Interface.

In this case the first model sends a request to the second for it to accept a load. All the necessary information for condition evaluation is passed along with the request. The second interface evaluates all the conditions and indicates to the first model if load transfer is successful or not. If the transfer is not successful the appropriate action is taken by the first model.

4.4.3.2 Active-Passive Interface.

The message passing process here is the same as the above case except that no checking done to ensure that the second model has actively participated in the transfer.

4.4.3.3 Passive-Active Interface.

In this case the active interface sends a request to the passive interface. This is a request for a load transfer from the passive side of the interface to the active side. If a load is present and ready for transfer, all necessary information and the load is sent to the passive side. The required conditions are checked. If transfer cannot occur then the active side takes appropriate action.

A full protocol specification for robot/conveyor, conveyor/conveyor and AS/RS/Conveyor interface types is shown in Appendix C.

4.4.4 Summary.

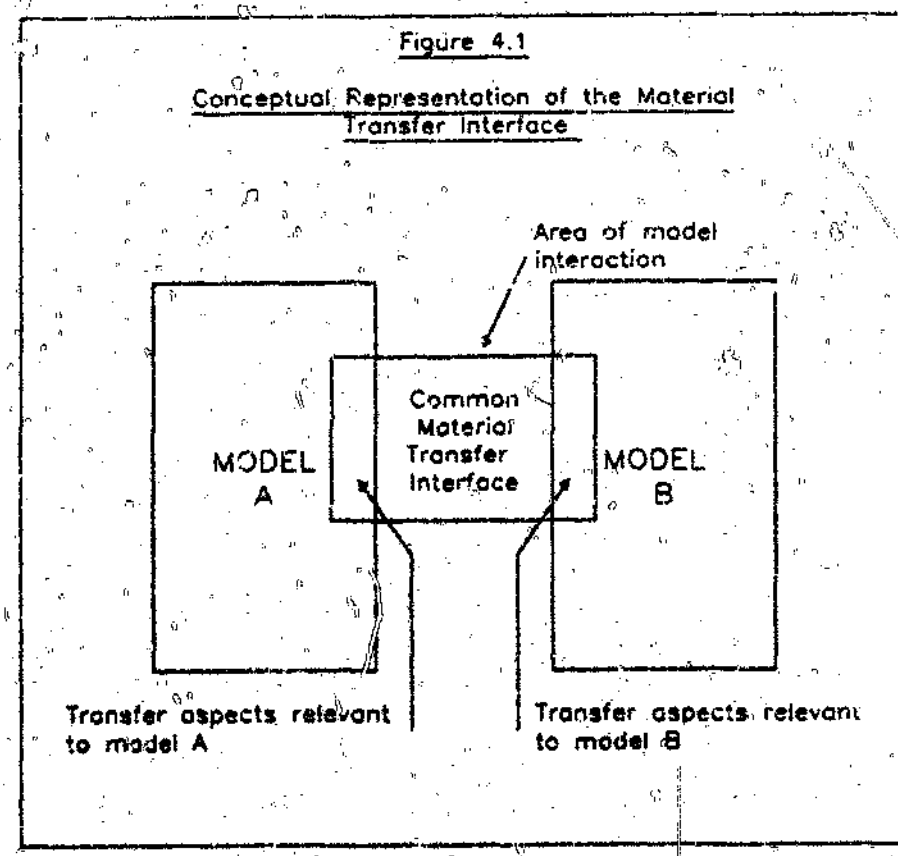
In this section we have defined a material transfer protocol. The protocol specifies conditions which must be fulfilled for load transfer and the action necessary if these conditions cannot be met. The protocol includes message passing, which defines how each side of the common interface communicates,

so that the nature of the physical interfaces is preserved in the modelled interfaces. The condition section of the protocol is specific to the partners that form the common interface, while the manner of message passing is specific to the nature of the interfaces.

This completes the stages necessary for the definition of a common material transfer interface. The ideas presented here can be conceptualized in the structure shown in Figure 4.1.

Figure 4.1

Conceptual Representation of the Material Transfer Interface



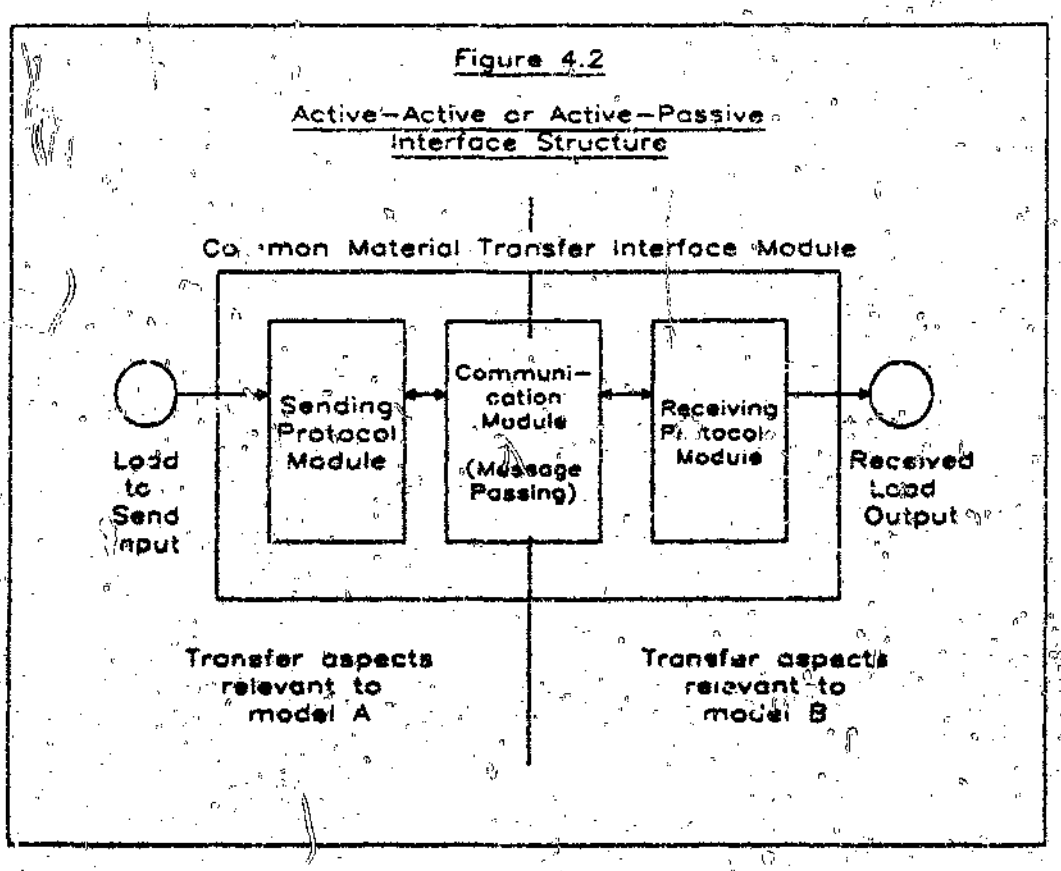
4.5 A Structure For Protocol Implementation.

In this section we present a structure which allows the implementation of the material transfer protocol. Two structures are described. One for active-active and active-passive interfaces and the second for passive-active interfaces. The ability to change protocols for different model interfaces, according to a given interface nature, is also supported.

The conceptual representation of the active-active or active-passive interface structure is shown in Figure 4.2. The message passing procedure, which is constant for these interfaces irrespective of the condition/action specification, is shown in flowchart format in Figures 4.3 and 4.4.

Figure 4.2

Active–Active or Active–Passive Interface Structure



The conceptual representation of the passive-active interface structure is shown in Figure 4.5. The message passing procedure, which is constant for these interfaces irrespective of the condition/action specification, is shown in flowchart format in Figures 4.6 to 4.7.

The necessary condition checking and action taken in case of condition failure is implemented from the condition/action specification section of the protocol. Each of the interfaces shown in Table 4.2 will have its own protocol for sending and receiving loads. The sending and receiving protocol modules are implemented from a protocol specification. Examples of protocol specifications are shown in Appendix C.

Figure 4.3

Flowchart for an Active–Active or Active–Passive Material Sending Protocol Module

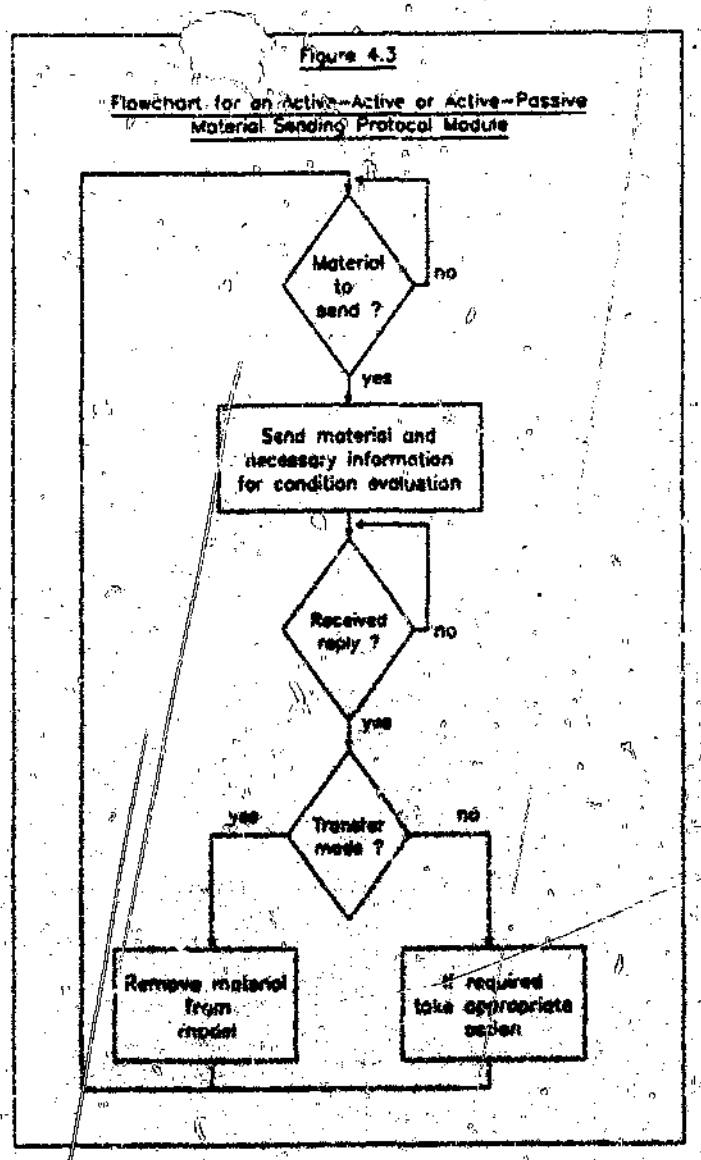


Figure 4.4

Flowchart for an Active–Active or Active–Passive Material Receiving Protocol Module

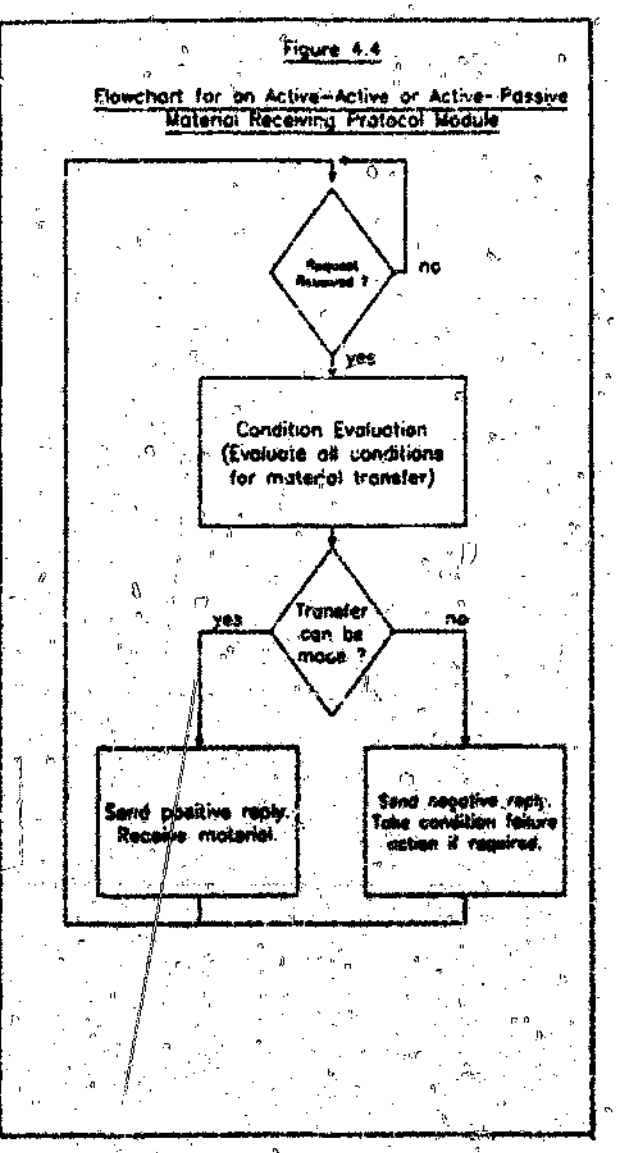


Figure 4.5

Passive—Active Interface Structure

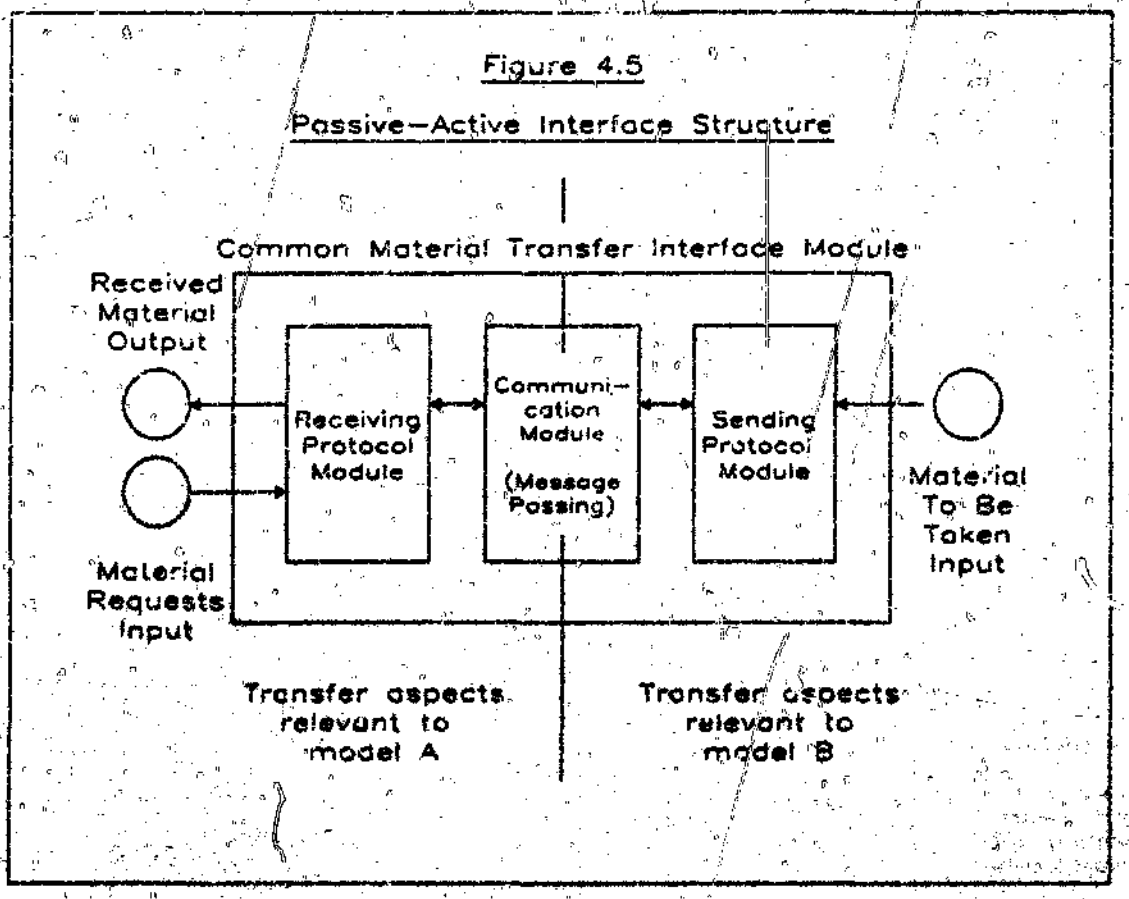


Figure 4.6

Flowchart for a Passive–Active Material Receiving Protocol Module

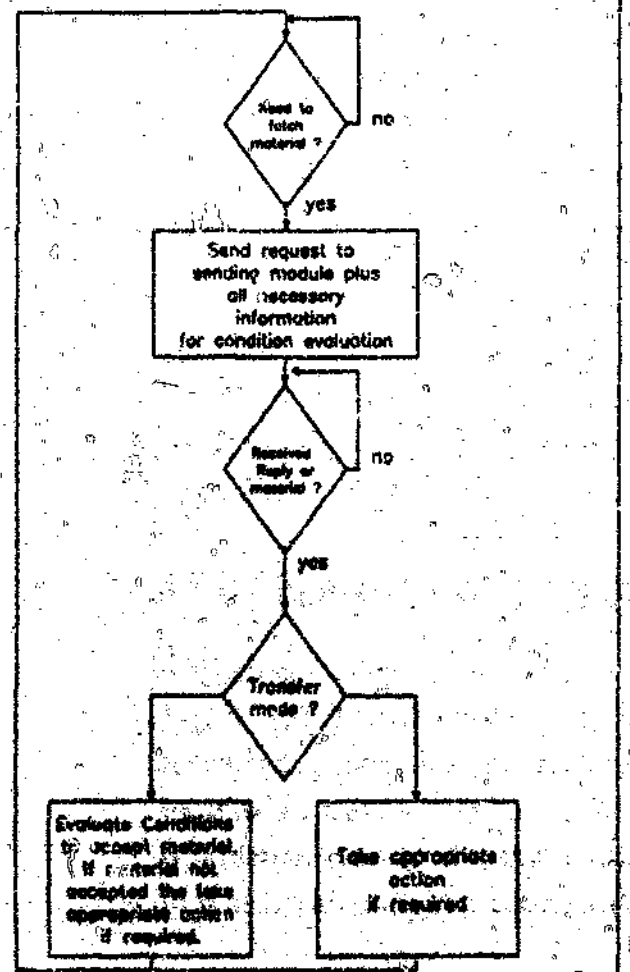
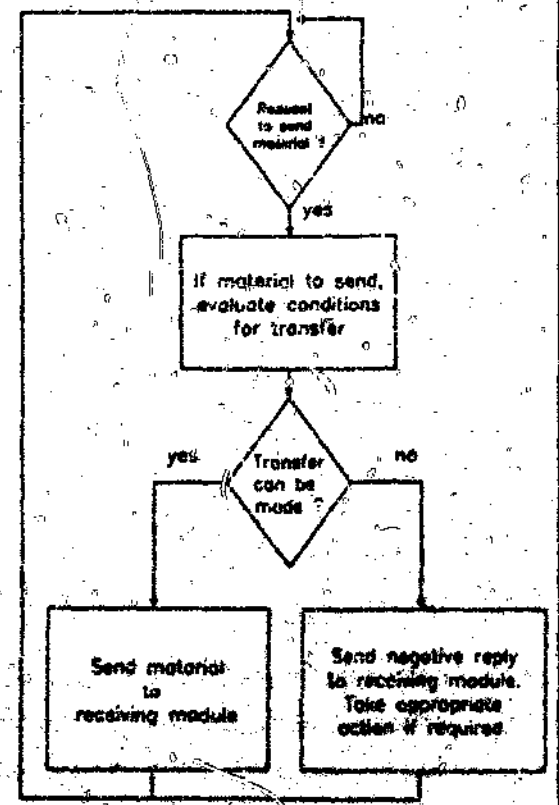


Figure 4.7

Flowchart for a Passive–Active Material Sending Protocol Module



4.5.1 Model Interaction With The Material Transfer Interface.

The models activate the material transfer interface module using the material transfer functions shown in Section 4.1. These functions are called at the appropriate point of simulated motion when logical material transfer is to occur. The operation of each function with respect to the common material transfer interface module is discussed.

4.5.1.1 Send_Material:

This function deposits material in the Load to Send Input shown in Figure 4.2. The interface module then takes care of the necessary operations for material transfer.

4.5.1.2 Receive_Material:

This function fetches material from the Received Load Output (see Figure 4.2). This function is associated with simulated motion which receives material. In the case where there is no motion associated with material arrival, Eg: a robot deposits material on a conveyor (the material is "received" by the conveyor), it is the responsibility of the receiving model to respond to material arrival.

4.5.1.3 Material_Ready_For_Transfer:

This function deposits material in the Material To Be Taken input (see Figure 4.3). This is usually performed when material requires a precise location for material transfer. When material has been given the precise location such as by a snap and clamp operation, for example, the material will remain available for transfer, in the Material To Be Taken Input, as long as it maintains its precise location.

4.5.1.4 Get_Material:

This function is used by a model to fetch material from another model. For example, a robot fetches material located on a conveyor. This function is the complement to function (3) above. If material has been made ready for transfer (by the Material_Ready_For_Transfer function) and the Get_Material function is called, material transfer will occur as long as all conditions needed are fulfilled.

4.6 Chapter Summary.

This chapter has discussed aspects of modelling the interfaces for material passage between models of the shop floor equipment. The model view hides all the necessary error checking for material transfer. To support the model view, a common material transfer interface module is required. The interface module operates according to a transfer protocol. The protocol specifies the conditions for material transfer and the actions necessary if transfer is not successful.

The definition of such an interface facilitates the arbitrary interconnection of models according to the interface classification shown in Table 4.2. From this classification, the general nature of the interface is observed. This characteristic is modelled in the simulation system material transfer interface. In proposing material transfer protocol specifications we have attempted to model the physical aspects of material transfer in the real-world to the greatest extent possible.

In the definition of material, we have not included the orientation of material but have supported the material's position as it moves around the simulated shop floor. The reason for this being that the models needed to support material orientation and the calculations involved would preclude the models from running in real-time.

On the point of condition failure at an interface, it should be noted that, from the point of view of the FAS operating system, the physical interface is supported by an information interface. Thus, based on sensors which detect the presence of material at the interface, the interface mechanisms will be activated. If there are errors in this information interface and hence incorrect activation, the simulation system will detect the errors by conditions at the material transfer interfaces not being met.

In conclusion, a common material transfer interface has been defined. The interface serves three major purposes:

1) The interface defines a standard which allows the arbitrary, yet consistent interconnection of the models of the shop floor equipment.

2) The interface forms part of the modelling process since the interface incorporates the physical properties inherent in the interconnection of the shop floor machinery.

3) The interface hides all aspects of material transfer not related to the model itself.

Note that the concept of Passive and Active material transfer is derived from Muller's (1983) classification of load transfer for driverless transport systems (see Figure B.1) and has been extended in this chapter for the shop floor devices under discussion. Further information is derived from the discussion by Groover and Wiginton (1986) on the mechanical interfacing of the various systems which constitute an AMS.

5 PAS OPERATING SYSTEM AND BUILDING BLOCK MODEL INTERFACES

In Chapter 3, the control, AIS and signal interfaces for building block models were identified. These interfaces provide the simulation system user with a flexible and consistent means of controlling the building block models. Further, these interfaces ensure that the PAS operating system cannot differentiate between the physical system and the logical system. This chapter describes these interfaces in more detail.

5.1 Control Interfaces

The control interface allows model communication with the PAS operating system. Based on the work presented by Bloch (1986), there are three aspects that the interface must support. These are:

1) PAS commands or primitives.
2) Equipment status.
3) Program downloading/uploading.

These aspects of the interface are discussed.

5.1.1 PAS Commands Or Primitives.

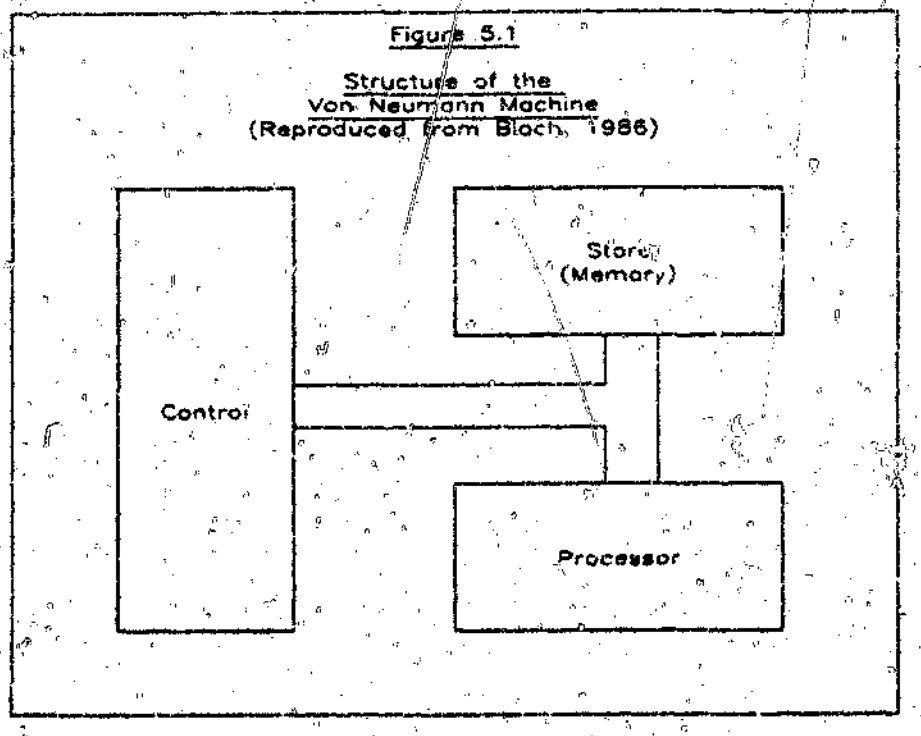
Bloch (1986) suggests that in order to maintain total automation system flexibility and to ensure that the system control is at least as flexible and as capable as the actual equipment on the shop floor, it is important that a command exists for every capability of the shop floor equipment. That is, a command or primitive, which is the smallest, discrete action that the shop floor equipment can be called upon to perform.

Based on Bloch's analogy of the PAS and computer (see Figures 1.1 and 5.1), the PAS primitives represent the machine code instructions of the central processing unit (CPU). These machine code instructions are fixed from a users point of view, but flexibility is obtained from the sequencing, ie: programming

of these atomic machine code instructions. The program execution is controlled by the PAS operating system which therefore orchestrates the operation of the shop floor equipment.

Figure 5.1

Structure of the Von Neumann Machine (Reproduced from Bloch, 1986)



Primitives can be connected together in sequence to form groupings known as "operations" (see Bloch, 1986). An operation is the set of primitives needed to direct a particular machine to perform a logically distinct job. These operations are used by the PAS operating system to perform a "process", ie: the total transformation performed on any input into the PAS.

Shop floor equipment units typically have between 10 and 30 primitives or commands, depending on their complexity. Also influencing the number of primitives is the sophistication of the equipment controllers, which then perform the low level equipment control.

5.1.2 Equipment Status.

The PAS operating system needs to maintain records of the current operational status of the shop floor equipment and information relating to the equipment or process on the shop floor, describing aspects of the process. The PAS

operating system accesses this information from the equipment controller and updates its internal data structures, which are used for real time decision making etc.

5.1.3 Program Downloading/Uploading.

One of the features of advanced manufacturing systems is the capability to produce a variety of products with virtually no time losses to change over from one product to the next. This permits the system to produce variable combinations and schedules of products. This capability is achieved by device controllers which can accept a program, corresponding to the particular product, that has been developed off-line and downloaded directly to the manufacturing equipment controller (Groover and Wiginton, 1986). The control interface must support this feature, especially in the case of robots or CNC (computer numerically controlled) machines.

In summary, we propose a model control interface which can support all of the above features and would be exactly the same as the real PAS control system/equipment controller interface. Appendix D shows some examples of the various primitives and status information requirements for a number of the equipment or building block models under discussion.

5.1.4 Standard Control Interfaces.

In order to standardize the control interface it is necessary to define all possible primitives and status information requirements for each shop floor equipment unit. Although such an interface may incorporate a certain amount of redundancy, it will allow total control flexibility from the point of view of the PAS operating system.

Some of the reasons that standard interfaces do not yet exist are due to the fact that no attempt has been made to dictate any limitations on the type of equipment that can be introduced into the automated manufacturing system (Bloch ,1986). A second reason is the proliferation of a number of unique solutions for different production situations where each solution is designed to measure.

Furthermore, primitives and status information may depend on the operations performed by the various types of shop floor devices. This is apparent in the case of robots and CNC machines. These machines are programmable and hence their operations may vary. Also, the interface may depend on the various equipment configurations possible. The equipment controller software may be dedicated to each individual application, thus the primitives may vary from the ones shown in Appendix D.

In order to standardize the control interface, a survey of all possible equipment types and processes in automated manufacturing systems is necessary. From this one can develop a set of all encompassing primitives and status information requirements for each shop floor equipment type. Since this is not within the scope of this work, we have described an interface in which the user may develop his own set of primitives which will form the standard for the particular application.

5.2 Signal Interfaces.

"Signal" interfaces allow the interconnection of the equipment controller models with the shop floor equipment or building block models. This must constitute an especially flexible interface since it must allow all the various equipment configurations possible to be taken into account. For example, the real-world analogy of this interface is the connection of motors, sensors and actuators to equipment controllers, ie: this interface models the wiring connection between the shop floor controller and the equipment.

From the point of view of the controller model a primitive or command is a macro-instruction and signals are micro- instructions. The controller model must translate the primitives into a sequence of signals being sent or monitored, for the low level control of the equipment models. 5.2.1 Conceptual Representation Of The Signal Interface.

The signal interface can be conceptualized as an array of signals, having the values of true or false. Whenever a state change of the signal occurs a message indicating this state change is sent between the controller and equipment models. For example, to start the conveyor in the forward direction, a message corresponding to a location within the signal array (the location identifies which motor to start)

is sent from the controller to the equipment model indicating a state change from false to true. When this signal is received by the equipment model the conveyor operation in the forward direction can be simulated. The interface has an array format so that a given signal may be associated with a specific sensor or actuator within the equipment model. The conceptual structure for this interface is shown in Figure 5.2.

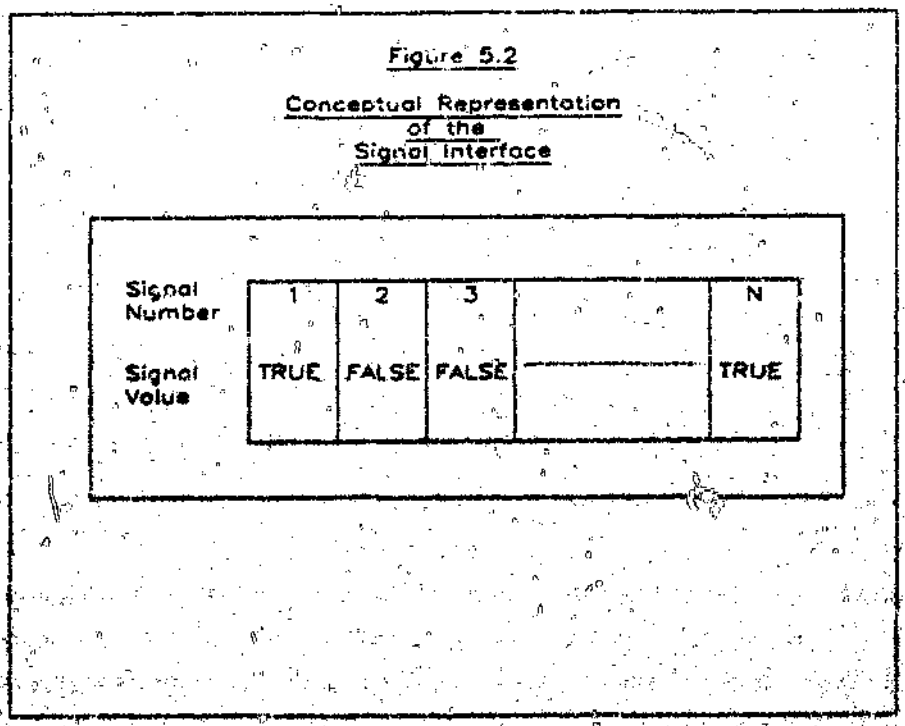


Figure 5.2

Conceptual Representation of the Signal Interface

In summary, the signal interface supports the transfer of messages, indicating a signal change of state, between the controller models and equipment or building block models, in both directions.

5.2.1 The Need For Signal Interfaces.

One may argue that the definition of a signal interface advocates an extremely low level approach to model construction. This may indeed be the case. An alternative approach to modelling the equipment models would be from the level of the PAS primitives. The problem is one of standardization of equipment

units and modular control software for all levels of control. Since these are not limited in any way, the approach of modelling from the primitive level, would limit the scope of application of the models.

Further, the structure proposed here, ie: the use of a signal interface, serves two purposes. As a research tool the resulting models would be extremely flexible in allowing all types of logical equipment configurations, which can then be analysed to provide suitable, modular, automated manufacturing system building blocks. Secondly, as a post-design tool, the controller model software will be available and must be tested. Using this approach this is possible.

The modelling approach adopted here is feasible as long as there is no standard for equipment units and control software. Once a standard has been developed to indicate how PAS cell controllers, for example, communicates with the shop floor equipment models, the flexibility provided by the signal interface will no longer be required. This will be due to the fact that equipment control will be completely specified in terms of primitives and equipment status.

One further argument against the use of a signal interface is the set-up effort needed to connect equipment models to controller models in the simulation system, as this is similar to wiring the shop floor equipment. This problem is offset by the fact that the automated system designers are forced not to take short cuts in running a system simulation. Secondly, the use of aids such as light pens and graphic tablets can be used effectively to minimize the simulation system configuration time.

5.3 Automatic Identification System Interfaces.

Tompkins and Wallace (1980) note that in the automated factory, the ability to know when and where the resources of production are needed is of paramount importance to achieve proper control. Detailed information on every product and component found in an exploded bill of materials as well as the tools, fixtures and other production resources must be captured and maintained on a real-time basis. This can be realized by the use of an automated identification system (AIS) with the appropriate sensors.

Since the simulation system is intended as a tool for analysing the control of the advanced manufacturing system, it is necessary to support the use of an AIS and associated sensors. To achieve this we define an "AIS" interface for each equipment or building block model. This interface forms the boundary between the model and the AIS central computer. It is assumed that the AIS computers will form part of the PAS operating system and all necessary information will be exchanged between the AIS computers and the PAS operating system control computers such as the PAS cell controller (e.g. Bloch, 1986), for example.

The interface should model the communication between the shop floor sensors and the AIS computers. There are a large variety of systems which range from bar code, radio frequency, surface acoustical wave, magnetic stripe and machine vision systems (Hill, 1985). We do not attempt to define an interface in this work but note that it is necessary to support the AIS in the simulation system.

Whenever material passes an automatic identification sensor in an equipment model, a message to the AIS computer is generated which contains such information as the unique material identity and the time and location where it is identified.

5.4 Chapter Summary.

In this chapter three manufacturing equipment interfaces for each equipment or building block model has been discussed. These include the control, signal and AIS interfaces.

The control interface allows the interconnection of models with the actual PAS operating system whether it is the PAS cell controller, the AGVS supervisory computer, the storage system control computer or the AIS computer. The interface supports all the features found in the interface between the PAS operating system and the real system.

The suggested identification interface models the use of automatic identification sensors. Information on material position, identification, etc., can be passed to the AIS computers so that it is possible to simulate feedback for more effective manufacturing system control.

The signal interface forms the boundary between models of the shop floor equipment (manufacturing system hardware) and their controllers (software aspects). The signal interface is structured so that all equipment types and their various configurations may be simulated. The array structure of the interface allows individual sensors and actuators to be addressed.

6 BUILDING BLOCK MODEL IMPLEMENTATION

At this stage of the modular modelling approach, the simulation system has been developed at the highest level of abstraction. That is, the building block models have been identified and their interfaces specified. We now proceed down a level of abstraction in order to "look" inside the models and describe how they are implemented.

There are five areas of concern needed to identify the internal modules of the building block models. These are:

1) Logical models.
2) Mathematical models.
3) A real-time simulation algorithm and faster-than-real-time simulation requirements.
4) Modelling equipment breakdown.
5) Graphical, animated display of building block model operation.

The above form the basis for implementing each building block model. Each is discussed in this chapter.

6.1 Logical Models.

In this work we define a logical model as a collection of parameters or attributes which characterise models. A simulation system user may then select some or all of these parameters together, specify their values and thus characterise the model for his particular use. The logical model provides a framework in which the user selects his view of the equipment or controller models. It is thus up to the user to ensure that his logical view matches the physical system.

Based on the discussion presented by Lennox, Thompson and Tshwele (1986), we present the requirements of the logical model. The logical model for each building block model contains a description of all the information which describes the model. This is compiled from a survey of all the shop floor devices which perform

the functions that the building block simulates. Examples are, possible velocities of the AGVS trucks, the various types of load transfer mechanisms possible for the trucks, etc.

The major function of the logical model is to guide the simulation system user in selection of parameters which ensures that the model will accurately simulate the operation of the physical equipment that the user is modelling. Again, using the AGVS as an example, the logical model framework will guide the user to define a specific truck speed if a constant velocity model is assumed. Also, the user will be asked to specify which load transfer mechanism the truck will incorporate. The user will be given all the options, Eg: lift/lower top, conveyor section or fork lift etc. The user continues through the framework until the specific model the user is interested in has been chosen or defined.

In this work we have not provided a comprehensive information base upon which a useful logical model framework may be constructed. However, Appendices E to I include examples of the parameters necessary for each equipment model type including a logical model for the controller building block models.

6.2 Mathematical Models.

In modelling the shop floor equipment units in real-time, it is necessary to ensure that the responses to model inputs are generated when required. That is, if a command or primitive is sent to a model and the corresponding actual physical unit at the same time, then the model should generate a response at the same time (or as close as possible) as the physical unit whether the response times are measured at the signal or control interface. This time condition, which is necessary to relate model inputs to model outputs, can be generated from mathematical models.

Specific types of models specified from the logical model may need different mathematical models to simulate the required behaviour. More exact models may be chosen under various circumstances and this may require extensions to the usual mathematical models in use. In general, the mathematical models required here, relate motion to time and are thus based on Newton's Laws of Motion. The

models may range from a simple time delay (for time invariant operations where the time of operation can be easily measured from the physical system) or linear motion models at constant velocities to models needed to describe robot dynamics.

Part of the function of a mathematical model is to incorporate the parameters of each logical model which allows us to distinguish between the basic operations of each equipment or building block model type.

6.3 Real-Time And Faster-Than-Real-Time Simulation.

This section presents the real-time simulation algorithm which forms the basis for building block model operation. More specifically, this is the generation of the correct sequence of output messages and their timing in order to model the physical processes (PPs) of the AMS. We first discuss the required model descriptions for event generation, the modelling constraints the algorithm is required to solve and then present the algorithm. The algorithm presented is based on the asynchronous simulation algorithm discussed by Misra (1984). The case of faster-than-real-time simulation is also discussed.

6.3.1 Model Description.

According to the concepts of distributed simulation (see 1.5.6) the simulation system building block models are called logical processes (LPs). These LPs model the operation of the corresponding PP on the shop floor by message passing. All building block model interfaces are message passing interfaces. From an external viewpoint, these messages occur at discrete instances in time and hence the LPs are discrete event models.

Each building block model is defined in terms of equipment states, model states and a model structure description. Each are considered:

6.3.1.1 Equipment States:

The equipment states defines the physical operation of the building block models. For example, the conveyor transport model has three states. These are On_Forward, On_Reverse and Stop. The equipment states are asso-

ciated with commands or signals sent from the PAS operating system or controller models respectively. For example, signal 1 implies start the conveyor in the forward direction. The conveyor transport model state changes to On_Forward on reception of this signal. The equipment states are dependant on the physical mechanisms and their associated operation.

6.3.1.2 Model States:

Apart from the controller building block model, all other models simulate material and/or mobile shop floor equipment motion over the shop floor. Therefore, the material and mobile equipment units change their location on the shop floor. The locations of simulated material and mobile equipment units at discrete instants in time specify the model state. The model state is a collection of all values needed to describe material or mobile equipment over time.

6.3.1.3 Model Structure Description:

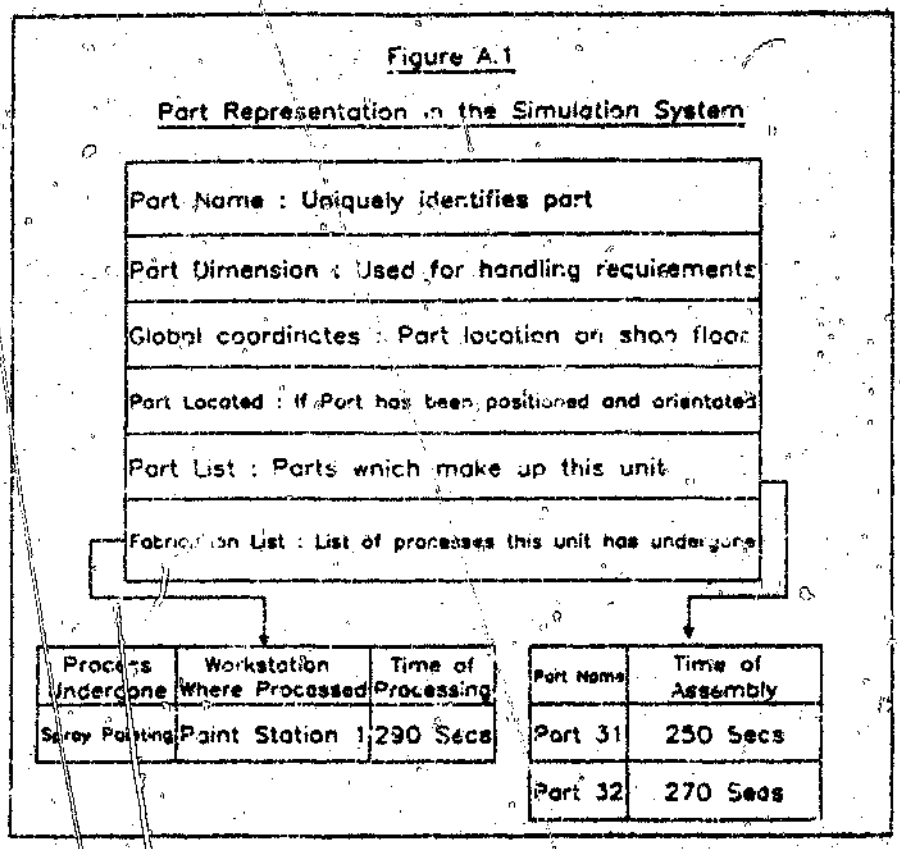
The PPs on the shop floor have a defined structure. For example, the placement of sensors on the conveyor transport model, the storage structure of the AS/RS, paths of travel for the AGVS trucks and sensor placement in the AGVS floor installation.

In the simulation this structure must be represented in a convenient manner. In this way material position on a conveyor relative to a sensor, for example, may be maintained. Further, the distance between the current AGVS truck position and a branch in the path of travel can be calculated. This structure can thus be used in conjunction with the model state to generate events. For example, the next event for the truck is when it reaches the branch point where a decision must be made as to which path to take.

The model structure description is compiled from the logical model of the building block. In this way the user may be asked to specify the AGVS floor installation, for example, which is recorded in the floor installation model (see Section 3.3).

Figure A.1

Part Representation in the Simulation System



By representing parts as described above, it is possible on perusal of the part and fabrication lists, to uniquely identify the part (along with the part name) and the processing it has been subjected to. Also, a comparison of a part representation as an input to a model and its corresponding output representation, uniquely identifies the processing operations of the model.

A.2 Tools.

Tools are a means of performing manufacturing processes. In programmable automation systems it is possible to program the shop floor equipment to produce various types of parts. This requires that machinery be capable of changing tools. Tools are thus necessary in the logical system to simulate tooling operations. These operations include tool transport to the correct machines, tool loading and tool changing, etc. A tool is also a unique way of identifying the fabrication operations

as an environment for the implementation and operation of an AS/RS demonstration model. Further, the tools available for software development make these computers ideal for this purpose on a small scale.

8.3 Selection Of A Software Development And Run-Time Environment.

Throughout this work the concept of modularity has been adopted. The advantages of this approach are apparent in the ease of partitioning the simulation system and for future maintenance, not only for the simulation system software, but also for the concepts presented in this project report. Therefore, the possibility of maintaining this modularity in terms of the simulation system software are of primary importance. For this reason, the programming environment for the Modula-2 programming language has been selected.

Modula-2 supports the concepts of modularity in that the language uses modules for the construction of software systems. Also supported is the ability to compile modules separately, thus allowing the collection of tested and verified modules in a software library.

In Modula-2, modules consist of two parts. These are a Definition Module and an Implementation Module. The definition module specifies the entities of the module that are available to the programmer. This essentially specifies the module interface. The implementation module includes the code that implements these entities. The implementation module may not be available to the programmer once the module has been verified. Using the module concepts of Modula-2, the modularity of systems and the concept of abstraction and information hiding are supported.

8.4 A Modula-2 Software Environment For Simulation.

There are a number of Modula-2 systems for IBM-PC or compatible systems. For the purpose of a demonstration model, we note that the model itself is made up of further modules. These modules have been developed by adopting a process viewpoint. That is, many of the modules which make up the model, run concurrently. We wish to support this in the simulation otherwise the module code must

The model states, the equipment states and the model structure description constitute the inputs to the real-time algorithm from which it is possible, in conjunction with inputs received, to generate the appropriate LP output messages.

6.3.2 Building Block Model Events, Messages And Event Generation.

Events are occurrences of sensor activations and material transfers. The actual occurrence of the event is manifested by associating a message with the event and the transmission of the message. An event is thus said to occur when a message corresponding to the eve , transmitted between LPs. In general, an event is a tuple (t,m) where t is the real-time of occurrence of the message m. In general, the message m may specify a sensor activation or material transfer. Messages will consist of a sent-time field and the message contents. Further fields may be added, such as the source and destination of the message.

Events are generated as a function of simulated material or mobile shop floor equipment motion. Since the mathematical models relate motion to time, it is possible to predict when a sensor will be activated by virtue of the equipment or material position relative to the sensor, for example. It is thus necessary to describe the building block model structure with respect to physical motion and sensor placement in a convenient manner (this constitutes the model structure description). This facilitates the generation of the next event based on the current material or mobile equipment unit locations. The ability to generate events forms the basis for the simulation of the equipment models in real-time.

6.3.3 The Real-Time Simulation Algorithm.

The algorithm proposed by Misra (1984) specifies that LPs operate asynchronously with respect to each other. Synchronization is achieved by incorporating a time field in messages passed between LPs. In this way some LPs may simulate ahead of others.

that a part is subjected to. For example, a spray painting tool at a robot work-station may cause a part fabrication list to include the entry: spray painted at work-station 2.

A tool may be subjected to material handling operations. A tool is represented by the structure shown in Figure A.2.

A.3 Pallets.

In the logical system, a pallet is the general name given to material containers and supports. Since it is not possible to include all the various types of containers and supports used in automated manufacturing, we generalise them in terms of pallets.

Therefore, a pallet may be thought of as representing tote boxes, bins, cartons, pallet boxes, skids, etc. Since we are not interested in the exact physical nature of these supports and containers, we characterise pallets as follows: In the logical system, a pallet is a means for transporting, storing and locating parts, tools and pallets. Note that a pallet may contain another pallet. A pallet is thus the support that is used to transport material and to structure the material it contains.

The pallet representation can support the unit load principle. A unit load can be defined as the unit to be moved or handled at one time. In some cases the unit load is one item of production. In other situations the unit load is several cartons (in the terminology given here this would be several other pallets) each containing items of production (Tompkins and White, 1984).

Pallet Operation As A Locating Element.

Engelberger (1983) notes that for robotised manufacturing systems, part position and orientation should be defined at the pick-up point of the robot. This must be arranged so that the robot will be able to pick out the parts one by one, in a sensible and repeatable gripper to part relationship.

In practice, palletization often provides a suitable solution. Parts can be arranged on a pallet in a grid, located on a spigot or between guides. The robot can be taught to pick up the parts until the pallet has been emptied.

as an environment for the implementation and operation of an AS/RS demonstration model. Further, the tools available for software development make these computers ideal for this purpose on a small scale.

8.3 Selection Of A Software Development And Run-Time Environment.

Throughout this work the concept of modularity has been adopted. The advantages of this approach are apparent in the ease of partitioning the simulation system and for future maintenance, not only for the simulation system software, but also for the concepts presented in this project report. Therefore, the possibility of maintaining this modularity in terms of the simulation system software are of primary importance. For this reason, the programming environment for the Modula-2 programming language has been selected.

Modula-2 supports the concepts of modularity in that the language uses modules for the construction of software systems. Also supported is the ability to compile modules separately, thus allowing the collection of tested and verified modules in a software library

In Modula-2, modules consist of two parts. These are a Definition Module and an Implementation Module. The definition module specifies the entities of the module that are available to the programmer. This essentially specifies the module interface. The implementation module includes the code that implements these entities. The implementation module may not be available to the programmer once the module has been verified. Using the module concepts of Modula-2, the modularity of systems and the concept of abstraction and information hiding are supported.

8.4 A Modula-2 Software Environment For Simulation.

There are a number of Modula-2 systems for IBM-PC or compatible systems. For the purpose of a demonstration model, we note that the model itself is made up of further modules. These modules have been developed by adopting a process viewpoint. That is, many of the modules which make up the model, run concurrently. We wish to support this in the simulation otherwise the module code must

For real-time simulation, however, De Almeida, MacLeod and Ypma (1985) propose that the time field of output messages be compared with a global real-time clock. The LP then sends the messages when the difference between message sent-time and the real-time clock is some defined constant ss. Thus only when:

(message sent time) - (real-time) = ss

will the transmitting LP send the message to other LPs.

The constant ss takes into account delays in the transmission of messages. This indicates the requirement that a LP must be able to generate the message before its message transmission time, ie: the LP must be able to simulate faster-than-real-time. Another requirement is that each LP must have synchronized real-time clocks or a global time reference. Faster-than-real-time simulation is a special case of real-time simulation and is discussed in Section 6.3.4.

For real-time simulation the LPs take on the properties of real-time systems. That is, they must be able to respond to inputs and take these inputs into account up to the message transmission time. Because the LPs must respond to all inputs we can predict model operation some time into the future. However, the LPs cannot be certain that the predictions made are correct until the events are to be simulated by message transfer in real-time. This is due to the fact that all output messages at time T are a function of inputs received up to time T.

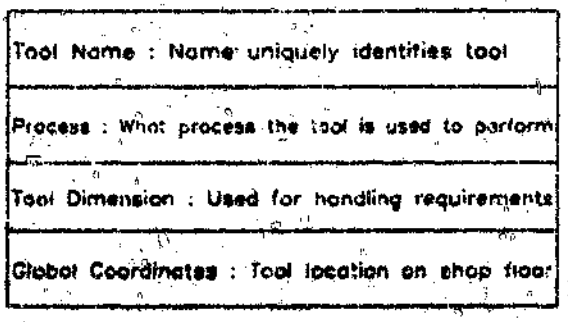
The above constitutes a constraint on the algorithm proposed by Misra (1984), since we can predict operations up to time T but if the predictions are incorrect, ie: some input is received between the time of prediction and T, then the LP must be able to update the predicted messages and this may require old predictions to be discarded and new ones to be made. This is a difficult problem to solve for which no simple solution exists. This is discussed by Misra (1984) in terms of rollback and recovery. This aspect must be included in the simulation algorithm.

Further, because of this it is not desirable to predict too far ahead, since the LPs may spend too much time generating events (predictions) which may be discarded later. We therefore impose the requirement that only one event be predicted ahead for each event that is independent. All events which occur

Figure A.2

Tool Representation

in the Simulation System



The above can be generalised for automated manufacture since in part handling, it is necessary to know the part position and orientation, not only for robots but also for machine tools, etc.

In the representation of parts and pallets, their location on the shop floor is specified by global co-ordinates. Material in a pallet has the same global co-ordinates of the containing pallet. The representation does not include orientation. Therefore a problem arises in satisfying the need for precise positioning requirements of material as in automated manufacturing systems. We attempt to solve this problem by modelling the pallet as a sequenced list. Parts will be removed from the pallet in sequence. If the parts are put in the pallet in the correct initial sequence and not according to position in the pallet, then this problem is solved. This is because part location is modelled as a sequence instead of in a programmed position.

be adapted for sequential execution. Therefore, the Modula-2 environment must support the execution of processes. Also since the simulation system is driven by a real-time clock, the processes must have access to the system's real-time. Finally, the models and the internal modules communicate by message passing, hence the environment must support these requirements.

Modula-2 provides a number of low-level features which includes co-routines. However, a kernel written in Modula-2, which supports the Monitor concept, described by Terry (1986), is deemed suitable for demonstration model implementation purposes. Monitors allow for inter-process communication and hence message passing. The kernel supports time-slicing where the user selects the required time-slice.

The only problem, however, is that the kernel was initially written for the Volition System Modula-2 Implementation which was not available for this project report. Further investigation into the various implementations available found that many implementations, although supporting co-routines, could not support task switching initiated by interrupts which is necessary for performing task switches at the appropriate time. This is mainly the lack of the IOTransfer primitive in these implementations. However, the Logitech Modula-2 implementation was found which supports the majority of these requirements.

A further problem with the Logitech system was identified. The problem being that the IBM-PC operating system, MS-DOS, is not re-entrant. Therefore, it is not possible to perform a task switch while MS-DOS is in a "critical section". However, this problem is overcome by checking whether MS-DOS is in a critical section before performing a task switch.

The kernel proposed by Terry (1986), had to be modified for execution by the Logitech system since the Logitech and Volition system implementations are not directly compatible. Further, a means for providing the system real-time and a check to see if MS-DOS is in a critical section or not, was added. The finalised Modula-2 kernel is shown in Appendix J. The following paragraphs describe the changes and additions that have been made to the kernel for execution on the Logitech Modula-2 implementation and also in order to meet the requirements presented for demonstration model implementation.

independently must be generated. An example of this is the material units "on" the transport model of the conveyor. For each material unit an event must be generated since these events are all independent. According to the above requirement we cannot generate more than one event for a material unit since these will be dependant on each other, ie: one must occur before the other.

A simulation algorithm which takes the above into consideration is proposed on the following page. The algorithm applies for the building block models under discussion. The real-time simulation algorithm allows the generation of events based on the model states, the model structure description and the inputs received in real-time.

We note that when messages are scheduled for transmission, the message with the shortest time until its transmission in real-time is sent next (unless it is cancelled).

6.3.4 Faster-Than-Real-Time Simulation.

For faster-than-real-time simulation we are still faced with the constraint that it is not possible to send output messages for predicted events until it is known that no further inputs will be received which may affect the output message. This constraint is a function of the response time of the PAS operating system.

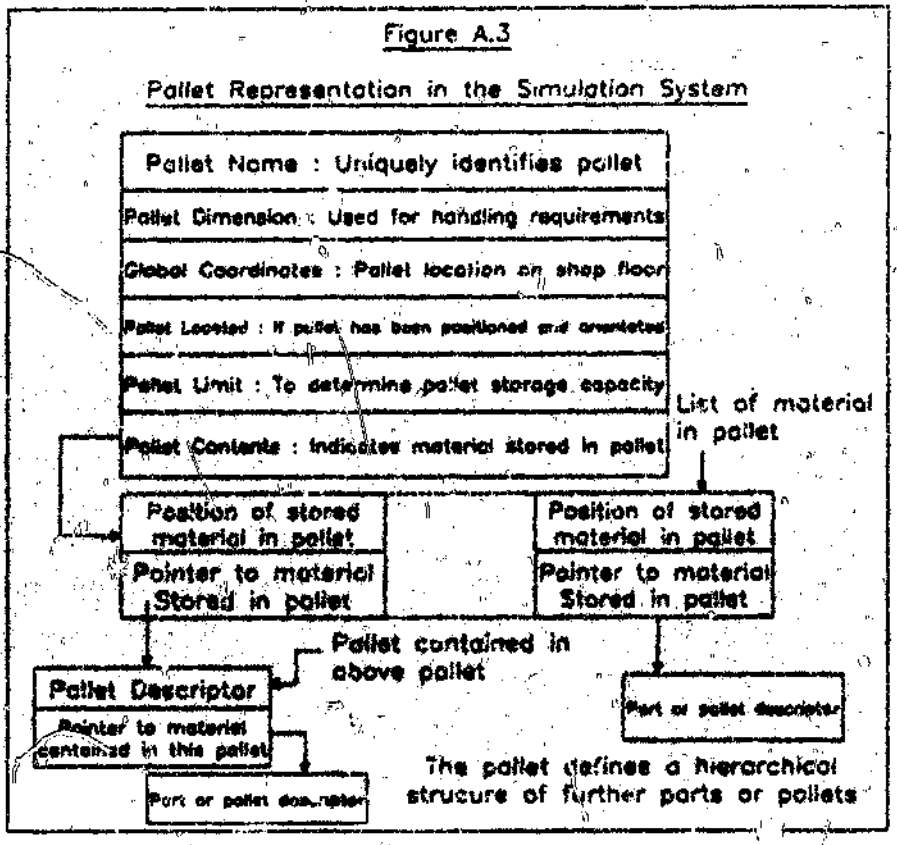
Further, LPs cannot run asynchronously as proposed by Misra (1984), since some LPs may simulate further in time than others. It cannot be assumed here that the PAS operating system will base its outputs on the times that inputs were received. That is, we cannot assume that the PAS operating system can identify that a process which normally takes 30 seconds to occur has been simulated in 5 seconds. Therefore, the inputs that the PAS operating system receives in faster than real-time operation must be globally synchronized. This is to ensure that sequence of decisions made in real-time simulation matches those made in faster-than-real-time simulation.

Taking the above into account, the representation of pallets is shown in Figure A.3. Pallets may only undergo material handling operations with the condition that the largest dimension of material being put into the pallet may not be larger than that of the containing pallet.

There are devices whose operation fall under the category of material handling that are used in automated systems to locate parts, containers and supports. If a part or pallet is subjected to these operations we assume that its position and orientation are then known. The operation may update the global co-ordinates of the material. If the material undergoes a locating operation this is also reflected in the material representation.

Figure A.3

Pallet Representation in the Simulation System



A.4 Summary.

In this appendix, material as used in the simulated manufacturing system, has been defined and described. We have also presented some of the reasons for the selection of the given material representations. The treatment of "material" in this

The module CLOCK was extensively modified to allow the installation of a clock interrupt handler in terms of the Logitech primitives available for this. These primitives are described in the Logitech library module named System.

The module CheckCriticalSection was added to determine whether MS-DOS is in a critical section, ie: MS-DOS functions are being executed at the times that a task switch is required. The module CheckCriticalSection uses an undocumented DOS call (34H) which checks if DOS is in a critical section and thus overcomes the non-reentrancy of MS-DOS (Logitech Modula-2/86 User's Manual).

To maintain the system real-time, a small section of code was added to procedure ClockHandler in module CLOCK. The code counts the number of clock interrupts which occur. In IBM-PC computers and compatibles there are approximately 18.2 clock interrupts per second. The simulation modules which require the system real-time use the module SystemTime. The procedure GetSystemTime in this module, returns the time in seconds that the system has been running.

8.5 Chapter Summary.

Because of the problems encountered with the Modula-2 programming environment the development of the AS/RS demonstration model was not completed. However, this is validated by the advantages that can be gained by the use of Modula-2 as simulation model implementation language, irrespective of the hardware system in use. Coupled with this is the fact that Modula-2 can be used in a distributed computing environment.

This chapter has presented a kernel written in Modula-2 which supports Monitors for the Logitech Modula-2 system. The use of this kernel is proposed for the implementation of the AS/RS demonstration model.

Real-Time Simulation Algorithm

```
While not simulation ending time do
    Begin
        While no inputs received and no messages scheduled for
            output
            do nothing.
        End {while}.

        If message(s) scheduled for output
            Then
                LP time := time of message transmission.
                Send all messages scheduled for output.
                Update the LP model state if required.
                Generate new event(s).
                Schedule message transmission time(s)
                corresponding to when the next event(s)
                are to occur in real-time.
        End {if}.

        If inputs received
            Then
                LP time := time of input message.
                Case input of
                    equipment
                    state change:
                        Cancel all message predicted beyond LP
                        time
                        Update LP model state.
                        Generate new event(s) (if equipment state
                        allows).
                        Schedule message transmission
                        time(s) corresponding to when the
                        next event(s) are to occur in real-time.
                    material arrival:
                        Generate new event for the material
                        (if equipment state allows).
                        Schedule message transmission time
                        corresponding to when the next event is
                        to occur in real-time.
                End {case}.
        End {if}.
    End {while}.
End {Real-Time Simulation Algorithm}.
```

Appendix is very general and will require an in depth discussion in order to solve more subtle problems which arise when material representation is considered in greater detail.

9 CONCLUSIONS

9.1 Review.

The modular modelling approach has led to a number of modules being identified each with a number of interfaces. The initial decomposition approach provided building block models which form the basis for the construction of a logical AMS system whose operation can then be simulated. The interfaces defined for these building block models allow their interconnection with the PAS operating system for logical system control. The models can also be connected to each other to simulate the interaction of the physical system shop floor devices with respect to material transfer between them. A common material transfer interface module is used to manage material passage between models and hides all aspects of material transfer that the models are not aware of at a physical level.

The building block models were further decomposed to identify an internal module structure. The internal modules specify how the building block model operation is implemented. The internal module structure is based on logical and mathematical models needed to cater for various types of shop floor devices and their behaviour. Each building block model incorporates a simulation algorithm which accepts building block model inputs and, in conjunction with the model state and model structure description, generates the necessary outputs in real-time or faster-than-real-time.

A distributed computing system architecture to support logical systems constructed from building block models and the execution of the simulation has been proposed. The architecture allows any LP to communicate with any other LP. This supports model communication for simulating the passage of material between the models and allows models to be controlled by controller building blocks. Further, the architecture supports the connection of the models with the PAS operating system. The simulation architecture thus maintains the modularity of the physical system being simulated. Building block models may be configured for execution over a number of processors. This facilitates real-time or faster than real-time simulation and animated displays of the simulation. The use of a distributed computing system architecture allows for the simulation applications presented in Section 1.4.

A global time reference is proposed for the faster-than-real-time case. Synchronization is achieved by a global real-time clock reference where the clock time is scaled by a constant Φ. Thus:

$$\text{(logical system time)} = \text{(real-time)} \times \Phi$$

where $\text{(logical system time)} > \text{(real-time)}$

The scale factor Φ is chosen so that the PAS operating system can respond sufficiently quickly to the logical system operation. This allows the sequence of decisions made by the operating system in faster-than-real-time operation to match those made in real-time operation.

In summary, for faster-than-real time simulation, a global logical system time reference is required where system time is some multiple Φ of real-time. Φ is a function of the PAS operating system response, the execution speed of the individual LPs and the required speed of simulation.

6.4 Modelling Equipment Breakdown.

An important aspect of modelling automated equipment units is the ability to simulate equipment failures. In a real time simulation system as proposed here, the effect of breakdown will test the control system in its ability to take contingency action.

There are various ways of modelling equipment failures and whichever method is chosen it can simply be implemented by manipulating model outputs. For example, the failure of a conveyor transport model motor can be simulated by not scheduling any output messages for the particular conveyor model.

The user can be given the choice of selecting which equipment breaks down, the time of breakdown and the equipment downtime. Breakdowns may also be modelled statistically where the user provides the probabilities for equipment breakdown.

Although we have not incorporated the possibility of equipment breakdown in the internal module structure, this should not be too difficult to add. Each LP can incorporate a breakdown routine which prevents the transmission of messages or the reception of inputs and hence material transfer or sensor activation.

APPENDIX B: AUTOMATED MANUFACTURING SYSTEM INTERFACES

Of the interfaces that exist in AMS, one of the most important is the area where material is exchanged between shop floor equipment. Material must move between equipment so that it can be processed to achieve manufacturing objectives. These types of interfaces are characterised by their operation and physical structure. To model these interfaces and to include all the necessary detail, the interfaces are discussed in this appendix. This is to provide a knowledge base whereupon the characteristics of the interfaces can be deduced and hence modelled. The various type of interfaces are discussed.

B.1 Automated Storage and Retrieval System Interfaces.

The AS/RS consists of storage racks, a storage and retrieval machine (S/R machine) and Input/Output (I/O) stations or Pick and Deposit (P/D) stations. The S/R machine is mounted along a rail on the floor. The frame of the S/R machine is guided by support from the storage structure at the top of the structural frame. Mounted on the structural frame of the S/R machine is a carriage that moves vertically along the frame with the load. The shuttle is the load-supporting mechanism on the carriage that moves loads into and out of storage locations and on and off the P/D stations. The P/D stations provide the locations at which loads enter and leave storage. The P/D stations support the load in a manner that is suitable for interface with the S/R machine. The handling mechanism on the S/R machine usually consists of a shuttle table or a mechanical clamp. Typically, each S/R machine operates in a single isle and services storage racks on each side of the aisle.

There are various types of AS/RS. The unit load AS/RS is used to retrieve loads that are palletized or unitised and weigh over 500 pounds (Tomkins and White, 1984). The miniload AS/RS is appropriate for storing and retrieving small parts that can be stored in a storage bin or drawer. The S/R machine for the miniload system is similar to that of the unit load system, however, it is generally equipped

Finally, Modula-2 has been proposed as a high-level language for expressing the models in executable form. The reason for this choice being that Modula-2 supports the concepts of abstraction and information hiding which forms the basis of the modular modelling approach used throughout this project. Modula-2 has also been chosen since it supports simple real-time programming primitives which may be extended to include advanced real-time (see Terry, 1986) and distributed real-time programming primitives.

9.2 The Modelling Methodology.

In this project report a modular modelling approach has been used to identify the steps necessary to construct real-time models of AMS shop floor devices. The modular approach has been applied to specific systems or devices in the AMS. Although the models for each case cannot be generalised, the steps needed to develop real-time models for each case apply. The steps required in developing real-time models form the modelling methodology of this project report and are enumerated in this section.

Consider the case of modelling a specific AMS shop floor device or system. The steps necessary to develop real-time models for use in the simulation system architecture are as follows:

1) Identify the building block model(s) which represent the physical process to be modelled.

   A number of models may be identified to represent the shop floor device or system. In general, the models identified will represent the independent operations or activities which together represent the operation of the physical process.

2) Specify building block model interfaces.

   The building block model interfaces are identified in terms of a common material transfer interface, control, AIS and signal interfaces. These interfaces are needed for model control and the processing of logical parts. Material transfer protocols must be compiled for all possible interfaces which

In summary, the LPs can also incorporate the requirements of modelling equipment breakdowns which is implemented by interfering with the normal transmission of messages.

6.5 Graphical Animation Of Building Block Model Operation.

The simulation system would be useless if the system users cannot gain insight into the equipment operation by means of the building block models. Obviously, information may be obtained from the PAS operating system monitoring processes which give the PAS operating system a view of the overall process. However, the equipment models have one very important property. This is the ability to allow the measurement of variables which cannot be easily measured, if at all, in the physical system. Coupled with computer aids the information obtained may be manipulated into a format most useful to the simulation system user.

In the proposed simulation system there are numerous ways of obtaining information from the models. Since we are at the level of detailed model operation, this is an appropriate place to discuss the aspects of presenting graphically animated model operation in real-time.

Each equipment model contains all the information necessary to describe its operation at variable, discrete instances in time. This forms the basis for animated displays of each equipment or building block model or possibly the overall shop floor operation.

What is intended here is to display the motion of material, the travel of the AGVS trucks on the shop floor, the S/R machine in action, etc. This type of animation forms a very convenient means of providing information to the user. However, there are problems involved and these are discussed.

The major problem is due to the fact that information necessary to describe model operation is only available at variable, discrete instants in time. Animated displays are continuous and movement is depicted by updating the moving objects at small enough time increments on the screen so that the motion seems continuous to the viewer. In the simulation system presented here, the time interval between the discrete time instances may vary in the order of seconds or even minutes. Therefore, to provide an animated system will require the interpolation of model operation between these intervals.

with additional or more sophisticated extractors and guidance mechanisms in order to meet the requirements imposed by tighter clearances. Other types of AS/RS include deep lane AS/RS and storage carousels.

Interfacing AS/RS and Conveyor Systems.

In many cases, AS/RS systems are interfaced at the P/D stations via a conveyor system. In all cases, the conveyor system eventually interfaces with the pickup and discharge spurs of the S/R machine. Sometimes, the transverse conveyor (which crosses the front of the system) is a chain driven live roller conveyor with chain transfers and [illegible] conveyors bringing the material into the pickup station. In most cases, a squaring operation is performed at the point of transfer and a final positioning operation is performed at the S/R machine pickup point. On the output side, deposit stations operate similarly but do not square the pallets (Budill, 1984).

Interfacing AS/RS and AGVS.

The interface of the Automated Guided Vehicle System (AGVS) is necessary for many types of automated systems. In unit load systems, AGVS equipment drops loads at pickup stations of the AS/RS aisles. However, AS/RS throughput must not be limited by AGVS throughput or vehicle availability. One method of preventing this is to install a conveyor system as a buffer between the AS/RS and AGVS. This permits accumulation, positioning and other functions necessary for proper flow of materials and frees the AGV equipment to simply drop off and pick up material (Budill, 1984).

In the case of unit loads, the AGV unit load carriers may have either lift/lower tops for depositing material on stands or powered roller conveyor tops to interface with conveyors.

In general, the AS/RS and conveyor interface is active-passive in nature. The S/R machine load handling mechanism is active in that it is responsible for acquiring or disposing of the load at the pickup and deposit stations and the storage locations in the rack.

transfer material if they do not exist already. Other interfaces may be specified between the models which make up the physical process and are necessary for the device or system operation as a whole.

3) Construct a logical model of the physical process.

This step requires an in-depth analysis of shop floor devices or systems used in practice which perform the same basic function as the physical process performs. A collection of all parameters needed to describe the variations in operation are formed into a structure which leads the user in the selection of these parameters. In this way the user specifies building block model operation for his particular use.

4) Identify the internal modules needed for building block model implementation.

Steps 1 to 3 identify the external view of the building block models. These steps are performed under the constraint that the models identified must be modelled to look exactly the same as the shop floor device or system.

In step 4 and the following steps, the modelling viewpoint changes to facilitate the implementation of the behaviour of the building block model. In this step each of the following are considered and a convenient means of representation specified:

(i) Mathematical model:
This is the mathematical representation which models the physical operation of the shop floor device or system at the required level of detail.

(ii) Equipment states:
This is the identification of the different operational states that describe the physical operation of the shop floor device or system.

(iii) Model states:
This is a representation of material movement or mobile equipment motion over time. This representation is updated as specified by the real-time simulation algorithm.

The reason for this is that the models themselves cannot be concerned with generating the graphic information necessary since this may ensure that the models do not run in real-time. The process must be handed over to a graphics display module for each equipment model. The graphic display module must then be responsible for receiving system state information when available, interpolating model operation between time intervals and displaying this graphically in real time.

The overhead associated with this scheme is not minimal. The scheme requires a graphics interface whereby each model sends the necessary state information to the corresponding graphics module. The graphics module must incorporate the same mathematical models as the equipment or building block model so that interpolation can be performed. The graphic module must also have the capability to ensure that the state of the displayed model matches that of the equipment model at the end of the interpolation period.

The advantages of this scheme include the fact that the graphic modules may each run on separate processors which leaves the equipment models free to simulate and will aid in achieving real time graphic displays. Also, each equipment or building block model operation may be displayed separately or may be combined with a central graphics process which can then display sections of the shop floor with each section containing a number of building block models. Finally, the graphic modules can store the information received so that the logical system operation may be replayed off-line.

The level of graphic detail depends on the amount of useful information that the user needs. Obviously, the detail of material position and orientation is not a requirement of the displays for this simulation system. The flow of material through the system and the occurrence of real time events is important. In the case of robots, although enough information exists to show the detailed arm motion, only the end effector position is required. CNC machines can be displayed by showing their states as busy or free and material loading/unloading. Real time events such as material being lost can be depicted by showing the material falling on the floor.

In summary, the role of information presentation is of major importance in any simulation. The design of an information display interface occupies an important aspect of any model development process. Further, the manipulation of raw data

B.2 Automated Guided Vehicle Interfaces.

The Automated Guided Vehicle System (AGVS) is an especially flexible system for horizontal transport and is suitable for both simple transport operations with a small number of destinations as well as complex and centrally controlled transport processes. The flexibility of the system lies in the guidance wire which is usually responsible for the inductive steering of the truck (the AGV).

The concept of an AGVS includes all systems which are capable of functioning without a driver. The main components of an AGVS are:

1) the truck or trailer.
2) the floor system with the installation of the wire guidance system and the information transfer system.
3) the load transfer equipment which can be both on-board the truck and/or in a stationary position.
4) the truck and traffic control system.

The load handling system represents a common component belonging to the truck side and the network side on the physical level. The following three possibilities exist:

1) the truck is exclusively active.
2) the stationary load handling system is exclusively active.
3) the truck and the load handling system are both active.

There are a large number of different types of AGV and different mechanisms for load transfer. These are succinctly summarised by Muller (1983). The graphic summary is reproduced here in Figure B.1. Some of the interfaces which apply to automated manufacturing system more than others are discussed.

Consider the case where both the truck or the tractor train and the load transfer station are active during load handling. In this case the transfer of pallets occurs from stationary conveyors to conveyor segments mounted on the tractor train (chain or roller conveyors). The tractor trains with lifting platforms are also used to lift pallets from stationary chain conveyors or deposit pallets on them.

(iv) Model Structure Description:

This is a convenient representation of the shop floor device or system structure. The representation facilitates the generation of next events associated with material or mobile equipment.

5) Relate model inputs to model outputs in real-time.

In this step the real-time simulation algorithm is used to relate model inputs to model outputs using the mathematical model, the equipment states, the model states and the model structure representation. In this step the modules of the building block model are identified.

6) Specify the internal module interfaces.

The interfaces of the internal modules are specified in terms of how modules interact with each other. This may be in the form of message passing or procedure calls.

7) Implement all the modules identified in the previous steps in the chosen implementation language.

8) Building block model validation.

In this step the building block models are analysed for all types of operation specified by the logical model and the various interconnections with other building block models. Model validation requires comparing the sequence of model outputs and their timing with the corresponding shop floor device or system in real-time.

9) Addition of building block models to the simulation system library.

Once model validation has been completed and the associated modules operate as required, the building block model is then added to the library of the simulation system.

9.3 The Requirements For Distributed Simulation Of AMS.

Section 9.2 enumerates the steps needed in the real-time modelling of AMS shop floor devices or systems. This section presents the requirements for successful operation of a simulation constructed from autonomous modules which com-

into an understandable and useful format will greatly enhance system understanding and give credibility to the simulation system. Therefore, the incorporation of graphic modules as described, albeit at a high overhead, is justified.

6.6 Constructing Building Block Models.

The previous sections have identified all the aspects which must be taken into account to construct each building block model. These aspects identify a number of modules which, collectively, implement the operation of each building block model. The internal module structure of each building block must support the logical model and the mathematical models necessary to describe the physical behaviour of the corresponding shop floor device.

In general, each building block model will include a model structure description module and some means of maintaining the model state over time. The structures necessary to implement each building block model cannot be generalised and are thus discussed in an appendix for each case.

The conveyor transport model internal structure is discussed in Appendix E and an example of this model's operation with respect to the real-time algorithm is presented. The AS/RS, the AGVS, the robot and the controller model internal module structures are briefly discussed in Appendices F to I respectively.

6.7 Chapter Summary.

This chapter presents the requirements for the identification of the internal modules of the building block models so that real-time and faster-than-real-time models may be developed. Further, requirements for modelling equipment breakdowns and a means of presenting model operation by graphical, animated displays have been discussed.

With pallet trucks, which use forks for load transfer, automatic unloading is carried out by lowering the forks at the destination. Automatic loading of forward moving trucks is achieved by means of an additional load transfer station with lifting and lowering equipment.

For skid tractors, the load is either taken from a console, a conveyor or a loading unit with enough clear height. The lifting and lowering equipment on the truck is usually active during handling.

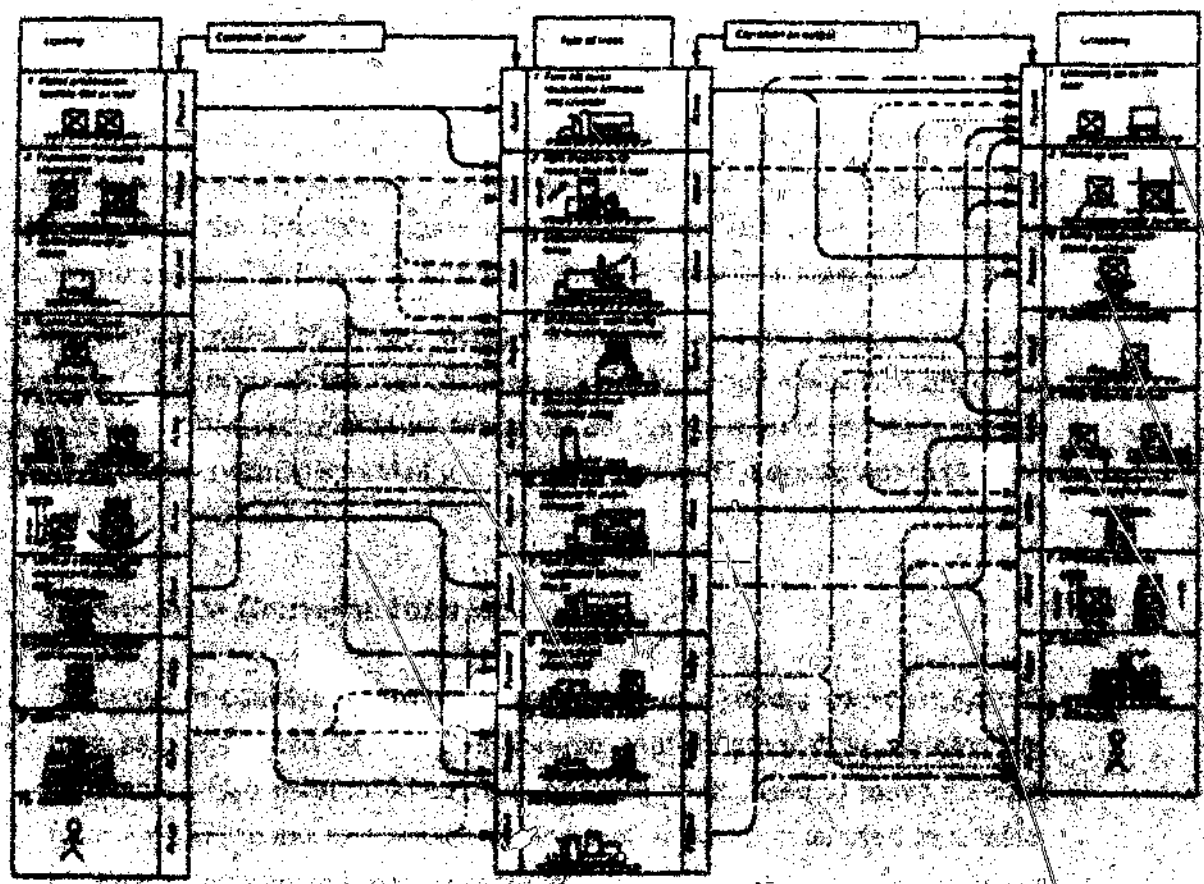


Figure B.1
Classification of Load Transfer for Driverless Transport Systems
(Reproduced From Muller, 1983)

municate by exchanging messages. These requirements are discussed by De Almeida, MacLeod and Ypma (1985) and are presented here specifically for the case of AMS simulation.

The requirements for distributed simulation are:

9.3.1 General Simulation:

(i) From an external observer's point of view, a LP at any level is modelled to look exactly the same as its corresponding PP.

(ii) A suitable simulation architecture must be chosen to ensure that any LP can communicate with any other LP.

(iii) A suitable algorithm must be devised to ensure that parallel execution of LPs is possible.

9.3.2 Real-Time Simulation:

(i) All the general simulation requirements.

(ii) Every LP must be chosen to ensure that it always simulates faster-than-real-time.

(iii) Each LP in the distributed simulation must have access to a global real-time reference.

9.3.3 Faster-Than-Real-Time Simulation:

(i) All the general simulation requirements.

(ii) All the real-time simulation requirements.

(iii) All the LPs must execute at the same multiple of real-time so that the PAS operating system can maintain synchronization between LPs.

9.4 Recommendations For Future Investigation.

The areas for future investigation, as identified in this project report, are presented.

7 AN ARCHITECTURE FOR SIMULATION

Up to this stage of the work a modular simulation system has been proposed where the advanced manufacturing system has been divided up into a number of models called logical processes. The logical processes include AS/RS, robot system, AGVS, controller and conveyor building block models. Collectively, these models simulate the operation of the AMS.

This chapter discusses a computing system architecture that supports the interconnection of these models in forming the logical AMS and the running of the real-time or faster-than-real-time simulation. The computing system also supports the configuration of the logical system and this is also discussed.

7.1 An Architecture For Advanced Manufacturing System Simulation

The architecture proposed is adopted directly from that proposed by De Almeida and MacLeod (1985). This is a locally-distributed multiple-processor computing system architecture for real-time control applications.

The architecture of a simulation cluster is shown in Figure 7.1. The simulation cluster consists of the following processing units (PUs), which communicate over a LAN.

1) Interface processing unit (IPU).
2) Library server (LS).
3) User interface/cluster configuration (UI).
4) Processing units. Each processing unit has its own memory denoted by M as shown in Figure 7.1.

The interface PU is discussed here while the other PUs are considered in Section 7.2.

9.4.1 Standardization Of Interfaces.

As noted there are no standard interfaces for shop floor device interconnection with host computers. In terms of the interfaces presented here a standard means for equipment control is necessary. This requires an in-depth investigation into the primitives necessary to control shop floor devices used in practice. A set of primitives for each device must be compiled. Further, the information necessary to maintain sufficient information relating to device operation must be specified. Finally, a standard for program downloading/uploading is necessary.

9.4.2 Construction Of Models.

This project report has considered the requirements for model construction with the objective being to define a methodology for the development of real-time shop floor device models. The work presented in this project report has been tailored to this goal and thus the detail necessary for the implementation of general logical system building block models, along with comprehensive logical models, has not been considered. For this reason, the modelling methodology must be applied to develop such models to the required level of detail for model implementation and validation.

9.4.3 Simulation System Use As An Analysis Aid For AMS.

In this project report possible applications for the distributed simulation of AMS have been outlined in Section 1.4. Further investigation is required as to how the logical system is actually used in such applications. These applications require information and the analysis of this information. Apart from the graphical capabilities of the models, the logical system can generate information at all levels of the control hierarchy. A means for collating this information and deriving useful measures of system performance from this information is an area of future investigation.

In summary, the characteristics of the interface of AGVs and other equipment on the shop floor include the following possibilities: active-passive, passive-active, or active-active.

B.3 Conveyor System Interfaces.

Conveyors are used when material is to be moved frequently between specific points over a fixed path. The major types of conveyors used in automated manufacturing systems include power and free conveyors, monorail conveyors, belt, roller, trolley or tow type conveyors. Because of the characteristics of roller or chain conveyor systems, they play an especially important part in automated manufacturing systems.

Continuous floor transporters, such as powered roller and chain conveyors, are suitable for transport, collection and distribution. Roller and chain conveyors are often used at the interface between a storage area and other equipment as decoupling units. They serve as buffers at the interfaces to transport systems.

Accumulation roller conveyors occupy a special position among continuous transporters because of the task of filling time by buffering which is as important as other functions. Accumulation conveyors are always used in conveyor systems when the delivery and disposal of transport units cannot be synchronized (Muller, 1983).

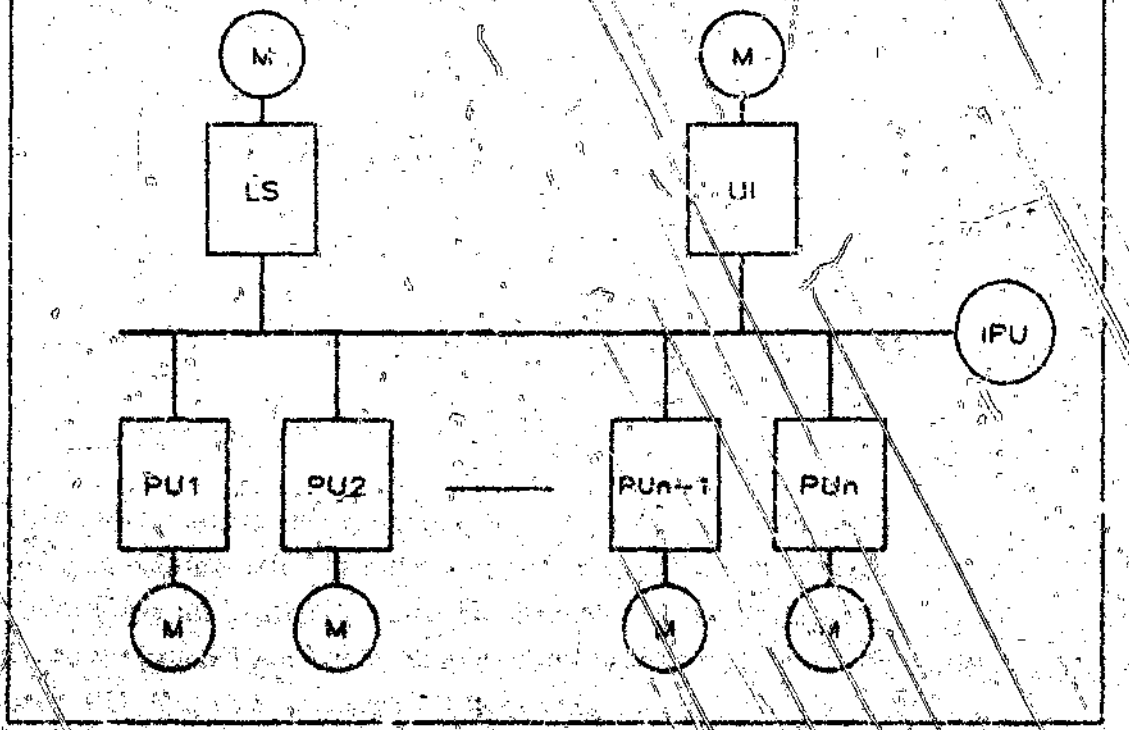
Conveyor to Conveyor Interfaces.

Conveyor to conveyor interfaces exist when two different conveyor systems interface. These may be conveyor sections with different drive systems, two conveyors which meet at 90 degrees, outward points used to move material from a main section to a branch section and inward points, used to transfer material from branches to main sections.

Load transfer is effected when the load is transported to the extremity of the conveyor system and moves onto the next conveyor section. One conveyor section may be active while the other is passive or both may be active (Eg: two powered roller conveyor sections both in motion).

Figure 7.1

Architecture of a Simulation Cluster
(Reproduced from De Almeida and MacLeod. 1985)



7.4.1 The Interface Processing Unit.

The main reason for choosing this simulation architecture is that it allows the on-line connection of the PAS operating system and physical system to the logical system. This is achieved by connection of the PAS system LAN with that of the simulation cluster via the interface processing unit. Such an interconnection allows the realization of the applications enumerated in Section 1.4.

The interface processing unit manages message transfers between the simulation clusters and the PAS system. The receiving interface PU renames inter-cluster messages so as to ensure that they are compatible with the naming conventions of the receiving cluster and will be routed to the intended receiving PU.

9.5 General Observations

Shapiro (1987) notes that the real value in computer-integrated manufacture (CIM) is derived from the total integration of data collection and from the automation of all aspects of the manufacturing process. To be effective, a CIM system should encompass everything from conceptual design through product delivery and include computer-integrated drafting, design, engineering and test.

In attempting to provide a modular modelling philosophy which can be integrated into the concept of CIM, a number of implications in the work presented are noted:

1) The system level approach adopted for AMS simulation has required that all aspects of AMS be considered. This has precluded a detailed investigation. However, some detail has been presented to allow discussion of the simulation concepts proposed.

2) The information that must be assimilated for a systems approach is greater than that required for a dedicated investigation into specific areas of AMS simulation. It is necessary to include this information in the report as background to the concepts proposed.

The major impact of the above is the length of the report as well as a lack of detail in certain areas of investigation.

In review of the work presented, I think that the objective of the report has not been achieved entirely. This is due to the fact that the modelling philosophy presented has not been verified by practical examples of AMS simulation. However, the report has presented a basis for a modular modelling philosophy which could be used to construct a practical AMS simulation. This is achieved on the following basis:

1) An integrated approach to AMS simulation.

2) The development of a Material Transfer Interface.

3) A modular architecture for AMS simulation.

In taking up this project at the point of this report, I would suggest a concerted effort be made to develop the AS/RS and conveyor building block models. This would allow a simple simulation system to be implemented which could then be

B.4 Robot Interfaces.

The robot, as defined by the Robot Institute of America, is a programmable, multi-function manipulator designed to move materials, parts, tools or specialised devices thr ugh variable programmed motions for the performance of a variety of tasks (Productivity International, 1980). Industrial robots generally consist of the frame/structure, the manipulator, the controller, the power supply and in more sophisticated robots a sensor system. The manipulator is a servo-controlled mechanical arm which is capable of positioning the end effector (gripper) anywhere in three dimensional space within the limits of the working envelope of the robot.

Robot tasks fall into two categories: material handling and fabrication. The robot can execute these tasks because of its ability to use tools and grippers. Grippers are mechanical grippers and surface attachment type devices such as magnets and suction cups or various tools. The mechanical grippers usually employ moveable, finger-like, levers and any one robot hand might have several sets of these opposed fingers. The grippers of the robot form the material transfer interface with other equipment since this is how material and tools are manipulated.

Robots can generally interface with any type of equipment. However, robot interfaces with other equipment are characterised by their precise position and orientation requirements of material, tools, etc. This requires that the position and orientation of any material handled by the robot be "programmed" so that the robot will know its position when the robot attempts to pick up the material. Because of the robot repeatability, the robot can return loads to programmed positions.

In the case of robot-conveyor interfaces, the material must be correctly positioned and orientated before the robot may pick it up. In some cases, a means of preserving the orientation and position of the material is necessary when the robot deposits the material on the conveyor. To this end, stopping and clamping devices on the conveyor are used to stop and then clamp material in position. Also pallets with parts or tools precisely positioned in the pallet are used. The pallet is also stopped and clamped in a known position. The robot can then manipulate the individual parts. This is generally the case for all robot interfaces and jigs and fixtures along with customised means are used to position and orientate material.

7.2 Constructing A Logical System For Simulation.

We discuss how each processing unit (PU) in the simulation cluster is used in the construction of a logical simulation system, using the building block models or logical processes.

7.2.1 Processing Units.

The building block models execute on the processing units. The internal modules of the building block models may be distributed as required over a number of PUs. Alternatively, more than one model may run on a single PU.

7.2.2 Library Server Processing Unit.

The building block models and their associated modules are stored in the library server processing unit. Once the user has selected how the models are distributed over the processing units, the modules from which the logical system is to be constructed, are sent to the appropriate processing units from the library server PU. 7.2.3 User Interface/Cluster Configuration Processing Unit.

The user interface/cluster configuration PU is used for the following:

In setting up the logical system the user must select each equipment or building block model. For each model selected, the user is guided within the framework of the building block's logical model. In this way the specific module characteristics are chosen and the control, signal, AIS and material transfer interfaces are set up. If necessary, equipment models which require controller models are interconnected by specifying the signal interface connections, Eg: controller model output 1 is connected to conveyor transport model 1, input 3. This can be realized using graphical means.

Once the logical system has been constructed and the distribution of the internal modules of each building block model over the processing units has been decided, the necessary modules are transferred to the respective processing units from the library server PU. The user can then specify the starting and

In automated manufacturing systems the robot is used to load and unload and to change tools of computer numerically controlled (CNC) machines. These machines may have their own customised load/unload and tool changing facilities but will generally exhibit the characteristics of robot type interfaces. The robot interface is active-passive in nature since the robot is responsible, in almost all cases, for acquiring a load and disposing of the load.

One further interface not yet discussed is the robot-robot interface. This is the case where two robots manipulate and transfer material between them. Research is currently underway to study the requirements of this interaction so that an understanding of manufacturing-device co-ordination can be developed. It is hoped that the deductions from this study will aid the development of fully flexible automated manufacturing systems (Hawker et-al, 1986).

used to test the proposals put forward here. The major point is to limit the scope of the development such that a working prototype may developed within the time frame allowable for such a development.

In summary, this project report has served as a vehicle for acquiring a small part of the extensive knowledge base needed in working with advanced manufacturing systems. Valuable experience in the fields of software engineering, simulation and real-time distributed computer control systems has been gained.

9.6 Concluding Statements.

In Chapter 2, the use of physical and digital simulation for the analysis of AMS is presented. Physical simulation provides flexibility with regard to control system evaluation and is the major advantage of this approach. In the case of digital simulation, the programmability of the computer and the man/simulation interface allows a variety of systems to be simulated at almost any level of detail. Packages for simulation fulfil their role in the design and analysis of AMS but are generally inflexible with respect to control system evaluation as is required in today's AMS environment.

The simulation system proposed here utilizes concepts of modular modelling, previously used mainly in the control field, along with distributed simulation concepts to provide a flexible simulation system. Thus various types of AMS systems may be simulated where control is exercised from the actual control used in practice. Further, faster-than-real-time simulation allows the control system to be tested to its limits.

This simulation approach closes the gap between physical and digital simulation methods by providing for control system evaluation with the capabilities of digital simulation, which allows almost any AMS to be evaluated. The system proposed in this project report can be used as a design tool for AMS analysis, verification and optimization and as a research tool for:

1) the standardization of AMS building blocks and hence the provision of off-the-shelf software and
2) for providing an accepted and integrated AMS control system structure.

stopping times of the simulation run and the simulation can be initiated. Note that the internal modules of single building block models may also be distributed over a number of PUs depending on processing speed requirements.

The graphical output from each model can be observed by display screen(s) connected to the PUs where the graphics modules are executing. Also sections of the logical system operation may be displayed from the display screen(s) of the central graphics modules. This is achieved by displaying the output of a number of graphics modules on one display screen, thus giving an overview of the operation of sections within the logical system.

7.3 Chapter Summary.

In this chapter we have adopted the distributed simulation computing system architecture as proposed by De Almeida and MacLeod (1985). The major reasons for adopting this architecture is the realization of the applications as enumerated in Section 1.4. Secondly, the simulation architecture captures the modularity of the physical system, thus providing ease of logical system configuration. Finally, the architecture provides the advantages associated with distributed architectures, namely, modularity, flexibility of communication, reliability and maintainability .

APPENDIX A: REPRESENTING MATERIAL IN THE SIMULATION SYSTEM

In Chapter 4, material processed by the simulation system, that is, models of the actual material in the automated manufacturing system, is classified as a collective description of the following :

1) Parts.
2) Tools.
3) Pallets.

In this appendix, each of the above is described in detail and a structure for each is presented. Note that in the discussion here, the simulated automated manufacturing system is denoted by the term "logical system".

A.1 Parts.

Parts are the units of production of the logical system. A part in the logical system represents material from the raw stage, the work-piece stage (during processing), up to the finished product. The structure used to represent parts is shown in Figure A.1.

Since parts cannot be described quantitatively (unless a geometric or solid model is used), we describe a part in terms of the production operations it is subjected to. For example, transport operations affect the location of the part hence, the need for global co-ordinates. These are three dimensional cartesian co-ordinates which specify the location of the part on the logical shop floor relative to an arbitrary reference point. We are not concerned with part orientation in this work.

A part is also characterised in terms of a part list and a fabrication list. The parts list shows the parts which have been used to assemble the part under discussion. The parts list also shows the assembly sequence, and if required, the time of assembly. The part list can be compared to a bill of materials. The fabrication list is a list of all processing operations which a part has been subjected to. An entry on this list might include the following: riveting at machine 1. The fabrication list is similar to a route sheet. The part list and fabrication list may only be updated at robot or machining stations which are capable of performing these operations.

APPENDIX C: MATERIAL TRANSFER PROTOCOL SPECIFICATIONS

This appendix includes the protocol specifications for three model interfaces. These are:

1) Conveyor/Conveyor Interface.
2) Automated Storage Retrieval System/Conveyor Interface.
3) Robot/Conveyor Interface.

These examples have been chosen to illustrate the protocol specifications for active-active, active-passive and passive-active interface types. The robot interface protocol has been illustrated due to its special nature. The following are the terms used in this appendix.

Located:

This is any material handling operation which accurately locates a part or pallet with respect to its position and orientation. To change the material state from not located to located will require an active operation on the material. For example, on the conveyor, a pallet or part is stopped and then physically clamped in position using pneumatic or other means. This locates the pallet or part with a known position and orientation and hence the material the pallet contains. To maintain this state jigs and fixtures are used. In the simulation system, material that is put into a jig or fixture will maintain its "located" status.

Load:

This refers to material which is to be transferred between building block models.

Receiving or Sending Modules:

The terms receiving or sending modules refer to the sending and receiving protocol modules that exist within the common material transfer interface module (see Chapter 4). In some cases the terms sending or receiving interface modules are used when the use of the above terminology is confusing.

8 MODEL IMPLEMENTATION FOR DEMONSTRATION OR VALIDATION PURPOSES

There are three major reasons for developing demonstration models. These are:

1) To gain experience with the theoretical concepts proposed.
2) Validate the modelling approach and simulation algorithm.
3) Gain credibility for the modelling approach.

Work has been started towards the development of demonstration models. The most suitable model for demonstration purposes is deemed to be the AS/RS. The reason is that this system incorporates all the concepts proposed. The system requires a controller model and the incorporation of all options as specified by the AS/RS logical model. Secondly, the possibility of comparison and interconnection with the AS/RS at UWTec (see Lun, 1986) for model validation purposes exists.

This chapter discusses the selection of a computer and software system along with a real-time environment for model implementation.

8.1 A Demonstration Model Proposal.

A working demonstration model of the AS/RS is proposed where simple operation of the PAS operating system and material transfer to and from the AS/RS logical system is simulated as well. This necessitates the sending of PAS primitives to the AS/RS controller model with associated material arrivals and removals. Also a simplified means of monitoring the AS/RS operation along with extensive animation display facilities is necessary.

8.2 Selection Of A Computer System.

Due to the availability of IBM-PC computers and compatible computers, which have the desired graphic capabilities, these computers are deemed most suitable

C.1 The Material Transfer Protocol Specification For A Conveyor/Conveyor Interface.

This specification describes the protocols for material transfer between two conveyor models. This specification supports an active-active interface. An example of this type of interface is the interconnection of two powered roller conveyors.

Input/Output Point Definition.

1) Points at which material is received by the conveyor model are defined as "input" points. Each input point is characterised in terms of three dimension cartesian co-ordinates. The dimension of the input point corresponds to the physical conveyor width.

2) Points at which material is transported off or removed from the conveyor model are defined as "output" points. Each output point is characterised in terms of three dimensional cartesian co-ordinates.

3) The same point along the logical conveyor length may be defined as either an input point or output point or both.

The input/output co-ordinates are called global co-ordinates and are defined relative to an arbitrary reference chosen on the logical (simulated) shop floor.

Material Transfer From A Sending Conveyor To A Receiving Conveyor.

In this section the message passing, condition evaluation and condition failure action for the conveyor/conveyor interface is described.

Condition Evaluation at the Sending Conveyor.

Material transfer is initiated by the sending conveyor model when the simulated direction of conveyor motion is towards the output point and the load for transfer reaches the output point. In this case, the sending conveyor interface module sends a request to the receiving conveyor interface module for it to accept the load. The sending conveyor interface module sends the load to the

receiving conveyor interface module along with all the necessary information for condition evaluation. The sending conveyor interface module then waits for a reply.

Condition Evaluation at the Receiving Conveyor

The receiving conveyor interface module will accept the load if the following conditions are fulfilled. This is called condition evaluation.

A) The sending conveyor output point corresponds to the receiving conveyor input point (in terms of global co-ordinates).

B) The load can be handled by the receiving conveyor model (in terms of logical conveyor width).

C) The input point is not occupied by another load.

D) The receiving conveyor transport mechanism must be in motion in the correct direction to accept the load at the input point.

If all the above conditions are fulfilled, material transfer is successful. The receiving conveyor interface module sends a positive reply to the sending conveyor interface module.

Condition Failure Action at the Receiving Conveyor.

We discuss the action taken by the receiving conveyor interface module if any one of the conditions A to D above fail.

A) Failure of this condition implies that conveyor models on the logical shop floor have been configured incorrectly. For example, the conveyor heights are different, ie: the conveyors are not aligned. In this case the load is assumed lost to the simulation system. The reply to the sending conveyor interface module is negative.

B) Failure of this condition implies that the load is too big to fit on the receiving conveyor. The load is assumed lost. The reply to the sending conveyor interface module is negative.

C) If condition (C) or (D) fail then transfer cannot be completed. The real world analogy is that the load queues up at the output point of the sending conveyor model. The reply to the sending conveyor interface module is negative along with the fact that the load is not lost.

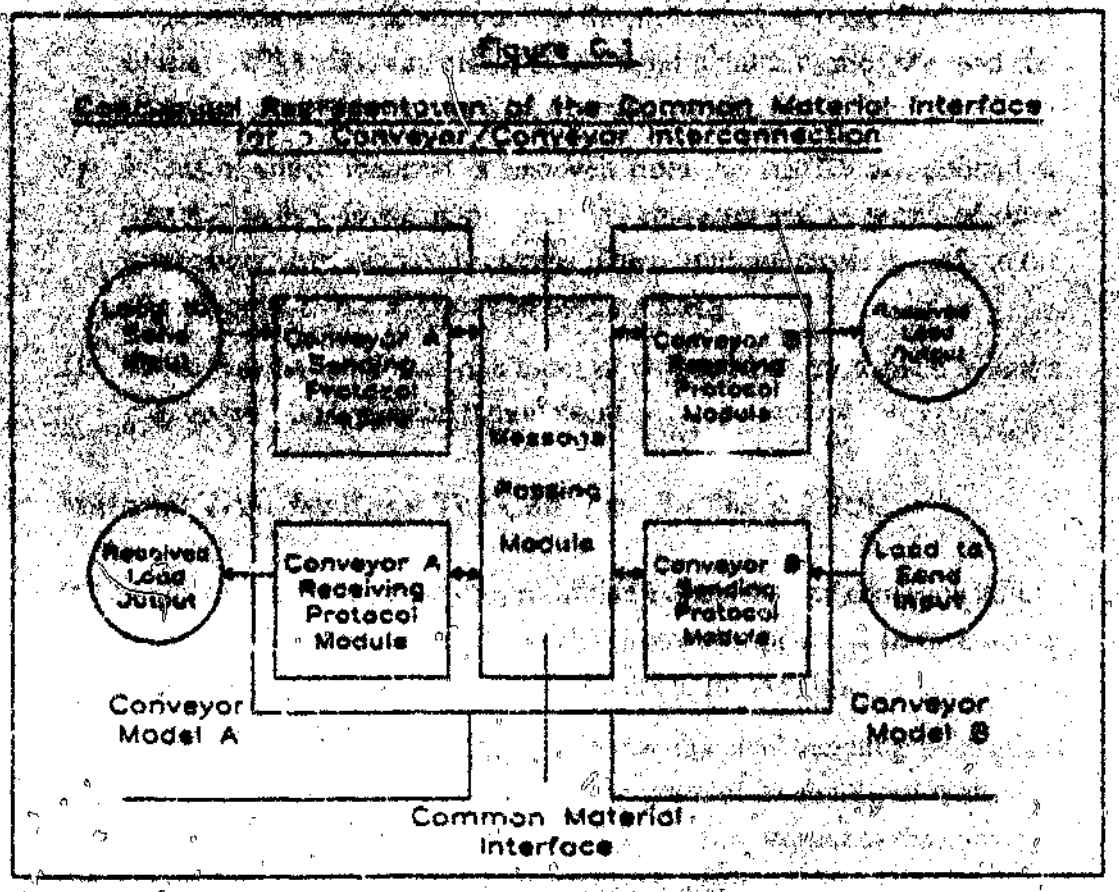
When the sending conveyor interface module receives a positive reply, load transfer is assumed to have occurred. The load is removed from the sending conveyor model. If the reply is negative the sending conveyor module deletes the load as it is assumed lost. In the case of the reply being negative with the fact that the load is not lost, then the following occurs:

The sending conveyor interface module will be informed by the sending conveyor model if material transfer is still being attempted. If this is the case, the above process will be repeated. If not, the conveyor model will perform the necessary operations on the material such as transporting the load away from the output point or it will remain stationary at the output point.

This completes the conveyor to conveyor, active-active interface protocol specification. The conceptual representation for the common material transfer interface in this case is shown in Figure C.1.

Figure C.1

Conceptual Representation of the Common Material Interface for a Conveyor/Conveyor Interconnection



C.2 The AS/RS/Conveyor Material Transfer Protocol Specification.

This specification describes the protocol for material transfer between AS/RS and conveyor models. This specification supports an active-passive interface where the AS/RS is the active entity in the interface.

Input/Output Point Definition.

The input/output point definitions for the conveyor model are shown in Section C.1, Input/Output Point Definition. The following are the definitions for the AS/RS.

1) Points at which material is received by the AS/RS storage/Retrieval (S/R) machine model are defined as "input" points. Each point is characterised in terms of three dimensional cartesian co-ordinates. The input point corresponds to the pick-up point of the pickup and deposit (P/D) station of the AS/RS. The dimension of the input point indicates the load size handling capabilities of the S/R machine model.

2) Points at which material is removed from the AS/RS are defined as "output" points. Each output point is characterised in terms of three dimensional cartesian co-ordinates. The output point corresponds to the deposit point of the P/D station of the AS/RS.

3) There may be more than one input or output points or a given location may be both an input and output point.

Material Transfer From The Conveyor To The AS/RS.

In this section the message passing, condition evaluation and condition failure action for the interface is described. The protocol supports a passive-active interface for transfer from the conveyor model to the AS/RS.

Material transfer is initiated when the S/R machine load handling mechanism has been activated and reaches the extreme point of its simulated motion to pick up a load. The AS/RS receiving module sends a request to the conveyor sending module showing the intent to pick up a load.

Condition Evaluation at the Conveyor.

When the conveyor sending module receives the request, it will send the load and all necessary information to the AS/RS receiving module if the following conditions are met:

A) There is a load to transfer to the AS/RS at the output point of the conveyor.

Condition Failure Action at the Conveyor.

If the above condition fails the AS/RS, receiving module is given a negative reply and the simulation system is informed. If the above condition is true, then a load is sent to the AS/RS receiving module with all necessary information for condition evaluation on the AS/RS side.

Condition Evaluation at the AS/RS.

If the AS/RS receives a negative reply from the conveyor, the transfer cycle at the receiving module is terminated. The S/R machine continues operation but without a load. If a load is received the load transfer is said to be successful if the following conditions are met:

A) The load has been located on the conveyor.

B) The S/R machine is not already carrying a load on the handling mechanism.

C) The S/R machine location corresponds to the output point of the sending conveyor.

Condition Failure Action at the AS/RS.

If any of the above conditions fail then the load is assumed lost to the simulation system and the simulation system is informed of the occurrence. If all conditions are met load transfer is successful and the S/R machine load handling mechanism (the shuttle) carries the load.

Material Transfer From The AS/RS To The Conveyor Model.

In this section the message passing, condition evaluation and condition failure action for transfer from the AS/RS to conveyor is described. The protocol supports an active-passive interface, where the AS/RS is the active entity.

Condition Evaluation at the AS/RS.

Material transfer is initiated by the AS/RS sending module when the load transfer mechanism has been activated and has reached the extreme point of its simulated motion in depositing a load. The AS/RS sending module will send a request to the conveyor receiving module for the conveyor to accept a load if the following conditions are fulfilled:

A) The S/R machine is carrying a load.

Condition Failure Action at the AS/RS.

If the above condition fails no load transfer is initiated and the simulation system is informed.

If the condition is met, then a request is sent to the conveyor receiving module asking it to receive a load. The load and all information necessary for condition evaluation is sent to the receiving module.

Condition Evaluation at the Conveyor.

When the conveyor receives the load, load transfer is said to be successful if the following conditions are met:

A) The input point co-ordinates correspond to the AS/RS input point co-ordinates.

B) The conveyor can handle the load.

C) The conveyor transportation mechanism is not in motion or is such that the load will be accepted onto the conveyor.

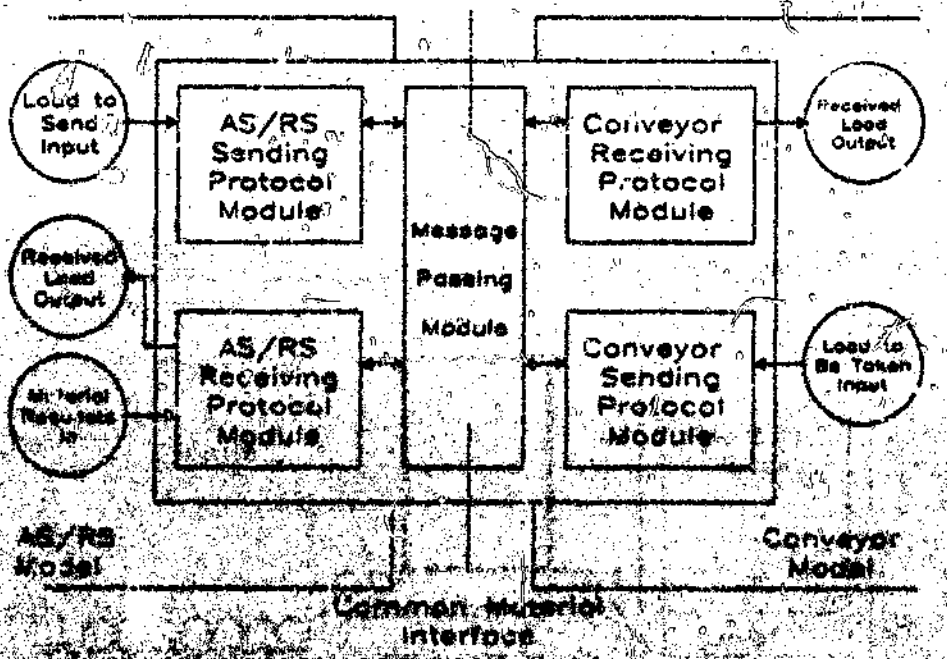
Condition Failure Action at the Conveyor.

If any one of the above conditions fail, the load is assumed lost to the simulation system. The simulation system is informed of the occurrence. If all conditions are met then the conveyor carries the load.

This completes AS/RS/conveyor interface protocol specification. The conceptual representation for the common material transfer interface in this case is represented in Figure C.2.

Figure C.2

Conceptual Representation of the Common Material Interface for an AS/RS/Coveyor Interconnection



A Practical Example Of The AS/RS/Conveyor Interface.

An example where this protocol might apply, is the UWtec (University of the Witwatersrand Technology Centre) AS/RS. The UWtec FMS, as described by Linn (1986), includes an AS/RS which is interfaced with a conveyor system. We present this interface as a practical example of where the protocol specification given above can be used to model this interface.

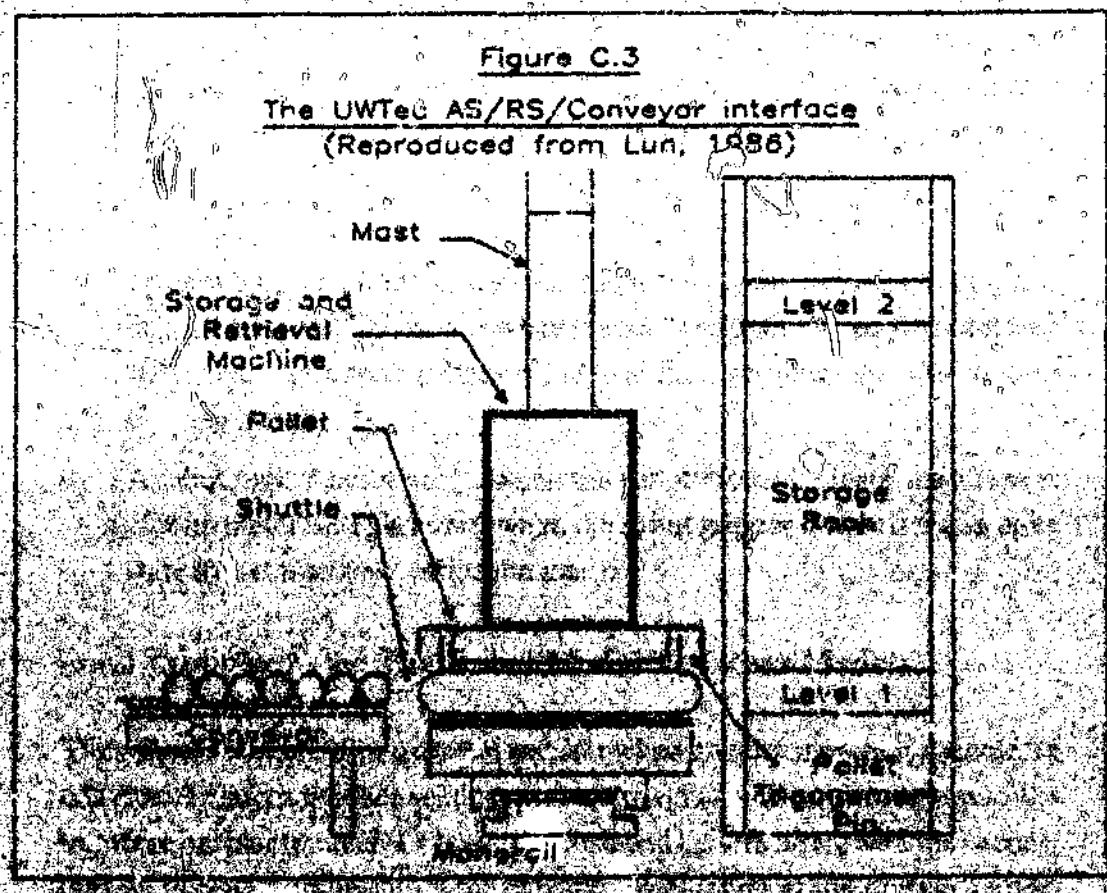
The interface structure is shown in Figure C.3. The shuttle was designed at UWtec. The load extraction method is by pins welded onto two chains which are driven by a small induction motor. The pins engage an appendage on the pallet. The pallet can thus be dragged onto the shuttle or pushed onto the conveyor depending on the direction of motion.

The AS/RS/... protocol specification can be used to model the UWtec equipment interface. The protocol will detect any inconsistencies relating to sensor operation and layout requirements of the physical interface.

Figure C.3

The UWTeC AS/RS/Conveyor interface

(Reproduced from Lun, 1986)



C.3 The Robot Model Material Transfer Protocol Specifications.

This specification describes the protocols used for material transfer between a robot model and other models. The robot sending and receiving sections of the protocol specification can be generalized for arbitrary partners in the interface as long as the interface exhibits an active-passive nature, where the robot is the active partner. The robot sending and receiving protocol is described and then a conveyor sending and receiving protocol, which interfaces with the robot is described.

Robot M[illegible]nput/Output Point Definition.

Since the robot material manipulating mechanism is the robot gripper which moves in the robot work-space, the robot model material interfaces do not occur at defined locations. The interface location can vary according to the robot program currently being executed. Therefore, it is necessary to be able to determine the robot model's intent for material transfer according to the robot program. Therefore, the robot model input/output points are defined as follows:

1) A robot "output" point occurs when the robot model indicates its intent to deposit material. This occurs when the robot gripper moves from a closed state to an open state (releases the load it is carrying) and the robot is carrying a load.

2) A robot "input" point occurs when the robot model indicates its intent to pick up material. This occurs when the robot gripper moves from an open state to a closed state (grips the load).

Load Transfer From The Robot Model To Other Models.

The robot model initiates material transfer when an output point is defined. In this case the robot model's sending protocol module will send a request to the interface partner to receive a load. The load will be sent along with the request plus all necessary information for condition evaluation at the partner's interface. Note that the request will only be sent if the robot has a load to deposit. This completes the robot sending interface protocol specification.

Load Transfer From Other Models To The Robot Model.

This is a passive-active interface hence the robot model initiates load transfer when an output point is defined. The robot model's receiving module will send a request to the interface partner indicating the robot's intent to pick up a load. The receiving module sends all the necessary information for condition evaluation at the interface partner.

The robot receiving interface module then waits for a reply. If the reply is negative, the robot model continues operation without the load and the simulation system is informed of the occurrence. This completes the robot receiving interface protocol specification.

Load Transfer From The Robot Model To A Conveyor Model.

The conveyor receiving module waits for a request from the robot sending module. When a request is received, it implies that the robot wishes to deposit a load on the conveyor. On receipt of the request, the conveyor receiving module must evaluate the following conditions for successful load transfer:

Condition Evaluation at the Conveyor.

For this material transfer, there are two cases for processing. Case A occurs if a pallet exists at the conveyor input point. Case B occurs if there is no material at the input point.

CASE A: Pallet at conveyor input point. In this case, load transfer is said to have occurred if the following conditions are met:

A) The conveyor input point corresponds with the robot output point.

B) The pallet on the conveyor has been located.

C) The load (part or pallet) from the robot can fit into the pallet on the conveyor.

D) The pallet on the conveyor is not full.

Condition Failure Action at the Conveyor For Case A.

A) If this condition fails the load is assumed lost to the simulation system. The real world analogy is that the gripper is at the wrong location and hence drops the load at an unknown location.

B) If this condition fails the load is assumed lost to the simulation system. The real world analogy is that the positions within the pallet where the incoming load is to be placed is not known.

Failure of conditions (C) and (D) implies a load loss. In all cases of condition failure, the simulation system is informed. If all conditions are fulfilled the load from the robot model will be placed in the pallet on the conveyor.

CASE B: No material at input point.

In this case, load transfer is said to have occurred if the following conditions are met:

A) The conveyor input point corresponds to the robot output point.

B) The load from the robot can be handled by the conveyor.

Condition Failure Action at the Conveyor for Case B.

If any one condition fails the load is assumed lost to the simulation system and the simulation system is informed. If all conditions are met the pallet is deposited on the conveyor. This completes the protocol specification of the conveyor receiving interface.

Load Transfer From A Conveyor Model To The Robot Model.

Condition Evaluation at the Conveyor.

The conveyor sending module waits for a request from the robot receiving module. When a request is received it indicates the robot model's intent to pick up a load from the conveyor. The conveyor sending module needs the dimension of the material that the modelled robot gripper can handle along with the request. When the request is received, the conveyor sending module must process three cases. Case 1 occurs if there is no material at the conveyor output point. Case 2 occurs if there is a pallet at the conveyor output point and case 3 occurs if there is a part at the conveyor output point.

CASE 1: No material at conveyor output point.

This is always the first case to be evaluated. If there is no load for transfer, the simulation system is informed and the robot receiving module is sent a negative reply. If there is a load then case 2 and 3 are evaluated.

CASE 2: Pallet at conveyor output point.

Within this case there are two further cases for evaluation. Case A occurs when the gripper can handle the pallet and case B occurs when the gripper cannot handle the pallet (the pallet is too big for the gripper).

CASE A: Gripper can handle pallet.

In this case load transfer is said to have occurred if the following conditions are met:

A) The robot input point corresponds with the conveyor output point.

B) The pallet on the conveyor has been located.

Condition Failure Action For Case A.

If any one condition fails, no load transfer is said to have occurred. The simulation system is informed and the robot receiving module is sent a negative reply. No material is assumed lost. If both conditions are met then the pallet is transferred to the robot.

CASE B: Gripper cannot handle pallet.

In this case load transfer is said to have occurred if the following conditions are met:

A) The robot input point corresponds with the conveyor output point.

B) The pallet has been located.

C) The pallet is not empty.

D) The robot gripper can handle the first load in the pallet material list.

Condition Failure Action For Case B.

If any one condition fails, then no load transfer occurs. The simulation system is informed and the robot module is sent a negative reply. No material is assumed lost.

If all conditions are fulfilled the first load (part or pallet) in the containing pallet's material list is transferred to the robot receiving module.

CASE 3: Part at conveyor input point.

Successful part transfer is said to have occurred if the following conditions are met:

A) The conveyor output point corresponds to the robot input point.

B) The part has been located.

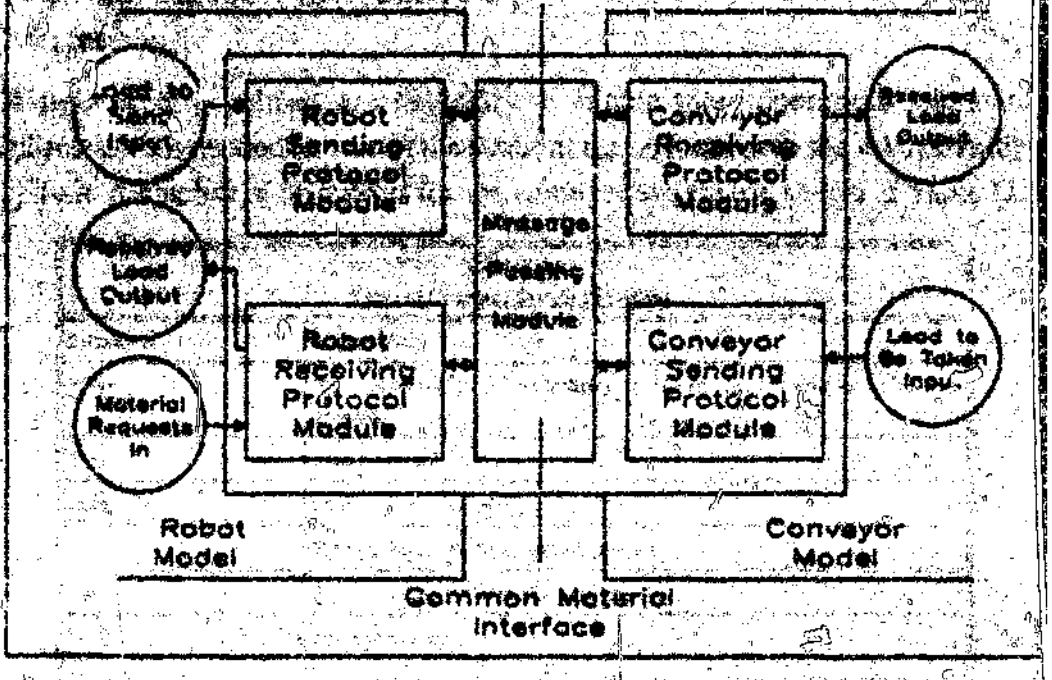
C) The robot gripper can handle the part.

Condition Action Failure For Case 3.

If any one of the above conditions fail, no part transfer is said to have occurred. The robot receiving module is sent a negative reply and the simulation system is informed. If all conditions are fulfilled then a part is transferred to the robot receiving interface module. This completes the protocol specification for the conveyor sending interface. The conceptual representation for the common material interface for the robot/conveyor interconnection is shown in Figure 4.

Figure C.4

Conceptual Representation of the Common Material Interface for a Robot/Conveyor Interconnection



```
        REPEAT Current := Current^.Next UNTIL Current^.Ready;
        Old^.Ready := FALSE;
        IF Current = Old
            THEN
                CheckTimeQueue
            ELSE
                TRANSFER(Old^.Routine, Current^.Routine)
        END
    END Deactivate;
(*------- Hoare's Approach -------*)
    PROCEDURE CPWait (VAR Monitor: MONITORS; VAR C: CONDITION;
                      P: CARDINAL);
    (* prioritised wait *)
    BEGIN
        AddToQueue(C, P);
        WITH Monitor^ DO
            IF Entry <> NIL
                THEN
                    Current^.Ready := FALSE; Reactivate(Entry)
                ELSE Occupied := FALSE; Deactivate
            END
        END
    END CPWait;

    PROCEDURE CWait (VAR Monitor: MONITORS; VAR C: CONDITION);
    (* wait on condition C in Monitor *)
    BEGIN
        CPWait(Monitor, C, DefaultPriority)
    END CWait;

    PROCEDURE CSignal (VAR Monitor: MONITORS; VAR C: CONDITION);
    (* signal waiting process at head of queue C in Monitor *)
    BEGIN
        IF C <> NIL THEN
            WITH Monitor^ DO
                (* place signaller at head of entry queue *)
                Current^.Queue := Entry; Entry := Current
            END;
            (* alternatively, AddToQueue(Monitor^.Entry,
            ReadyPriority); *)
            Current^.Ready := FALSE;
            (* signaller must release exclusion *)
            Release(C)
        END
    END CSignal;

    PROCEDURE CPriority (C: CONDITION) : CARDINAL;
    (* return priority of process at head of queue C if
    present *)
    BEGIN
        IF C = NIL
            THEN
                RETURN MaxInt
            ELSE
```

APPENDIX D: CONTROL INTERFACE PRIMITIVES AND STATUS INFORMATION

This appendix illustrates the PAS primitives or commands for the equipment types under discussion in this project report.

D.1 Conveyor Systems.

Conveyor systems support a number of operations which need to be control ed. These operations include:

Transportation.
Accumulation or buffering.
Sorting.
Collection and distribution.
Locating (positioning of material consistently within the tolerances dictated by automated machinery Eg: robots).
Part feeding.

A number of primitives are necess to initiate these operations and status information is needed to convey the status of the operations to the PAS operating system. A number of primitives for the conveyor system are shown in Table D.1.

TABLE D.1
Table of Conveyor System Primitives

Primitive	Explanation
FStart	Start conveyor in forward direction. RStart Start conveyor in reverse direction
Stop	Stop conveyor
Lock	Lock material in position
UnLock	Release material from locked position
Trap	Put up a trap to stop the material
UnTrap	Put down trap
Clamp	Combination of trap and clam lock, ie: stop the material and lock in position
UnClamp	Combination of unlock and clam untrap

```
            RETURN C^.Priority
        END
    END CPriority;
(*----  End of Howe's Approach ----  *)
    PROCEDURE WaitIO (Vector : CARDINAL);
    (* suspend caller waiting for I/O interrupt
    on machine dependent vector *)
    VAR
        Old             : ProcessState;
        NextProcess     : PROCESS;
        PR              : CARDINAL;
    BEGIN
        Old := Current;
        REPEAT Current := Current^.Next
        UNTIL Current^.Ready;
        Old^.Ready := FALSE; (* deactivate pending interrupt *)
        IF Current <> Old (* check potential deadlock *)
            THEN
                NextProcess := Current^.Routine;
            ELSIF Queue(Hold)
                THEN
                    (* prepare to activate those waiting *)
                    Current := Hold; NextProcess := Current^.Routine;
                    WITH Current^ DO
                        Hold := Queue; Ready := TRUE;
                        Queue := NIL; Priority := 0;
                    END;
                ELSE (* activate idle process *)
                    IdleProcess^.Ready := TRUE;
                    NextProcess := IdleProcess^.Routine
        END;
        TRANSFER(Old^.Routine, NextProcess, Vector);
        IdleProcess^.Ready := FALSE; Old^.Ready := TRUE;
        TRANSFER(Old^.Routine, NextProcess);
    END WaitIO;

    PROCEDURE CreateCondition (VAR C: CONDITION);
    (* compulsory initialization of condition queue *)
    BEGIN
        C := Nil
    END CreateCondition;

    PROCEDURE CreateMonitor (VAR Monitor : MONITOR);
    (* compulsory initialization of Monitor *)
    BEGIN
        Allocate(Monitor, TSIZE(Exclusion));
        WITH Monitor^ DO Occupied := FALSE; Entry := NIL;
            WaitingProcess := NIL
        END
    END CreateMonitor;
```

Status information would indicate the if the system is busy executing a command or when the command has been executed, Eg: PalletClamped, which indicates the completion of a clamping operation.

D.2 Automated Storage and Retrieval Systems.

The primitives used to control the AS/RS system with respect to the AS/RS functions are shown in Table D.2.

TABLE D.2
Table of AS/RS [illegible]
(Compiled from [illegible])

Primitive	Explanation
[illegible]	[illegible] the S/R machine [illegible]
NormalSpeed	velocity to run the S/R machine
SlowSpeed	[illegible] the S/R machine
Initialise	to initialise the S/R machine's [illegible] X and Y [illegible]
GotoXY	[illegible]
GetConveyor	[illegible]
PutConveyor	to remove a pallet from [illegible] put it on the conveyor
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	
[illegible]	GotoXY [illegible] [illegible] [illegible] [illegible] [illegible] [illegible]
[illegible]	GotoXY [illegible] [illegible] [illegible] GotoXY [illegible] [illegible]
[illegible]	([illegible] slot with pallet) GotoXY (slot containing pallet) GetPallet GotoXY [illegible] PutConveyor
RetrieveProduct	(remember slot with required product quantity) GotoXY (slot containing product) GetPallet GotoXY (head of conveyor) PutConveyor
RetrieveSlot	GotoXY (slot as specified) GetPallet GotoXY (head of conveyor) PutConveyor

```
PROCEDURE EnterMonitor (VAR Monitor : MONITORS);
(* called by all procedures in Monitor on entry *)
BEGIN
  WITH Monitor^ DO
    IF Occupied
      THEN
        AddToQueue(Entry, EntryPriority); Deactivate
      ELSE Occupied := TRUE
    END
  END
END EnterMonitor;

PROCEDURE ExitMonitor (VAR Monitor : MONITORS);
(* called by all procedures in monitors on exit *)
BEGIN
  WITH Monitor^ DO
    IF Entry <> NIL
      THEN
        Reactivate(Entry)
        (* but leave this process ready *)
      ELSE Occupied := FALSE
    END
  END
END ExitMonitor;

PROCEDURE Time () : CARDINAL;
BEGIN
  RETURN PseudoTime
END Time;

PROCEDURE Delay (T : CARDINAL);
(* suspend caller on global clock queue, to be
reactivated when pseudotime has advanced T units *)
VAR
  Alarm : CARDINAL;
BEGIN
  Alarm := PseudoTime + T;
  Queue(ClockQueue, Next, Alarm);
  Passive.Me := Alarm (* when routine advances *)
END Delay;

PROCEDURE COBEGIN;
VAR
  I : CARDINAL;
BEGIN
  IF NOT OneStart
    THEN
      Terminal.WriteString('Missing  COBEGIN'); Quit
    ELSIF NProcs > 0 THEN
      Main^.Ready := FALSE; Current := Main^.Next;
      TRANSFER(Main^.Routine, Current^.Routine);
      VisibleClock; OneStart := FALSE;
      (* after its all over, resume *)
  END;
END COEND;
```

These primitives are used at the UWTec FMS and are described in detail by Lun (1986). The UWTec AS/RS is interfaced with a conveyor system and hence a number of these primitives apply for this particular case. For real-time decision making, the status information might include whether the S/R machine is busy executing a command, responses on completion of a command or failure of operations as monitored by the AS/RS controller.

D.3 Automated Guided Vehicle Systems.

In the case of the AGVS, only inductively guided systems are considered. The trucks of the AGVS have on board control systems which can perform the following operations (Muller, 1983):

1) Steering control which keeps the truck on course.
2) Route control which brings the truck through the network to the predetermined destination. This function can be performed by a control computer outside the truck.
3) Drive control which first starts up the truck motor and then accelerates, brakes and stops the vehicle as a function of the traffic situation.
4) Load transfer control, which in the case of an automated transfer mechanism controls loading and unloading of the truck.

Assuming the above functions are on-board the truck, the primitives shown in Table D.3 are possible. Status information can include the truck identification, if the truck is busy, the completion of a command and the current location of the truck.

D.4 Robots or CNC Machine Primitives.

Robots and CNC machines operate according to a program which defines the particular operations that the equipment is to perform in the context of the cell operation. Robot or CNC machines must interface with other equipment types during operation execution. A means of sequencing the equipment is necessary and hence primitives are required. The primitives and status information needed in the case of CNC machines or robots will depend on the operation of the

```
PROCEDURE GetProcess (VAR P : PROC);
BEGIN
    P := Current^.Process
END GetProcess;
(* ------  ------        Kernel ----------*)
BEGIN
    NProcs := 0;
    PseudoTime := 0;
    CurrentID := 'a';
END Kernel;
(* ----------    End of Kernel ----------      *)
PROCEDURE Task;
(* all processes create an instance of this as a
coroutine to allow automatic addition to StopProcess *)
VAR
    ActualProcess : PROC;
BEGIN
    GetProcess(ActualProcess);   ActualProcess;
    StopProcess;
END Task;
PROCEDURE StartProcess (P: PROC; N: CARDINAL;
                        Parameter: ADDRESS);
(* Initiate a user process with procedure P, workspace N,
parameter at Parameter *)
VAR
    PR              : CARDINAL;
    NewProc         : ProcPointer;
    WorkSpace       : ADDRESS;
BEGIN
    IF NOT CoStart
        THEN
            TransferToNewProcess(ToKernel   COROUTINE); Quit
    END;
    Allocate(WorkSpace, N);
    Allocate(NewProc, TSIZE(ProcessDescriptor));
    WITH NewProc^ DO (* link into ring *)
        Ready := TRUE; Status := NIL;
        Priority := 0; Process := Parameter;
        Work := WorkSpace; Area := N;
        CreatedByMain := Main; ConditionList := NIL;
    END;
    NEWPROCESS (Task, WorkSpace, N, NewProc^.Routine);
    LinkProcess(NewProc, P);
END StartProcess;
PROCEDURE COROGIN;
BEGIN
    CoStart := TRUE
END COROGIN;
```

equipment types interfaced together. These primitives and the status message requirements will be application dependent and hence we do not attempt to show a set of primitives for this case.

TABLE D.3
Table of AGVS Primitives.
(Compiled from Muller, 1983)

Primitive:	Explanation
GoDestination	Goto a specified destination
PutLoadLeft	Initiate transfer mechanism to drop off load atdestination on the left hand side
PutLoadRight	Initiate transfer mechanism to drop off load atdestination on the right hand side
GetLoadLeft	Initiate transfer mechanism to pick up load atdestination from the left hand side
GetLoadRight	Initiate transfer mechanism to pick up load atdestination from the right hand side
SendStatus	Commands the truck to identify itself and send its current location and whether it is busy or not

```
PROCEDURE IDLE;
(* only activate when all else has been held up *)
BEGIN
   LOOP
      WriteString(' IDLE '); (* Debugging *)
   END
END IDLE;

(* ---------- MonitorKernel ---------- *)

BEGIN
   Allocate(Current, TSIZE(ProcessDescriptor));
   WITH Current^ DO
      ID := '*'; Next := Current; Ready := TRUE; Queue := NIL;
   END;
   Main := Current;
   CanStart := TRUE;
   StartProcess(IDLE, 399, NIL);
   CanStart := FALSE;
   IdleProcess := Current^.Next;
   IdleProcess^.Ready := FALSE;
   CreateMonitor(Simulation);
   CreateCondition(Hold);
   InitProcedure(InitClock);
   TermProcedure(TermClock);
END MonitorKernel.

(* ---------- End of MonitorKernel ---------- *)

IMPLEMENTATION MODULE [illegible]Storage[illegible];
(* [illegible] storage allocation, interrupt free.
P.D. Terry [illegible] *)
FROM SYSTEM IMPORT ADDRESS, TSIZE;
IMPORT Storage, Terminal;
(* EXPORT QUALIFIED Allocate, Deallocate, Available; *)
TYPE
   HEAP = POINTER TO HEAPSLOTS;
   BLOCKS = POINTER TO BLOCKSLOTS;
   HEAPSLOTS = RECORD
                  [illegible] : CARDINAL;
                  Next : HEAP;
                  [illegible] : BLOCKS;
   END; (* HEAPSLOTS *)
BLOCKSLOTS = RECORD
                  Adr : ADDRESS;
                  Next : BLOCKS;
   END; (* BLOCKSLOTS *)

VAR
   HeapTop : HEAP;
```

APPENDIX E: THE CONVEYOR TRANSPORT MODEL IMPLEMENTATION

This appendix discusses the development of the transport model's internal module structure. In the discussion, the major aspects needed to develop the structure, as discussed in Chapter 6, are presented along with an example of model operation.

E.1 The Transport Model Logical Model.

Conveyor systems not only vary according to their types, Eg: monorail, powered roller or chain conveyors etc, but according to their operating characteristics. This includes conveyor speed(s), length, width and placement on the shop floor. Sensors may also be placed on the conveyor for material counting or identification, for example. The logical model must take all these into account in providing the specific type and behaviour of the conveyor transport functions that the user requires.

E.2 Mathematical Models.

For the purpose of presenting an example of model operation it is assumed that material on the conveyor transport model will move a distance x in a time t at a constant velocity v where:

$$x = vt$$

E.3 Transport Model Description.

This section presents the transport model equipment states, a structure for representing the model states and discusses a model structure description.

```
PROCEDURE Allocate (VAR X: ADDRESS; Size: CARDINAL);
VAR
   Current, NewHeap : HEAP;
BEGIN
   IF Size = 0
      THEN
         TerminalWriteString('   Error in Allocation ');
         HALT
   END;
   Current := HeapTop;
   WHILE (Current <> NIL) AND (Size <> Current^.BlockSize) DO
      Current := Current^.Next
   END;
   IF Current = NIL
      THEN (* request for a new size of block *)
         Storage.ALLOCATE(X,  Size);
         Storage.ALLOCATE(NewHeap,  TSIZE(HEAPSLOTS));
         WITH NewHeap^ DO
            BlockSize := Size; Next := HeapTop; Free := NIL;
         END;
         HeapTop := NewHeap
      ELSE
         WITH Current^ DO
            IF Free <> NIL
               THEN (* can reuse old slots *)
                  X := Free^.Adr;  Free := Free^.Next
               ELSE (* must use new slot of known size *)
                  Storage.ALLOCATE(X,  Size)
            END
         END
   END
END Allocate;

PROCEDURE Deallocate (VAR X: ADDRESS; Size: CARDINAL);
VAR
   NewBlock     : BLOCKS;
   Current      : HEAP;
BEGIN
   Current := HeapTop;
   WHILE (Current <> NIL) AND (Size <> Current^.BlockSize) DO
      Current := Current^.Next
   END;
   IF (X = NIL) OR (Current = NIL)
      THEN TerminalWriteString('   Error in deallocation ');
         HALT
      ELSE
         Storage.ALLOCATE(NewBlock,  TSIZE(BLOCKSLOTS));
         WITH NewBlock^ DO
            Adr := X; Next := Current^.Free
         END;
         Current^.Free := NewBlock
```

1) Transport model equipment states:

The transport model is described by three operational states. These are:
On_Forward.
On_Reverse.
Stop.

2) Model state:

The model state is described in terms of a material list which includes all material "on" the transport model and a number of attributes associated with the material. This is the location of each material unit on the conveyor referenced to a convenient point on the conveyor and whether material is blocked or not. The material "blocked" attribute is needed for taking into account the case where material cannot leave the conveyor (if the receiving building block model cannot accept the material). The list also includes the next event associated with the material. Figure E.1 illustrates the material list structure.

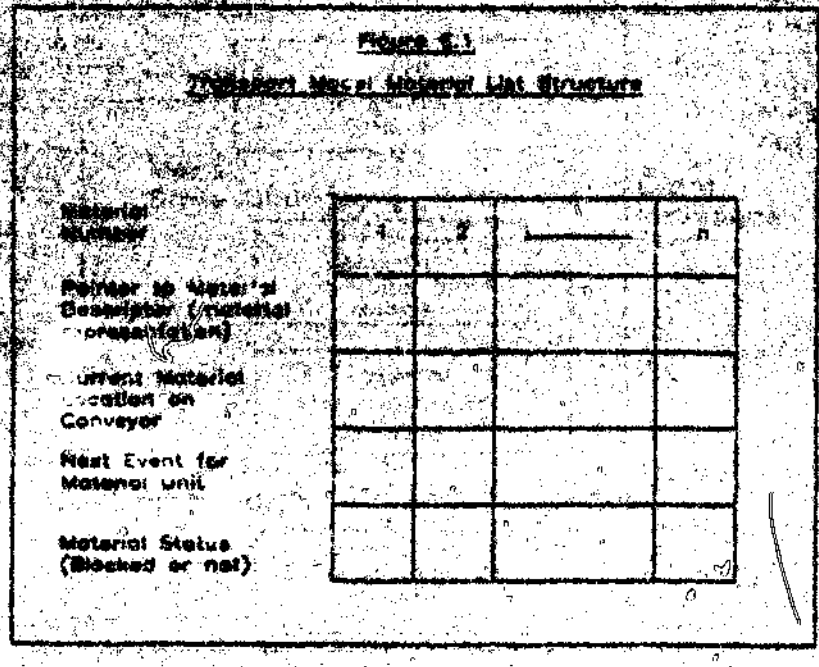


Figure E.1
Transport Model Material List Structure

```
        END;
        X := NIL;
    END Deallocate;

    PROCEDURE Available (Size: CARDINAL) : BOOLEAN;
    VAR
        Current : HEAP;
    BEGIN
        Current := HeapTop;
        WHILE (Current <> NIL) AND (Size > Current^.BlockSize) DO
            Current := Current^.Next
        END;
        IF Current = NIL
            THEN (* request for a new size of block *)
                RETURN Storage.Available(Size)
            ELSE
                IF Current^.Free <> NIL
                    THEN (* can reuse old slot *)
                        RETURN TRUE
                    ELSE (* must use a new slot of known
                        RETURN Storage.Available(Size)
                END
        END
    END Available;
(* ------------ SafeStorage ------------ *)
    BEGIN
        HeapTop := NIL
    END SafeStorage.
IMPLEMENTATION MODULE SystemTime[7];
(* module which manages getting system time to the
modules in the simulation *)
(* EXPORT QUALIFIED GetSystemTime, Seconds; *)
FROM MonitorKernel IMPORT GlobalTime, TimeRecord;
    PROCEDURE GetSystemTime(VAR TimeInSeconds : Seconds);
    (* procedure which calculates the time in seconds
    since the kernel has been activated. The procedure
    is used by modules in the simulation to get system time *)
    CONST
        MaxTimeInCardinal = 65535.0;
    VAR
        TempUnits, TempData : CARDINAL;
        TempSeconds : Seconds;
    BEGIN
        TempSeconds := FLOAT(GlobalTime.Ticks)/18.2;
        (* approx 18.2 clock ticks per second *)
        TimeInSeconds := (FLOAT(GlobalTime.Units))*
        MaxTimeInCardinal/18.2 + TempSeconds;
    END GetSystemTime;
END SystemTime.
```

3) Model structure description.

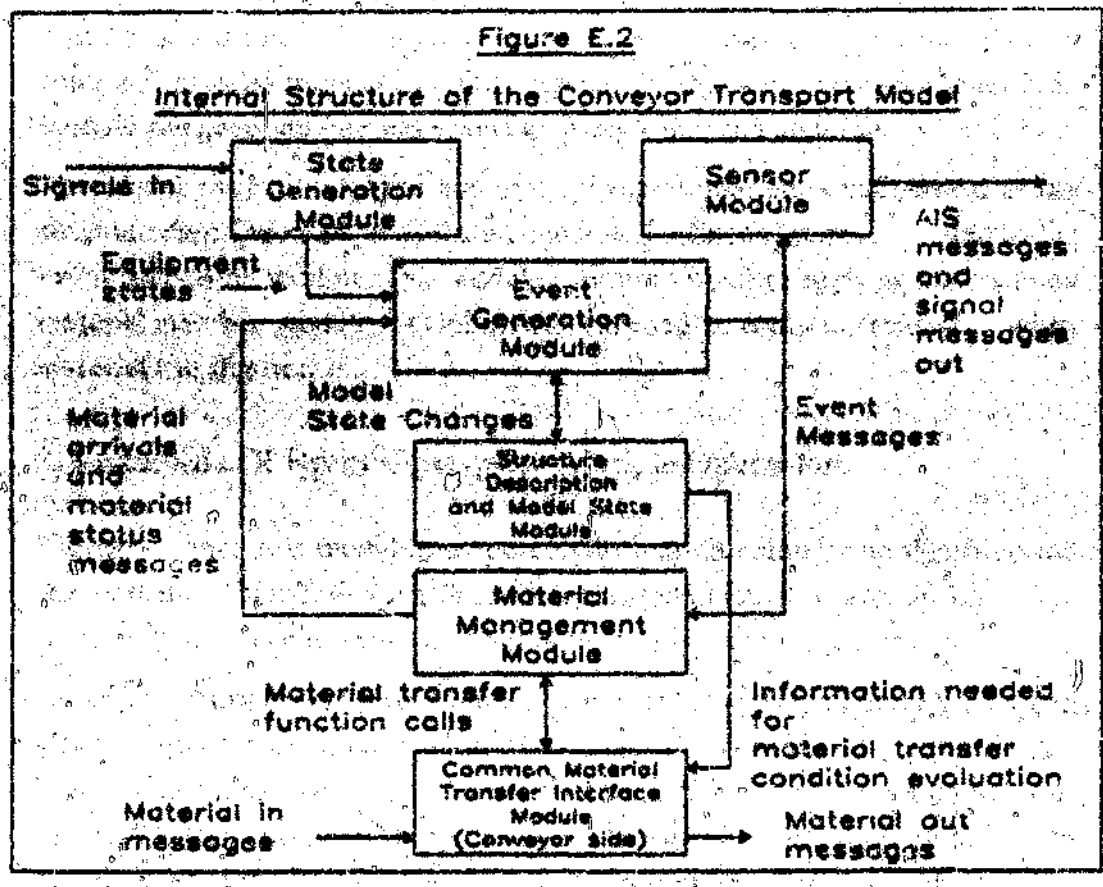
The model structure description is a convenient means of representing the transport model's physical structure. This includes the logical conveyor length, the material arrival point or the material departure point and sensor placement along the conveyor logical length.

E.4 Relating Inputs To Outputs.

The relation between inputs and outputs is best illustrated in terms of the internal module structure of the conveyor transport building block model. This is shown in Figure E.2. The modules are described.

Figure E.2

Internal Structure of the Conveyor Transport Model



The state generation module translates incoming signals into the appropriate equipment states. The sensor module contains models of all the sensor units that the user has placed along the logical conveyor length. Event messages which

REFERENCES

1. Barash, M. (1986), Manufacturing Systems Today And Tomorrow, In: Editorial, *The International Journal of Advanced Manufacturing Technology*, Vol. 1, No. 2, pp.1-3.

2. Bloch, G. (1985), Mechatronics And The Factory Of The Future, *Proceedings of the 1985 Production and Productivity Management Convention*, Johannesburg.

3. Bloch, G. (1986), Flexible Software For Flexible Automation, *Sarocim, Proceedings of the First South African Symposium on Robotics and Computer Integrated Manufacture*, Randburg, 26-27 March, 1986.

4. Booch, G. (1983), *Software Engineering With ADA*, Benjamin/Cummings Publishing Company, Inc. Canada.

5. Booch, G. (1986), *Object-Orientated Development*, IEEE Transactions on Software Engineering, Vol. SE-12, No. 2, pp.211 -221.

6. Brandolese, A. and Garetti, M. (1983), FMS Control Systems: Design Criteria And Performance Analysis, *Proceedings of the 2nd International Conference on Flexible Manufacturing Systems*, London, U.K. pp.365-380.

7. Browne, J. and Rathmill, K. (1983), The Use Of Simulation Modelling As A Design Tool For FMS, *Proceedings of the 2nd International Conference on Flexible Manufacturing Systems*, London, U.K.

8. Buddill, E.J. (1984), Unit Load AS/RS Capabilities and Design Considerations, *Papers on Automated Material Handling and Storage*, Auerbach Publishers, Inc.

9. Carson, G.B., Bolz, H.A. and Young, H.H. (1972), *Production Handbook*, Third Edition, Wiley and Sons, Inc. New York.

10. Cheng, T.C.E. (December 1985), Simulation Of Flexible Manufacturing System, *Simulation*, pp.299-302

11. De Almeida, A.R.C. and MacLeod, I.M. (1985), A Proposed Architecture For Distributed Dynamic Simulation And Control, *Proceedings 3rd SACAC Symposium on Control Theory and Applications.*

indicate the activation of sensors are translated by the sensor module into the appropriate signal or AIS output messages. The material management unit informs the event generation module of material arrivals and, in the case of material being blocked, whether or not material has been transported off the conveyor via the common material transfer interface.

The event generation module receives material which arrives and adds this to the material list. This module generates the next event for each material unit using the model state and the model structure description maintained in the structure description and model state module. The next event is sensor activations or material transfers off the conveyor. If material becomes blocked at the conveyor output point, no material transfer can take place and material accumulates at the output point. If material is blocked, the material management module informs the event generation module, based on the management modules interaction with the common material transfer interface module. In this way, the event generation module can generate internal events to manage blocked material.

The model structure and the material list representation is maintained in the structure description and model state module. The model state is updated by the event generation module. This module is responsible for supplying the common material interface module with information needed for condition evaluation when material transfer occurs.

E.5 Example Of Event Generation Module Operation.

The event generation module incorporates the real-time simulation algorithm and the mathematical model. An example is presented which illustrates the operation of this module since it forms the basis of the transport model operation.

Consider a transport model configured by the user as shown in Figure E.3. S1 and S2 are sensors which are activated when material passes by. All events are referenced to the conveyor transport model input point which has conveyor co-ordinate 0. The conveyor co-ordinates are related to the shop floor global co-ordinates as shown. For this example, a constant transport speed of 1 m/s has been selected.

12. De Almeida, A.R.C. MacLeod, I.M. and Ypma, T.J. (1985), Requirements For Distributed Real-Time Simulation Of Large Scale Systems, *Proceedings of Control 85*, Cambridge, pp.336-340.

13. Diesch, K.H. and Malstrom, E.M. (June 1985), Physical Simulator Analyzes Performance Of Flexible Manufacturing Systems, *Industrial Engineering*, pp.66-75.

14. Eary, D.F. and Johnson, G.E. (1962), *Process Engineering for Manufacturing*, Prentice-Hall, Inc. Englewood Cliffs, N.J.

15. Engelberger, J.F. (1983), *Robotics in Practice*, Kogan Page, London.

16. Engelke, H., Grotrian, J., Scheuing, C., Schmackpfeffer, A., Schwarz, W., Solf, B., and Tomann, J. (1985), Integrated Manufacturing Modelling System, *IBM Journal of Research and Development*, Vol. 29, No. 4, pp.343-355.

17. Feltner, C.E. and Weiner, S.A. (July 1985), Models, Myths And Mysteries In Manufacturing, *Industrial Engineering*, pp.66-76.

18. Ford, G.A. and Wiener, R.S.(1985), *Modula-2: A Software Development Approach*, John Wiley and Sons, Inc. Canada.

19. Grant, J.W. and Weiner, S.A. (August 1986) Factors To Consider In Choosing A Graphically Animated Simulation System, *Industrial Engineering*, pp.37.

20. Groover, M.P. and Wiginton, J.C. (January 1986), CIM And The Flexible Automated Factory Of The Future, *Industrial Engineering*, pp.73-85.

21. Hawker, J.S., Nagel, R.N., Roberts, R. and Olney, N.G. (1986), Multiple Robotic Manipulators, *Byte*, Vol. 11, No. 1, pp.203-219.

22. Hill, J.M. (1985), Flexible Material Management Systems, *Proceedings 5th International Conference on Automation in Warehousing*, Stockholm, Sweden.

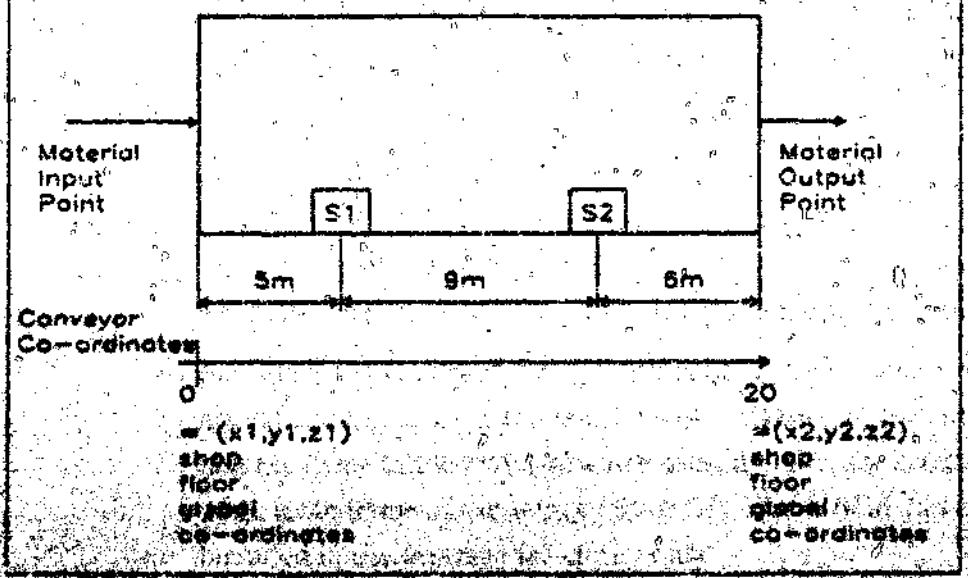
23. Johns, A.E. (1986), Why Simulate?, In Seminar: *Simulation-A Tool for Manufacturing Management*, The Institution of Production Engineers, Knightsbridge, London.

24. Kay, J.M. and Walmsley, A.J. (1982), Computer Aids For The Optimal Design Of Operationally Effective FMS, In: *Proceedings of the 1st International Conference on Flexible Manufacturing Systems*, Brighton, U.K. pp.463-480.

```
(*--------      Module CLOCK --------*)
MODULE CLOCK [ClockPriority];  (*--IBM  PC dependant--*)
IMPORT     NEWPROCESS, TRANSFER, IOTRANSFER, PROCESS, ADR, SIZE,
           WORD, ADDRESS, GETREG, SETREG, SWI, AX, ES, BX,
           WriteLn, Terminal, SetDeviceStatus, GetDeviceStatus,
           SysInterruptVector, RestoreInterruptVector,
           InstallHandler, UninstallHandler, WriteString,
           Current, MaxCard, GlobalTime;
EXPORT   InitClock, TermClock, SetClock,
         DisableClock, Disable;
  CONST
     ClockIVector    = 08H; (*--IBM  PC dependant--*)
     ClkConst        = 1;
     ClockDevice     = 0; (*--Clock  device number--*)
     Disable         = FALSE;
     Enable          = TRUE;
  VAR
     OldInterruptVector     : ADDRESS;
     OldClockDeviceStatus   : BOOLEAN;
     Activated              : BOOLEAN;
     CLKWSP                 : ARRAY [0..499] OF WORD;
     (* process workspace *)
     ClkPeriod,
     ClkTick                : CARDINAL;
     (* count for desired time slice *)
     UserProcess,
     ClockProcess           : PROCESS;
MODULE CheckCriticalSection(?);
(* module which sets up a pointer to an address
which holds a value that indicates if DOS is
in a critical section *)
IMPORT ADR, ADDRESS, SETREG, GETREG, SWI, AX, ES, BX;
EXPORT IsCriticalSection;
  TYPE
     BooleanPointer = POINTER TO BOOLEAN;
  VAR
     IsCriticalSection : BooleanPointer;
     ans               : ADDRESS;
  BEGIN
     SETREG(AX, 3400H);
     SWI(21H);
     GETREG(ES, ans.SEGMENT);
     GETREG(BX, ans.OFFSET);
     IsCriticalSection := BooleanPointer(ans);
  END CheckCriticalSection;
```

Figure E.3

Transport Model Configuration Example



Message Representation.

In order to show how the event generation module operates, its interface is defined in terms of input and output messages as follows:

1) Input Messages.

The input messages to the event generation module are defined as follows:

(I) For material arrival, the message is M(t,m,s). M denotes that the message is associated with material arrival. t is the real-time that the LP received the message, m is the material name and s is the material state (blocked or not). This message contains all the information needed from the material management unit.

25. Lennox, K.M. Thompson, B. and Tshwele, M.E. (1986), *Modelling of Physical Units Found in the Process Control Industries*, 4th Year Design, University of the Witwatersrand Publication, Johannesburg, 1986.

26. Lenz, J.E. (February 1983), MAST: A Simulation Tool For Designing Computerized Metalworking Factories, *Simulation*, pp.51-58.

27. *Logitech Modula-2/86 User's Manual*, Third Edition, March 1986.

28. Lun, V. (1986) *A Software Kernel for an Automated Storage and Retrieval System*, Master of Science Dissertation, University of the Witwatersrand Publication.

29. MacLeod, I.M. (1986) Real-Time Programming in Computer Integrated Manufacturing Systems. Sarocim, *Proceedings of the First South African Symposium on Robotics and Computer Integrated Manufacture*, Randburg, 26-27 March 1986.

30. Maimra, R., Giuffre, O. and Cavagnaro, F. (1982), A Generalised Approach To The Problem Of FMS On-Line Management, In: *Proceedings of the 1st International Conference on Flexible Manufacturing Systems*, Brighton, U.K.

31. Mills, R.J. (1983) Computer Simulation-A Feasibility And Planning Tool For FMS, In: *Proceedings of the 2nd International Conference on Flexible Manufacturing Systems*, London, U.K.

32. Mize, J. (1984), *Distributed Simulation*, Inc Monograph, Department of Computer Science, University of Texas, Austin.

33. Morris, H.M. (February 1984), Computerized Numerical Control Evolves In Response to Market's Changing Needs, *Control Engineering*, pp.76-79.

34. Müller, E.T. (1983), *Automated Guided Vehicles*, IFS (Publications) Ltd, U.K.

35. Nof, S.Y., Moler, W.L. and Diessenroth, M.P. (October 1980), Computerized Physical Simulators Are Developed To Solve IE Problems, *Industrial Engineering*, pp.70-75.

36. Productivity International Inc. (1983), *A Survey of Indutrial Robots*, Second Edition, Leading Edge Publishing, Inc. Dallas , Texas.

37. Rahnejat, H. (1986), Simulation Aids Design For Flexible Automation, *The International Journal of Advanced Manufacturing Technology*, Vol. 1, No.2, pp.91-108.

```
PROCEDURE SetClock (Count : CARDINAL);
BEGIN
        ClkPeriod := Count
END SetClock;

PROCEDURE DisableClock;
BEGIN
        SetDeviceStatus(ClockDevice,    Disable);
END DisableClock;

PROCEDURE ClockHandler;
(* clock interrupt handler that gets the next ready
process *)
BEGIN
    InstallHandler(ClockProcess,    Clock1Vector);
    (* associate clock handler permanently to the clock
    interrupt vector *)
    LOOP
        IOTRANSFER(ClockProcess,    UserProcess, Clock1Vector);
        Current^.Routine    := UserProcess;
        INC(ClkTick);
        WITH GlobalTime DO
            INC(Ticks);
            IF Ticks = MaxCard THEN
                Ticks := 0;
                INC(Units);
            END;
        END; (* with *)
        IF (ClkTick    = ClkPeriod ) THEN
            ClkTick := 0;
            IF NOT IsCriticalSection^    THEN
                REPEAT Current := Current^.Next;
                UNTIL Current^.Ready;
                UserProcess := Current^.Routine;
            END;
        END;
    END (* loop *)
END ClockHandler;

PROCEDURE InitClock;
(* institute a clock interrupt handler process *)
BEGIN
    IF NOT Activated THEN
        SaveInterruptVector(Clock1Vector,    OldInterruptVector);
        (* save old interrupt interrupt vector *)
        GetDeviceStatus(ClockDevice,    OldClockDeviceStatus);
        (* save old device status
        (interrupts Enabled/Disabled) *)
        Activated := TRUE;
        NEWPROCESS (ClockHandler, ADR(CLKWSP),  SIZE(CLKWSP),
                    ClockProcess);
        (* create the Module-3 process for the clock handler *)
        SetDeviceStatus(ClockDevice,    Disable);
        (* disable clock interrupts *)
        TRANSFER (UserProcess, ClockProcess);
```

(ii) For equipment state inputs, the message is E(t,s). E denotes a message from the equipment state generation module. t is the real-time of arrival of the message and s is the equipment state (On_Forward, On_Reverse, Stop).

2) Output Messages.

The output messages of the event generation module are defined as follows:

(i) For material transfer off the transport model, the message is MO(t,m). MO denotes a material output message. t is the real-time of material transfer and m is the name of the material which is to be transferred off the conveyor transport model.

(ii) For sensor activations, the message is S(t,m,s). S denotes a sensor activation message. t is the real-time of activation of the sensor, m is the name of the material which caused the sensor activation and s is the name of the sensor (S1 or S2). The m field has been included in the message for clarity only.

Real-Time Inputs.

For the purposes of the example, the inputs and their arrival times at the transport model are shown in Table E.1. Input messages are made up of material arrivals and signal messages (in terms of equipment states for the example) for model control.

Event Generation Module Input/Output Sequence.

The operation of the event generation module for the input messages, as specified in Table E.1, is shown in tabular form in Table E.2. This table shows how the input messages are used to generate output messages using the real-time simulation algorithm shown in Section 6.3.3.

The Messages Predicted And Not Sent are stored in the next event field of the material list. These messages are sent at their predicted times as long as they have not been cancelled previously. The Messages Sent column in Table E.2

38. Ranky, P.G. (1983), *The Design And Operation Of FMS*, IFS (Publications) Ltd. North Holland Publishing Company, U.K.

39. Rathmill, K., Greenwood, N. and Houshmand, M. (1983), Computer Simulation Of FMS, In: *Proceedings of the 2nd International Conference on Flexible Manufacturing Systems*, London, U.K.

40. Shapiro, S.F. (1987), Implementing CIM: Can the Industry Wait Much Longer?, *Computer Design*, July 1987 pp.53-73

41. Thomas, R. (1983a), Programming Expands Limits of Robot Controllers, *Industrial Engineering*, April 1983, pp.34-40.

42. Thomas, R. (1983b), Designing Controls for Robotic Workcells, *Industrial Engineering*, May 1983, pp.34-39

43. Tompkins, J.A. and Wallace, R.J. (November 1980), Interfacing Material Handling With The Automated Factory, *Industrial Engineering*, pp.62-65.

44. Tompkins, J.A. and White, J.A. (1984), *Facilities Planning*, John Wiley and Sons, Inc. New York.

45. Terry, P.D. (1986), A Modula-2 Kernel for Supporting Monitors, *Software Practice and Experience*, Vol. 16(5), pp.457-472.

46. Welb, T.L. (1983), Evaluating Robotic Systems Through Simulation, In Conference Proceedings: *13th International Symposium on Industrial Robots and Robots 7*, Chicago, Illinois, Vol. 1.

```
        (* transfer control to the clock handler *)
        SetDeviceStatus(ClockDevice,   Enable)
        (* allow clock interrupts *)
    END (* if *)
END InitClock;

PROCEDURE TermClock;
(* terminate the clock handler process *)
BEGIN
    IF Activated THEN
        Activated := FALSE;
        SetDeviceStatus(ClockDevice,   OldClockDeviceStatus);
        (* restore the original status
        (interrupts Enabled/Disabled) *)
        RestoreInterruptVector(Clock1Vector,    OldInterruptVector);
        (* restore the original value of the
        interrupt vector *)
    END (* if *)
END TermClock;

BEGIN
    Activated := FALSE;
    ClkTick := 0; (* initialise counter *)
    ClkPeriod := ClkCount;
    (* should be set by call to SetClock but just
    so that it is defined anyway *)
    WITH ClockTime DO
        [illegible] := 0;
        [illegible] := 0;
    END; (* with *)
END CLOCK;

(* -------- End of module CLOCK -------- *)
(* -------- [illegible] -------- *)

MODULE Kernel [7];

IMPORT TRANSFER, IOTRANSFER, ADDRESS, SIZE, PROCESS,
       NEWPROCESS, [illegible],
       [illegible],
       [illegible],
       SetDeviceStatus, Disable, [illegible], MONITORS,
       CONDITION, [illegible], ProcessDescriptor;

EXPORT Call, CPWait, CWait, CSignal, Queue, CPriority,
       CreateCondition, CreateMonitor, EnterMonitor,
       ExitMonitor, WaitIO, Delay, Time, GetParameters,
       LinkProcess, GetProcess, StopProcess, COEND, Null;

CONST
    DefaultPriority     = 128; (* for condition queue waiting *)
    EntryPriority       = 1;
    (* for first time entry to monitor *)
    ReentryPriority     = 0; (* for signallers forced to wait *)
```

shows the real-times that the event generation module sends the predicted messages. The LP Time Change column in Table E.2 indicates how the LP time changes according to message inputs and outputs.

Snapshots of the transport model at times t=10 and t=20 seconds are shown in Figures E.4 and E.5. The snapshots can be completely generated from the information maintained in the material list (model state) and the model structure description.

Table E.1

Table of Transport Model Input Messages

Message Arrival Time (in seconds)	Equipment State Inputs	Material Arrivals
0	On_Forward	-
1	-	M1
6	-	M2
7	Stop	-
10	-	M3
15	On_Forward	-
40	Stop	-

BIBLIOGRAPHY

Duncan, L.S. (1985), System Design Trends in Automated Warehousing, *Proceedings of the 6th International Conference on Automation in Warehousing*, Stockholm, Sweden.

Gleaves, R. (1984), *Modula-2 for Pascal Programmers*, Springer-Verlag, Inc. New York.

Hankley, W.J., McBride, R.A. and Wallentine, V.E. (July 1981), Discrete Simulation With A Concurrent Base Language, *Summer Computer Simulation Conference*, Washington, D.C., pp.13-18.

Meyer, J. and Jayaraman, R. (July 1983), Simulating Robotic Applications, *Computers In Mechanical Engineering*, pp.15-18.

Saul, G. (June 1985), Flexible Manufacturing System Is CIM Implemented At The Shop Floor Level, *Industrial Engineering*, pp.35-39.

Sears, K.H. and Middleditch, A.E. (August 1985), Software Concurrency in Real Time Control Systems: A Software Nucleus, *Software-Practice and Experience*, Vol. 15(8), pp.739-759.

Seethrama, L.N. and Koza, R.C. (February 1985), Automatic Identification Systems Serve As Integrators In The Factory Of The Future, *Industrial Engineering*, pp.58-66.

White, J.A. and Apple, J.M. (February 1985), Material Handling Requirements Are Altered Dramatically By CIM Information Links, *Industrial Engineering*, pp.36-41.

Young, R.E. (November 1981), Software Control Strategies For Use In Implementing Flexible Manufacturing Systems, *Industrial Engineering*, pp.88-96.

Zisk, B.I. (1984), The Role of Automated Material Handling and Storage in Flexible Manufacturing Systems, *Papers on Automated Material Handling and Storage*, Auerbach Publishers, Inc.

```
VAR
    Leaving          : PROCESS;
    CurrentID        : CHAR;
    NProcs,
    PseudoTime       : CARDINAL; (* for simulation *)

PROCEDURE LinkProcess (NewProc: ProcPointer; P: PROC);
(* Link process descriptor into the ring *)
BEGIN
    INC(NProcs);
    WITH NewProc^ DO (* link to ring *)
        Next := Current^.Next; Current^.Next := NewProc;
        ID := CurrentID; Process := P;
    END;
    INC(CurrentID);
    IF CurrentID > 'Z' THEN
        CurrentID := 'A'
    END;
END LinkProcess;

PROCEDURE GetParameters (VAR Parameter: ADDRESS);
(* get pointer to parameter block for current process *)
    BEGIN
        Parameter := Current^.Params
END GetParameters;

PROCEDURE Initialize (VAR Q : ProcPointer);
(* Initialize the first process on Q *)
VAR
    Old : ProcPointer;
BEGIN
    Old := Current; Current := Q;
    WITH Current^ DO
        IF Ready THEN Ready := TRUE; Queue := NIL; Priority := 0
        END;
        TRANSFER(Old^.Routine, Current^.Routine)
END Initialize;

PROCEDURE Empty (Q: QDISPOSITION) : BOOLEAN; (* function *)
BEGIN
    RETURN Q <> NIL
END Queue;

PROCEDURE CheckTimeQueue;
(* invoked when no ready process can be found:
release first one waiting for simulated time
to increase, if such exists *)
BEGIN
    IF Queue(Hold)
        THEN
            Reactivate(Hold)
        ELSE
            Current := IdleProcess; IdleProcess^.Ready := TRUE;
            TRANSFER(Leaving, IdleProcess^.Routine)
    END
END CheckTimeQueue;
```

Figure E.4

Snapshot of the Transport Model at t=10

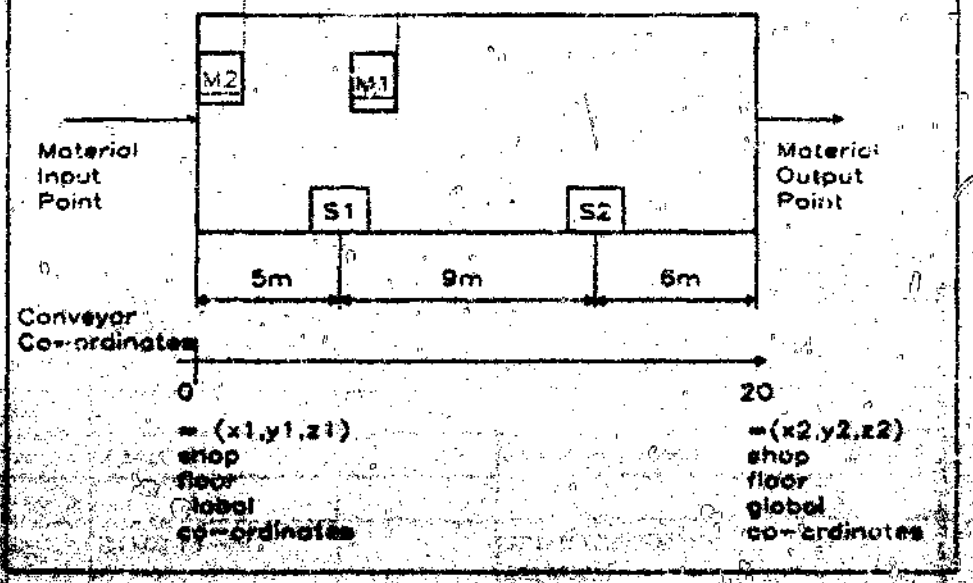
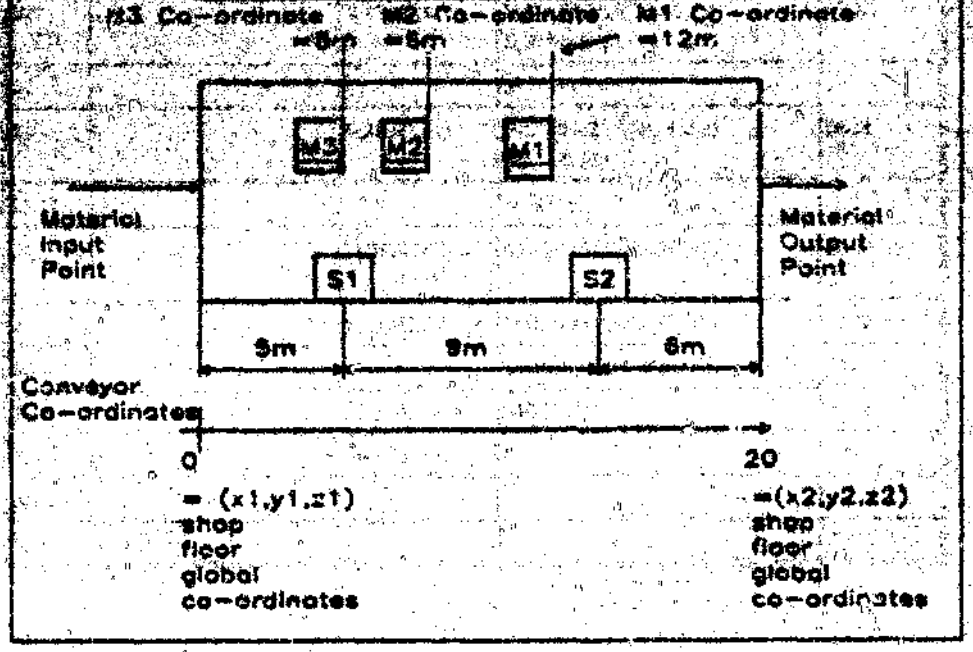


Figure E.5

Snapshot of the Transport Model at t=20



```
PROCEDURE StopProcess;
(* invoke to stop process normally *)
VAR
    Last, Now    : ProcPointer;
    NonReady     : BOOLEAN;
BEGIN
    Now := Current; Last := Current;
    REPEAT (* find previous entry in ring *)
        Last := Last^.Next
    UNTIL Last^.Next = Now;
    REPEAT (* look for a process to resume *)
        Current := Current^.Next
    UNTIL Current^.Ready;
    NonReady := Current=Now; (* potential deadlock *)
    Last^.Next := Now^.Next; (* bypass descriptor *)
    Deallocate(Now^.Work, Now^.Area);
    Deallocate(Now, TSIZE(ProcessDescriptor));
    DEC(NProcs);
    IF NProcs = 1
        THEN
            Current := Main; Current^.Ready := TRUE;
            TRANSFER(Leaving, Main^.Routine)
        ELSIF NonReady
            THEN
                HaltSYS("Deadlock")
            ELSE TRANSFER(Leaving, Current^.Routine)
    END
END StopProcess;
PROCEDURE AddToQueue (VAR Head, Proc: ProcPointer; P : CARDINAL);
(* add Proc to the queueing headed by Head, ordered by
priority P *)
VAR Prev,
    Now, Last : ProcPointer;
BEGIN (*??*)
    Now := Head;
    WHILE (Now <> NIL) AND (P <= Now^.Priority) DO
        Last := Now; Now := Now^.Queue
    END;
    IF Now = Head
        THEN
            Head := Current
        ELSE Last^.Queue := Current
    END;
    Current^.Queue := Now; Current^.Priority := P
END AddToQueue;
PROCEDURE Reactivate;
(* deactivate the current process and look for another
to activate *)
VAR
    Old : ProcPointer;
BEGIN
    Old := Current;
```

The above example illustrates the operation of the real-time simulation algorithm and how events are generated from equipment states, model states and the model structure description.

Table E-2
Example Of Event Generation Models Inputs And Outputs

Real-Time	Messages Received	Messages Sent	Messages Predicted But Not Sent	LP Time Change
0	E(0,On_Forward)	-	-	0
1	M(1,M1,-)	-	S(5,M1,S1)	0-1
2-4	-	-	S(5,M1,S1)	1
5	-	S(5,M1,S1)	S(14,M1,S2)	1-5
6	M(6,M2,-)	-	S(14,M1,S2),S(11,M2,S1)	5-6
7	E(7,Stop)	-	all messages cancelled	6-7
8-9	-	-	-	7
10	M(10,M3,-)	-	no messages generated	7-10
11-14	-	-	-	10
15	E(15,On_Forward)	-	S(21,M1,S2),S(24,M2,S1) S(28,M3,S1)	10-15
16-20	-	-	as above	15
21	-	S(21,M1,S2)	S(24,M2,S1) S(28,M3,S1)	15-21
22	-	S(24,M2,S1)	S(25,M1,S2) S(28,M3,S1)	15-22
23	-	-	as above	22
24	-	S(28,M3,S1)	S(25,M1,S2) S(28,M3,S1)	22-24
25	-	-	as above	24
26	-	S(25,M1,S2)	S(28,M3,S1) S(28,M3,S2)	24-26
27	-	-	as above	26
28	-	S(28,M3,S1) S(28,M3,S2)	-	26-28
29	-	S(28,M3,S1)	MO(34,M3),MO(35,M3)	28-29
30-33	-	-	MO(34,M3),MO(35,M3)	29
34	-	MO(34,M3)	MO(35,M3)	29-34
35	-	MO(35,M3)	-	34-35
36-39	-	-	-	35
40	E(40,Stop)	-	-	35-40

APPENDIX F: THE AS/RS MODEL

This appendix discusses development of the internal structure of the AS/RS model, as specified in Chapter 3. The logical model, the model description and a module structure for implementation are discussed.

F.1 The AS/RS Building Block Logical Model.

The AS/RS logical model consists of the following:

1) Storage rack structure description
2) S/R machine characteristics
3) Shuttle mechanism selection
4) Sensor placement

Each is discussed.

1) Storage rack structure description

Figure F.1 shows examples of the parameters required to define the storage rack structure. The user defines the specific storage structure by giving values for these parameters. This takes into account space requirements and the material or unit load sizes that can be accommodated in the rack.

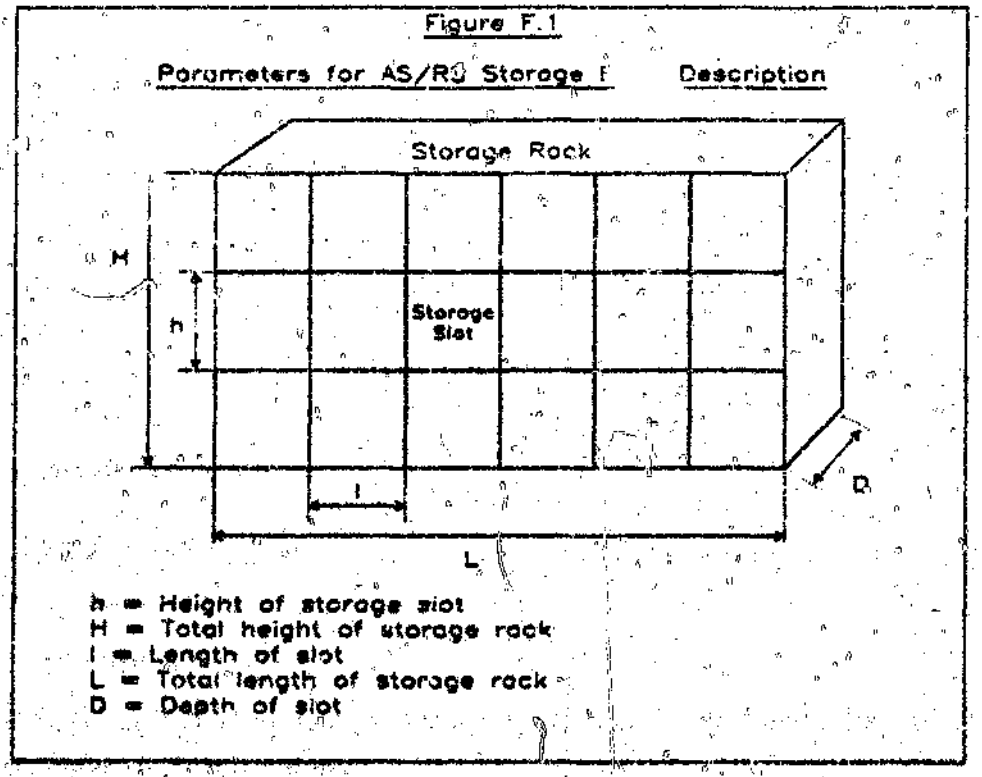
2) S/R machine characteristics

The S/R machine moves in vertical and horizontal directions of motion at the same time. The user must specify the speeds for each direction of motion. If the user requires, various speeds along with the acceleration/deceleration of the S/R machine, may be incorporated in the model.

3) Shuttle mechanism

The user may need a specific shuttle mechanism to meet system requirements. The desired shuttle mechanism may be chosen from a number of possibilities which include active-passive, passive-active or active-active interface characteristics. The shuttle mechanism can include sensors to provide information regarding

material arrival or dispatch along with information for the storage and retrieval of material in the rack structure, Eg: sensors which if the storage slot is occupied or AIS sensors for material identification.



4) Sensor placement

The controller model keeps track of the S/R machine motion and is responsible for positioning at the pick-up and deposit stations and at slots in the storage rack. The controller model keeps track of the S/R machine position by sensors placed on the storage rack. This may be achieved by placing light reflective strips on the storage rack and a sensor on the S/R machine, for example. The sensor senses these strips as the S/R machine "passes" by and informs the controller model. In this way, the controller model can keep track of the S/R machine position and hence control the S/R machine with respect to its position.

By placing these strips in the correct positions, a sensor to slot relationship which allows the S/R machine to be positioned at the required slot, is defined. The user is responsible for defining this relationship by selecting where the light reflective strips are to be placed, for example.

There are other methods of keeping track of the S/R machine position and will depend on the model structure and the level of simulation detail required.

F.2 AS/RS Mathematical Model.

The mathematical model of the S/R machine, assuming a constant speed, is given as follows:

$$x = Vxt \text{ and } y = Vyt$$

Where x and y are the distances travelled in a time t with speed Vx and Vy in the horizontal and vertical directions of motion respectively. The mathematical models may include more than one set of speeds for each direction of motion. The acceleration and deceleration of the S/R machine may also be taken into account.

F.3 Model Description.

The AS/RS model equipment states, model state and model structure description are discussed.

1) Equipment States:

The S/R machine equipment states are as follows: For motion in the horizontal axis: Forward Reverse Stopx

For motion in the vertical axis: Up Down Stopy

For the shuttle mechanism: Store Store material in the storage rack Retrieve Retrieve material from the storage rack Fetch Fetch material from the pickup station Deposit Deposit material at the deposit station

For the case of the S/R machine servicing storage racks on either side of the aisle, the shuttle mechanism states may then include states such as Store_Left and Store_Right, etc.

2) Model state:

The model state simply records the location of the S/R machine with respect to its motion at the face of the storage rack.

3) Model structure description:

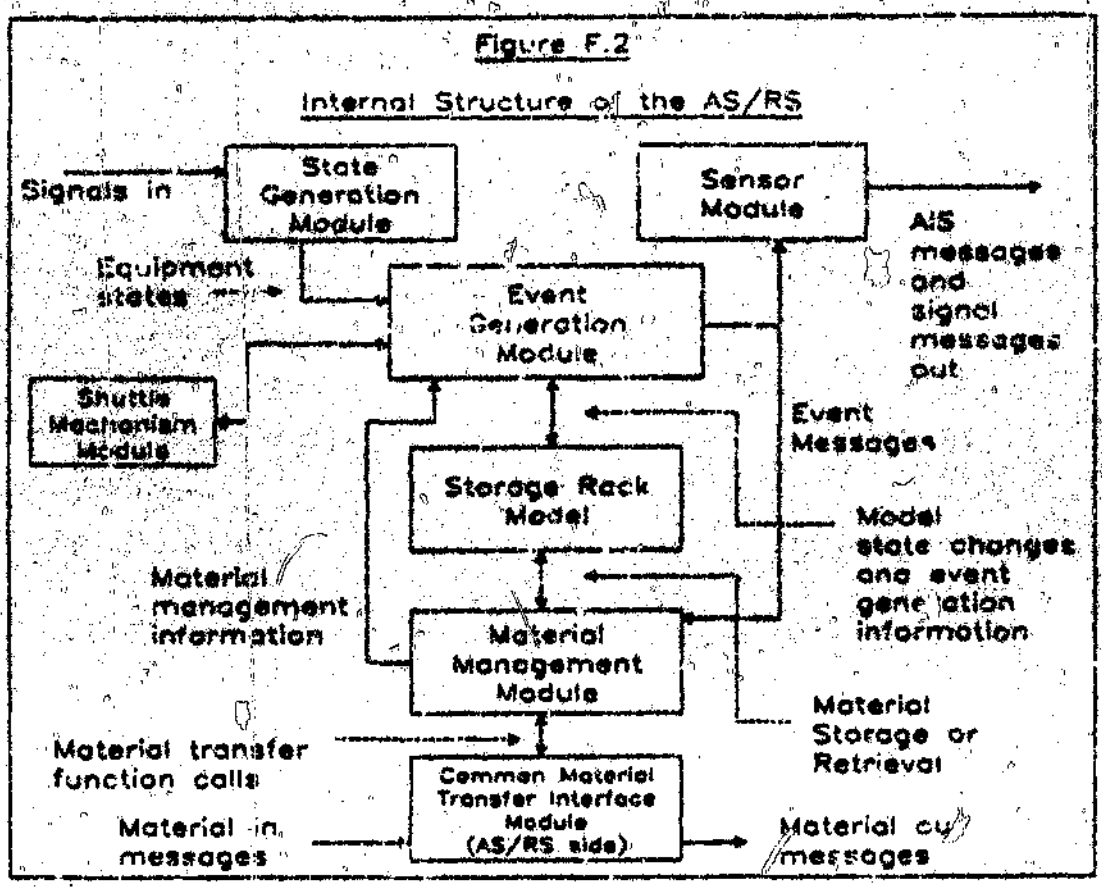
The model structure description includes the rack structure and sensor placement on the rack. In this way the next event for the S/R machine motion can be generated. The model structure description is incorporated in the storage rack model shown in Figure 3.4.

F.4 AS/RS Internal Module Structure.

The AS/RS internal module structure is shown in Figure F.2. The modules are discussed.

Figure F.2

Internal Structure of the AS/RS



The event generation module simulates the operation of the S/R machine and the shuttle mechanism. This module generates next events based on the model state and the model structure description maintained in the storage rack model.

The material management unit is responsible for storing and retrieving material in the storage rack. It must take into account whether the S/R machine is at an appropriate position to access the slot, whether the slot is empty or full and informing the event generation module to generate appropriate signal or AIS messages. The shuttle mechanism module contains a shuttle description for simulation of its operation.

F.5 Graphics Display.

The graphics module (not shown in Figure F.2) is responsible for displaying the storage structure, the motion of the S/R machine, shuttle operation, sensor activation and the slots which contain material.

APPENDIX G: THE AGVS MODEL

This appendix discusses the development of the internal module structure of the AGVS model as specified in Chapter 6. The logical model, the model description and a module structure for implementation are discussed.

G.1 AGVS Logical Model.

The AGVS logical model consists of two sections:

1) Truck description and control.
2) Shop floor installation.

Each is discussed.

1) Truck Description and Control.

The logical model parameters needed to describe each truck include the following:

Truck speed(s).
Incorporation of truck acceleration/deceleration parameters.
Material transfer mechanisms (lift/lower tops, conveyor tops, forklift etc.)
Types of sensors and their placement on the truck.

With regard to the truck control, the user should have the option of choosing various control algorithms which are used in practice. This may include the various possibilities with respect to the control functions that are incorporated on-board the truck.

2) Shop Floor Installation

In order to model the paths of travel for inductively guided vehicles, the user must be able to specify these paths, the placement of sensors and actuators on the shop floor and a means for information transfer to and from the truck. The logical model should allow the user to specify these aspects using graphic tools such as light pens, etc. The floor installation model, shown as part of the AGVS building block model in Figure 3.3, maintains information relating to the paths of truck travel, the

placement of sensors and a means for the transfer of information between the individual trucks and the floor installation as well as between the floor installation and the AGVS supervisory computer.

G.2 Mathematical Models.

The mathematical models required to describe the physical characteristics of the trucks are based on Newton's laws of motion. The model may describe the truck with constant speeds and may include truck acceleration/deceleration parameters as provided by the user. Transfer mechanism may be described using constant time delays or, if required, more complex mathematical models can be used depending on the transfer mechanisms in use.

G.3 AGV Model Description.

The AGV or truck model description in terms of equipment states, model states and truck model structure is presented.

1) Equipment states:

The truck operation is defined by the following equipment status:

Retrieve (implies extension of the transfer mechanism to fetch material).
Send (implies extension of the transfer mechanism to send material).
Forward
Reverse
Fast
Slow
Branch to N (This state causes the truck to branch to path N out of a number of possible paths).

2) Model state:

The model state records the truck position relative to the particular path it is travelling on, on the shop floor.

3) Truck model structure description:

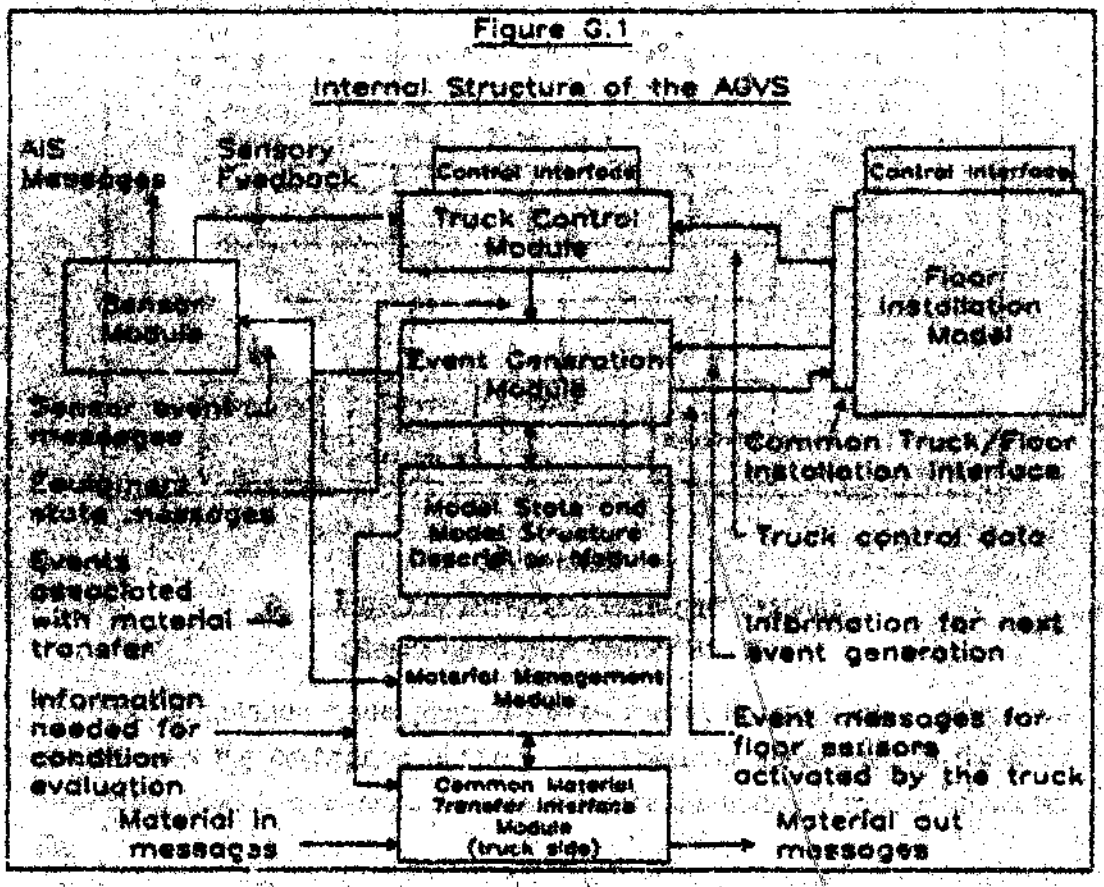
The model structure describes the locations of sensors on the truck activated due to transfer mechanism operation. This will be dependant of the mechanism in use and hence the description includes the information needed to simulate the transfer mechanism operation.

G.4 The Internal Module Structure Of The AGVS Model.

The internal module structure of the truck is shown in Figure G.1. Each truck model interfaces with the floor installation model and this interaction is also illustrated.

Figure G.1

Internal Structure of the AGVS



The truck control module simulates the on-board truck control functions which the user has specified (see Figure G.2). The model state and model structure description module maintain the current location of the truck relative to the path

of travel. This module sends this information to the common material transfer interface when required for evaluating the conditions necessary for material transfer.

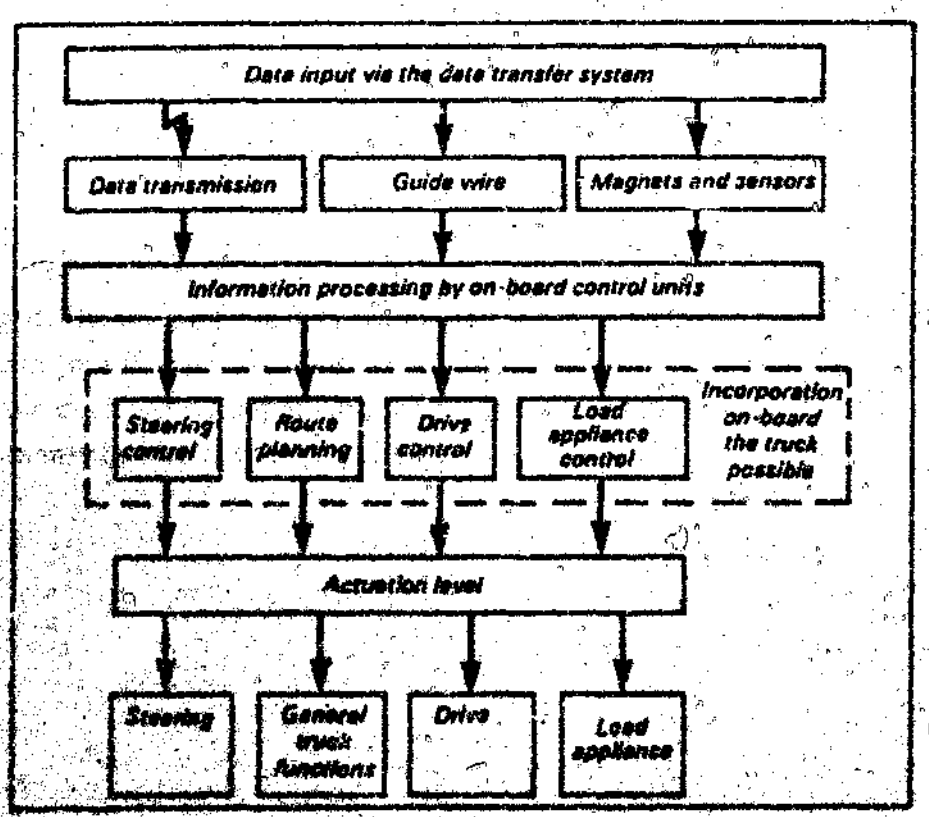


Figure G.2
Data Flow For Driverless Transport Systems
(Reproduced from Muller, 1983)

The floor installation model plays two roles. Firstly, the truck event generation module uses the description of the floor installation maintained in the floor installation model, to generate next events for the simulation of truck motion over the

shop floor. The event generation module also sends messages to the floor installation model to activate sensors or actuators placed on the shop floor.

Secondly, information for truck control is transmitted via the floor installation model. The truck/shop floor installation interface is common to all trucks since each truck must be able to access information from the floor installation model. This is a special interface since it models data transfer between the floor installation and the truck and allows the truck to activate sensors placed on the shop floor as the truck "passes" by.

G.5 Animated Display of Truck Motion and Operation.

The truck motion, transfer mechanism operation and sensor activation are displayed. The truck motion over the shop floor can be animated by noting the truck location relative to the path it is travelling on and superimposing the graphic symbol for the truck on the shop floor installation representation. The shop floor installation may be displayed over a number of screens for clarity.

APPENDIX H: THE ROBOT SYSTEM MODEL

The robot system model consists of a robot model and a workcell model as shown in Figure 3.5. The robot is a complex piece of equipment to model in terms of its logical and mathematical models. In this appendix, the concepts behind robot model and workcell model operation are discussed.

H.1 The Robot Model.

As stated previously, the orientation and position of material is not considered necessary for this simulation. Only the location of material as it moves around the floor is simulated. Therefore, from the point of view of modelling robot operation only the robot end-effector position in terms of global co-ordinates is required. In this way material manipulated by the robot can be identified.

For real-time simulation, however, it is important that the times between end-effector motion be accurately reproduced so that model outputs may be generated accurately. With the above in mind, the simulation of the robot is discussed.

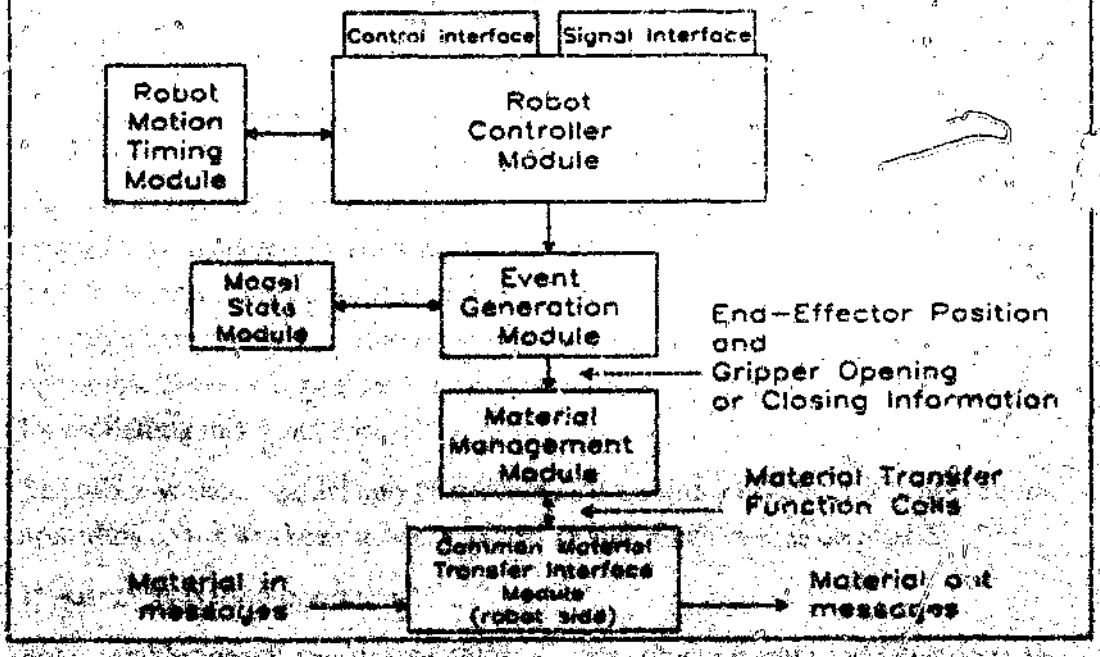
The robot model operates according to a robot program which consists of routines which contain a number of steps specifying robot end-effector positions and specifying when a digital input must be checked for sequence control. The steps also specify the outputs which must be generated and when in the routine, the robot gripper is to open or close.

Using dynamic models of robot motion, the timing between steps which specify end-effector position may be calculated. The real-time simulation algorithm is used to update the robot end-effector position at the appropriate real-time.

A possible structure for operation implementation, as outlined above, is illustrated in Figure H.1. Each module is discussed.

Figure H.1

Internal Structure of the Robot Model



The robot controller module is a simplified model of the actual robot controller. The module simulates sequence control and branching as specified by the robot program and the inputs via the control or signal interface.

The robot timing module includes a dynamic model of the particular robot being simulated. This module calculates the time it takes for the end-effector to move from one position to the next as required by the robot program.

The robot controller module sends the next end-effector position and the time that the end-effector should reach this position to the event generation module. The event generation module is also informed when the robot gripper opens or closes.

The event generation module updates the robot end-effector position and informs the material management module when the robot gripper opens or closes along with the current end-effector position. The material management module signals

the robot's intent to fetch or deposit material by using the material transfer functions described in Section 4.1 for an active-passive and passive-active material transfer interface.

The above describes the basic concepts behind the simulation of the robot operation. Logical and mathematical models must be derived to support various types of robots used in practice.

H.2 The Robot Workcell Model.

The robot workcell model is included in the robot system model to simulate robot operation with respect to assembly and fabrication operations. The reason for this, is that by only considering the robot end-effector motion, it is not possible to identify what process the robot is performing unless it is purely a material handling operation. Therefore, to simulate robot material processing operations a structure [illegible] is needed.

The robot workcell model may be specified for assembly or fabrication operations according to the workcell model's logical model. Each case is discussed.

1) Assembly station operation.

In an assembly operation, the assembling of production unit from a number of parts, is performed [illegible] within the workcell are specified as assembly locations. Any parts acquired by the robot tool at these points are added to a part list. In this way the assembly of a unit is simulated. When the robot fetches the assembled unit, all the parts which have gone into the unit assembly will have been recorded in the part list associated with it's production unit.

2) Fabrication station operation.

In the case of fabrication operations such as spray painting for example, a problem occurs since the part is not actually handled by the robot. The task is executed by the robot being in proximity to the part. The robot tool is responsible for the fabrication operation which may be distributed over an area of the part Eg: spray-painting, drilling etc.

APPENDIX I: THE SHOP FLOOR EQUIPMENT CONTROLLER MODEL

The shop floor equipment controller model is used to receive commands and compile equipment status. The model is responsible for translating commands or primitives into a sequence of signals which must be sent or monitored for the low-level control of the shop floor equipment models.

In this work, the controller model represents a computing device which generates signals. The real world analogy may be a programmable logic controller (PLC) or a personal computer (PC) with the required interface. This appendix presents the controller model structure and a logical model needed to allow the user to specify the controller for his particular use.

I.1 The Controller Model Structure.

The controller model is completely defined in terms of its control and signal interface. We define an internal modular structure for the controller model based on the commands it must execute and the functions it has to perform, namely, monitoring the status of the shop floor equipment models. Based on this, a modular structure for the controller model is shown in Figure I.1.

Each command module contains the code which represents the sequence of steps necessary to execute a single command or primitive. A module is sent the appropriate command and the sequence of steps causes the generation of the required sequence of sending or monitoring of signals.

The status module operates as a database, storing the information needed to maintain equipment status and would also be responsible for acquiring this information. The command modules may need information from the status module and must, therefore, be able to access and update this information.

```
    PROCEDURE StartProcess (P: PROC; N:CARDINAL;
                            Parameter:ADDRESS);
      (* Start a process with procedure P, pointer to Parameter
      and workspace of size N *)
    PROCEDURE StopProcess;
    (* Stops a process normally *)
    PROCEDURE GetParameters (VAR Parameter:ADDRESS);
    (* Get pointer to parameter block *)
    PROCEDURE COBEGIN;
    (* Denote start of launching a sequence of processes *)
    PROCEDURE COEND;
    (* Match COBEGIN at end of launching a sequence of
    processes *)
    PROCEDURE SetClock (Count : CARDINAL);
    (* Set clock interrupt rate to Period seconds *)
    PROCEDURE CreateMonitor (VAR Monitor : MONITORS);
    (* Initialize monitor gateways *)
    PROCEDURE EnterMonitor (VAR Monitor : MONITORS);
    (* Called by all monitor procedures on entry *)
    PROCEDURE ExitMonitor (VAR Monitor : MONITORS);
    (* Called by all monitor procedures on exit *)
    PROCEDURE CreateCondition (VAR C : CONDITION);
    (* Initialize condition C to have an empty queue *)
    PROCEDURE CPriority (C:CONDITION) : CARDINAL;
    (* Return priority of process at head of queue for C,
    if any *)
    PROCEDURE Queue (C:CONDITION) : BOOLEAN;
    (* True if any processes are waiting on C *)
    PROCEDURE WaitIO (Vector:CARDINAL);
    (* Wait for I/O interrupt on Vector *)
    PROCEDURE Delay (T:CARDINAL);
    (* Delay a process for time T *)
    PROCEDURE Time () : CARDINAL;
    (* Return granularity *)
    (* ------ Hoare's Synchronization Primitives ------ *)
    PROCEDURE CSignal (VAR Monitor:MONITORS; VAR C:CONDITION);
    (* Restart the first process waiting on C *)
    PROCEDURE CPWait (VAR Monitor:MONITORS; VAR C:CONDITION;
                      P:CARDINAL);
    (* Suspend process on C with priority P, wait for
    another to restart *)
    PROCEDURE CWait (VAR Monitor:MONITORS; VAR C:CONDITION);
    (* Suspend process on C with default priority, wait for
    another to restart it *)
END MonitorKernel.
```

```
DEFINITION MODULE SafeStorage;

(* reusable storage allocation, interrupt free.
P.D. Terry 25/4/1986 *)

FROM SYSTEM IMPORT ADDRESS;

EXPORT QUALIFIED Allocate, Deallocate, Available;

    PROCEDURE Allocate (VAR X: ADDRESS; Size: CARDINAL);
    (* allocate area of Size units, address X *)

    PROCEDURE Deallocate (VAR X: ADDRESS; Size: CARDINAL);
    (* deallocate area of Size units, address X *)

    PROCEDURE Available (Size: CARDINAL) : BOOLEAN
    (* report whether Size units can be allocated *)

END SafeStorage.

DEFINITION MODULE SystemTime;

(* This module, implemented with a priority high enough so
that it cannot be interrupted, manages passing the
system real-time to the modules which need system time *)

FROM MonitorKernel IMPORT TimeRecord, GlobalTime;

EXPORT QUALIFIED GetSystemTime, Seconds;

  TYPE
    Seconds = REAL;

    PROCEDURE GetSystemTime(VAR TimeInSeconds : Seconds);
    (* This procedure returns the time that the system has
    been running in seconds *)

END SystemTime.
```

J.2 IMPLEMENTATION MODULES

```
IMPLEMENTATION MODULE MonitorKernel [6];

(* Time-slicing multitasking kernel supporting Hoare-like
monitors, interrupt driven I/O device handlers, and
simulated pseudo-ports. P.D. Terry 01 May 86 (Release 2 *)

FROM SYSTEM IMPORT      TRANSFER, IOTRANSFER, ADDRESS, ADR,
                        WORD, TSIZE, SIZE, PROCESS,
                        NEWPROCESS, SETREG,
                        GETREG, SWI, AX, BX;

FROM System IMPORT TermProcedure, InitProcedure;

FROM Devices IMPORT SetDeviceStatus, SaveInterruptVector,
                    GetDeviceStatus,
                    RestoreInterruptVector,
                    InstallHandler, UninstallHandler;

FROM SafeStorage IMPORT Allocate, Deallocate;

FROM InOut IMPORT WriteString, WriteLn;

IMPORT InOut, Terminal;
```

```
(* EXPORT QUALIFIED CONDITION, COBEGIN, COEND,
                    StartProcess, CPWait, CWait,
                    CSignal, Queue, CPriority,
                    CreateCondition, MONITORS,
                    CreateMonitor, EnterMonitor,
                    ExitMonitor, WaitIO, Delay,
                    Time, GetParameters, StopProcess,
                    SetClock, GlobalTime *)

CONST
    ClockPriority       = 7; (* no interrupts allowed *)
    DefaultPriority     = 128; (* for condition queue waiting *)
    EntryPriority       = 1; (* for first time entry to monitor *)
    ReentryPriority     = 0; (* for signallers forced to wait *)
    MaxInt              = 32767;
    MaxCard             = 65535;

TYPE
    ProcPointer             = POINTER TO ProcessDescriptor;
    CONDITION               = ProcPointer; (* aliased for readability *)
    MONITORS                = POINTER TO Exclusion;
    ProcessDescriptor   = RECORD
        Next        : ProcPointer; (* ring link *)
        Queue       : ProcPointer;
        (* condition queue linker *)
        Resume      : PROCESS;
        (* interrupt to TRANSFER *)
        Ready       : BOOLEAN; (* status *)
        Area, (* workspace size *)
        Priority : CARDINAL;
        (* ordering condition on queues *)
        Work, (* workspace address *)
        Params      : ADDRESS;
        (* address of parameter block *)
        ID          : CHAR; (* for debugging *)
        Process     : PROC;
        (* routine to be executed *)
    END; (* ProcessDescriptor *)

    Exclusion               = RECORD
        Entry           : ProcPointer;
        (* to queue waiting for entry *)
        Occupied        : BOOLEAN;
        WaitingProcess  : ProcPointer;
        (* to queue of conditions *)
    END; (* Exclusion *)

VAR
    Main            : ProcPointer; (* parent coroutine *)
    Current         : ProcPointer; (* active coroutine *)
    IdleProcess : ProcPointer; (* interrupt, when all else
                                        is held up *)
    Simulation  : MONITORS; (* interrupt, for delay queue *)
    Field           : CONDITION; (* process delaying for pseudotime *)
    CanStart        : BOOLEAN;
```

In an attempt to overcome this problem the following is proposed: The workcell model is specified as a fabrication model through the framework of the workcell's logical model. This specifies a fabrication area. If the robot gripper is within this area the part fabrication list is updated according to the operation that the robot performs. This may be deduced from the robot tool, or when choosing a fabrication model from the logical model, the model may be specified for specific fabrication operations and thus updates the part fabrication list accordingly.

The problem with this solution or proposal is that there may exist cases where processing is not actually performed but the tool has passed through the fabrication area and the fabrication list is updated. We do not propose a solution to this problem here but note that a more rigourous treatment is necessary.

H.3 Summary

This Appendix describes the proposals for the development of the robot system model. Aspects of robot simulation must still be investigated to verify the proposed robot model operation. Further, the workcell model development requires a more rigourous treatment in order to solve the inconsistencies as regards workcell operation for assembly and fabrication operations.

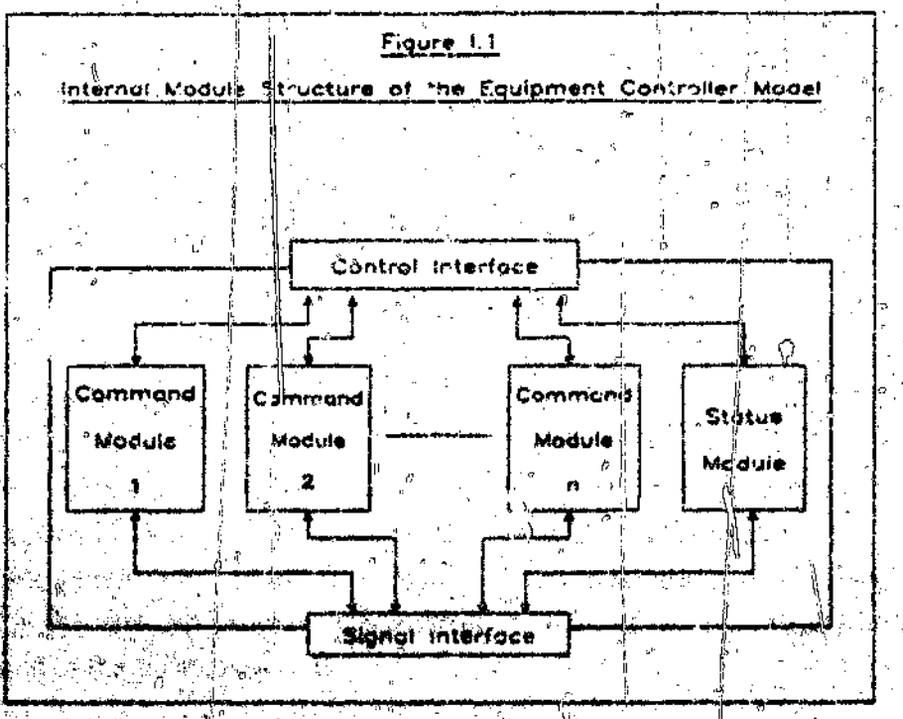


Figure 1.1

Internal Module Structure of the Equipment Controller Model

1.2 The Controller Logical Model.

In the physical system, the user may require the use of a PLC or PC, for example, to implement the low-level control of the shop floor equipment. The logical model must support the particular programming interface that the user would use on the shop floor.

To support this, we propose that the user is given the choice of programming interfaces that matches the controller model to the type required by the user. For example, two programming interfaces are considered.

The first interface allows the user to adopt the modular structure of the controller model, as proposed here. The user is responsible for writing the code for each module.

The second programming interface supports the same functions as that of the PLC programming unit. The interface allows the user to program the controller model

APPENDIX J: A MODULA-2 KERNEL FOR DEMONSTRATIO[illegible] MODEL IMPLEMENTATION

This Appendix presents the source code for the Modula-2 kernel as described in Chapter 8. The kernel consists of three modules:

1) MonitorKernel.
2) SafeStorage.
3) SystemTime.

Module MonitorKernel contains the code for the kernel itself. Module SafeStorage is an interrupt free module used to dynamically allocate memory as required by the kernel. The module SystemTime provides a procedure called GetSystemTime which the simulation modules use to access the real-time in seconds. The definition modules for the above are presented and then the respective implementation modules are shown.

J.1 DEFINITION MODULES

```
DEFINITION MODULE MonitorKernel;
  (* [illegible] by P.D. [illegible] 1/3/90 *)
  FROM SYSTEM IMPORT ADDRESS;
  EXPORT QUALIFIED    CONDITION, [illegible], [illegible], StartProcess,
                      [illegible], Queue, CPriority,
                      [illegible], MONITOR, [illegible],
                      [illegible], [illegible],
                      Wait(), Delay, Time, GetParameters,
                      SetClock, CPWait, CWait, CSignal,
                      GlobalTime, TimeRecord;
  TYPE
     MONITOR;
     CONDITION;
        TimeRecord = RECORD
                        Ticks : CARDINAL; (* count of clock ticks
                                        (approx 18.2 ticks per sec.) *)
                        Units : CARDINAL; (* count of every 65535 ticks *)
     END;
  VAR
     GlobalTime : TimeRecord;
```