University of Manitoba

DEVELOPMENT OF A MULTITASKING CONTROL ENVIRONMENT FOR A FLEXIBLE MANUFACTURING SYSTEM

by

Craig P. Judt

A Thesis

submitted to the Faculty of Graduate Studies
in partial fullfilment of requirements for
the degree of

Master of Science in Mechanical Engineering

Winnipeg, Canada, 1988

ISBN 0-315-48127-7

DEVELOPMENT OF A MULTITASKING CONTROL ENVIRONMENT FOR A FLEXIBLE MANUFACTURING SYSTEM

BY

CRAIG P. JUDT

A thesis submitted to the Faculty of Graduate Studies of
the University of Manitoba in partial fulfillment of the requirements
of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Craig P. Judt

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Craig P. Judt

The University of Manitoba requires the signatures of all persons using or photocopying this thesis. Please sign below, and give address and date.

ABSTRACT

Manufacturing systems have been slowly changing from traditional assembly line production systems to systems that are more flexible and smaller batch oriented. Computer Integrated Manufacturing (CIM) has been at the forefront of this change in strategy. CIM systems consisting of Computer Numerically Controlled (CNC) machines, material handling robots, and vision systems are rapidly becoming the state of the art production systems. Most currently available systems, unfortunately require a cohesive unit of selected robots, CNC machines, and computers. These systems generally have to be supplied by a single vendor and are "turn key" systems. Single vendor control has made implementation of the new technology expensive and geared to large corporations. Much of existing equipment is rendered obsolete by these turn key packages because of hardware incompatibilities between vendor equipment.

This thesis presents an alternative to the single vendor / system approach to automation. Developed under this thesis project is a prototype Manufacturing Control Environment (MCE) which ties together the various phases of a manufacturing system from product design through part

production. The system developed is based on a prototype CIM cell consisting of a CNC milling machine, a CNC lathe, a material handling robot, an assembly robot and an Automated Storage and Retrieval System (ASRS). Presented in this thesis, is a personal computer based manufacturing control system. The system utilizes readily available IBM PC/AT compatible personal computers and a real-time multi-user multi-tasking operating system. Geared towards the small to medium sized manufacturing industry, it demonstrates the ability to integrate typically stand alone incompatible devices into a cohesive CIM system, in a flexible and cost effective manner.

ACKNOWLEDGEMENTS

The author would like to thank Dr. S. Balakrishnan for acting as thesis advisor. His guidance, and dedication is appreciated gratefully.

The author would also like to thank Mr. K. Tark for his work on the contstruction of the various computer hardware interfaces which were required in this project.

Table of Contents Page no.

List of Figures Page no.

I. INTRODUCTION

1.1 Background

Manufacturing philosophies have gradually been changing over the past few decades. The early mass production, assembly line techiques, are being replaced by more efficient batch oriented production systems. Shorter product life and greater varieties, have demanded more flexible and efficient manufacturing systems. Computer Integrated Manufacturing (CIM), has been the basis of this shift in philosophy. CIM is centered primarily on one concept; the integration of shop floor operations into a computerized network. One possible definition of CIM may be considered as "a means of gathering, manipulating and distributing data" [1].

Computer Integrated Maufacturing systems are generally comprised of any combination of:

- Computer Numerically Controlled (CNC) machines such as milling machines, lathes, and machining centres;
- robots used for material handling, welding, and assembly applications;

- sensors for monitoring and coordinating various functions within the CIM cell;
- part orientation mechanisms, and material handling / holding devices;
- fixtures for automated loading/unloading of workparts;
- vision systems for part recognition, quality control; and
- control computers used for sequencing operations and transferring information between equipment.

The evolution of CIM has been controlled by the developments in the area of digital control and microprocessors. During the early days of CIM, real time control of CNC machines and robots had required expensive high performance computer systems. These primarily included mainframe and minicomputers, for a variety of reasons:

- CNC machines were not "intelligent" systems and required dedicated on-line control in order to integrate them for CIM applications;
- only high-performance mainframe and minicomputers were available and suitable for the factory floor; and
- mini and mainframe computers were the only systems capable of operating the equipment in real-time.

As a result, only large companies could afford to investigate CIM, since the capital investment and computer requirements were very large. Most plant operations were controlled from one or two mainframe computers. This was considered the centralized control approach. This approach had associated with it a number of problems;

- in the event of a malfunction of the central computer, the entire plant was affected;
- the central computers became bogged down doing low level repetitive tasks such as monitoring sensors;
- communication to individual devices became difficult due to logistics and transmission distances; and
- software development and maintenance became extremely difficult due to the size of the system.

With the development of the microprocessor, more intelligent and sophisticated CNC machines and computers have evolved. With this technology a new philosophy of "distributive control" has emerged [1-3]. The principle behind the distributive control philosophy is presented in section 1.3.

1.2 Requirments for CIM

The complexities of Computer Integrated Manufacturing systems vary based on control computers and the functions they can provide. CIM involves the linking together of office information systems and the factory floor production equipment. The computer control system must therefore be capable of tieing together the design through production phases of a manufacturing system. This would include:

- Computer Aided Design (CAD) systems;
- Computer Aided Manufacturing (CAM) systems;
- inventory control;
- providing production scheduling information; and
- factory floor machine control and production system sequencing.

These are the major components that must be included in the design of any computer control system for integrated manufacturing. The computer requirements for each area of the manufacturing system is different. The factory floor requires support for hardware control systems including communication to CNC machines, sensor monitoring, robot

control interfaces, and other material handling equipment. Office information systems require database management and high performance graphics display systems for CAD/CAM applications. A distributed network of control computers lends itself to these diverse computer application requirements. Sectors of the computer network may be taylored to the specific needs of one area providing relatively autonomous operation while at the same time providing information to the rest of the computer control system. This is another application which has propagated the distributive control philosophy in information processing.

1.3 The Distributive Control Philosphy

The distributive control philosophy is based on the concept of "distributing the management of information". On the factory floor this involves allowing "intelligent equipment" such as CNC machines and robots to control as much of the low level control processes as possible by using their own controllers. Further monitoring and integrated control of such devices is provided by a computer control network. Each computer or "node" in the network is assigned specific monitoring duties. Information is transmitted between nodes over the network. Computer Networks or Local

Area Networks (LANs) are the heart of the distributive control philosphy. Computer networks such as those based on the VAX / Micro-Vax family of computers by Digital Equipment are currently being used in manufacturing applications. These "high-performance" networks utilize the distributive philosphy. However they are not suitable for many small and medium sized industries due to the high cost associated. For example asmall networks comprised of a single VAX 11-780, and 3 Micro-VAX computers can cost in excess of $250,000 when installed. Third party computer software and hardware maintenance is also expensive.

LANs are primarily based on "multitasking" operating sytems. Multitasking operating systems use a process known as "time-slicing" to run multiple tasks or programs concurrently on a single computer. A network of computers operating under such a system allows the sharing of peripherals such as printers, and mass storage devices, thus providing increased flexibility and lower cost. With the current advancements in microcomputer technology, the personal computer based on the IBM-PC/AT bus is making many inroads into North American industries and not just in the office environment. Industrial versions of the Personal Computer are now available and are being used on the factory

floor. A high performance and cost ratio makes them very attractive for many industrial monitoring and control applications. Keeping this in mind, the present work focuses on utilizing industrial personal computers in the process control environment.

Over the past four years, multitasking operating systems for the personal computer have emerged. High-performance multitasking operating systems and industrial versions of the personal computer make low cost industrial control networks a reality. A network of industrial personal computers can cost as little as one fifth the cost of a high performance network based on the VAX Micro-Vax family of computers. While these computers may not provide the same performance as the VAX family, many configurations may provide more than adequate performance and flexibility for a variety of applications.

1.4 Thesis Objective

This thesis is intended to describe the development of a prototype Manufacturing Control Environment (MCE) based on networked Industrial Personal Computer (IPC) design. This control environment is based on a prototype CIM cell in the

Industrial engineering laboratory at the University of Manitoba. The cell, to be described in greater detail in section 4.1 consists of the following components:

- a Computer Numerically Controlled mill and lathe;
- a five degree of freedom 6 kg payload capacity robot for material handling;
- a five degree of freedom small articulated robot for assembly tasks;
- a custom designed, stepper motor controlled automated storage and retrieval system; and
- a central microcomputer on which the developed MCE is implemented; and
- an additional computer for CAD/CAM operations pertaining to the cell.

Several custom developed retrofit devices for part clamping on the mill, and automatic chucking of parts on the lathe are also included in this system.

This thesis is intended to focus on the ability to use networked industrial computers operating under a "real-time" multi-tasking operating system to integrate all phases of a manufacturing system. The control system developed to date, is based on an 8 MHz 'MIND' personal computer (IBM PC/XT compatible) as a cell controller, and a 12 mhz 'MIND' AT (IBM PC/AT compatible) as a CAD/CAM workstation. The CAD/CAM workstation ties together CAD/CAM, scheduling, inventory, and machine control. The developed system was based on a networked industrial computer concept.

In addition, the MCE demonstrates a unique on-line control display system for portraying cell activities at any given time. This is facilitated by a graphic window overlay concept.

1.5 Thesis Organization

Chapter 2 describes the current published trends in the area of Computer Integrated Manufacturing systems. Chapter 3 describes the system design requirements and presents justification for various component selections. Chapter 4 describes the prototype CIM Cell on which the control system is based. Chapter 5 presents a general overview of the Manufacturing Control Environment (MCE). Chapter 6 describes the design of the developed software sub-systems. Chapter 7 presents a conceptual design of an emergency monitoring system. Conclusions and recommendations are presented in the closing sections.

II. CURRENT DEVELOPMENTS IN CIM

The distributive control philosophy is becoming prevalent in the design of state of the art Flexible Manufacturing systems. Of the systems currently developed most are based on high power mainframe or mini computers such as the Micro-VAX from Digital Equipment Corp. The cell controller/network sturcture as illustrated in Figure 2.1, is a widely accepted concept, but to date only a few working systems of this type have been developed [4,5].

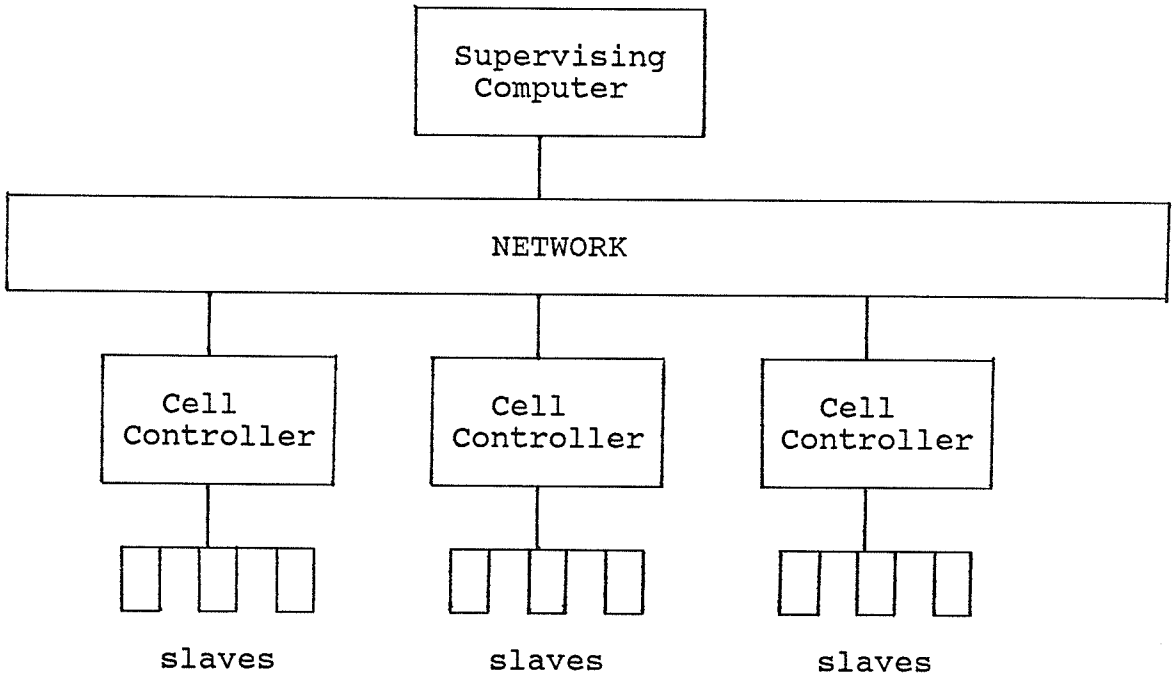


Figure 2.1 - Cell Controller Network Structure

The use of industrial IBM-PC/AT based computers as cell controllers and slaves is being presented as a feasible cost-effective direction for control of Flexible Manufacturing Systems [4,6,7,8]. The number of actual systems implemented to date appear to be few and in-depth details of such systems have not been published.

A system primarily used for research applications [25], utilizes networked IBM PC/AT microcomputers in controlling scheduling sequences for a series of machining centers. This system is quite simple and does not have any assembly or storage system. Each part entering the system is independent of each other. The system presented, in this thesis, requires combinations of components to be assembled thus complicating the control sequencing dramatically.

Many researchers have presented theoretical control schemes for a variety of CIM operations but very few have been implemented [9]. Control schemes based on conceptual machinery and handling mechanisms tend to over simplify the difficulties when the problem of integrating existing machinery is presented. Integrating an existing piece of machinery, designed initially to be operated as "stand-alone" with its own controller, presents many difficulties

not considered by most conceptual control schemes.

Solutions to this type of problem is not simply one of removing old equipment and replacing with new. Companies whether small or large generally cannot afford this approach. Instead, interfaces must be developed to remotely control these existing pieces of equipment. Many times the design of the control interface will directly affect the computer control scheme by forcing compromises. Certain components may have to operate sequentially due to hardware restrictions even though it may not result in an optimal production scheme.

This thesis intends to demonstate the development of a multi-tasking control environment for a prototype CIM cell at the Industrial Automation Laboratory of the University of Manitoba. All the computer controlled machines were not initially designed for integrated operation. Presented are the control interface solutions and compromises to the particular problems presented by the prototype CIM cell. The following chapter outlines the principles behind multitasking operating systems. The primary control system design criterias are also presented.

III. REAL - TIME PROCESS CONTROL USING INDUSTRIAL PERSONAL COMPUTERS (IPC)

3.1 Real-Time Multitasking Process Control

In the past, utilization of personal computers in control applications were based on single tasking operating systems such as MS-DOS. Single-tasking operating systems generally restrict applications to sequential control. Single tasking operating systems generally employ tight looped polling practices while waiting for information. Although a number of groups have outlined control software strategies based on PC-DOS [10], for CIM applications none implemented on a prototype system have been reported in literature. The MS-DOS / PC-DOS operating systems have so far been the primary system used to date, although many other operating systems are now available. XENIX by Microsoft Corp., and Concurrent PC-DOS by Digital Research are two examples. Control of a number of hardware components simultaneously requires a "real-time" multitasking operating system. There are two types of multitasking operating systems now available for personal computers. One of them is "real-time", and the other is "time-shared". QNX by Quantum Corp., is considered to be "real-time" while XENIX by Microsoft Corp. is "time-shared" [11-13].

First it is important to understand the three different types of operating systems. A single-tasking operating system (OS), such as MS-DOS allows only one program (task) to be run at any given time. A multitasking operating system allows a number of tasks to be run "concurrently" or, at the same time. An example of the improved computer utilization under a multitasking operating sytem is a print spooler. Under a single tasking OS the computer must dedicate all its time to printing files on a printer. Under a Multitasking OS the user may be printing on the printer and operating a word processor at the same time.

A multitasking system uses a process known as time-slicing to run a number of tasks concurrently. The time-slicing process switches between tasks every "fraction of second" thereby allowing a number of tasks to run almost "simultaneously". Task switching speeds vary between operating systems and is operating system dependent. The time-slicing technique was developed to improve the utilization of computing power. It has been found that most cpu-time under a single-tasking operating system is under utilized while waiting for action to take place. An example of this is someone using a word processor. Because of the

speed of the computer, most of the time the computer is waiting for a key press and not actually working even if the user is a high speed typist.

There are a number of differences between normal multitasking and "real-time" multitasking operating systems. The primary differences between the two include:

- the speeds at which task switching takes place;
- the available facilities for accessing devices attached to the IBM-PC/AT data bus, such as digital Input/Output boards, peripheral devices etc.; and
- available timing features for providing task wakeup accuracy to under 50 milliseconds.

Ordinary time-shared operating systems have very slow task switching speeds. Slow task switching speeds are a result of the "monolithic" design of most time-sharing operating systems, such as UNIX [14]. Slow task switching results in delays in the order of milliseconds between task servicing and thus produces unpredictable task response times. Benchmarked task switching speeds on an 8 MHz IBM-PC/AT showed XENIX, by Microsoft Corp to produce approximately 400 task switches per second, compared to QNX, by Quantum Corp., at 2800 task switches per second [14].

Control of processes in the case of CIM, requires accurate timing. Control of mechanical systems such as

pnuematic clamps and part ejection mechanisms, have to be accurately synchronized and may require finite time delays for proper functioning. Timing features for such processes is quite critical. Real-time multitasking operating systems minimize task switching time. High speed task switching ensures that if a number of tasks are running on a single machine, each task sees minimal degradation in response performance. The QNX operating system is considered to be a real-time system because of its high task switching speeds. On an Intel 80386 cpu based personal computer, task switching speeds are in excess of 6600 task switches per second while utilizing the QNX operating system.

The concurrent control of multiple pieces of hardware requires continuous checking of hardware conditions, and thus access to computer hardware interfaces. Checking for equipment malfunctions is an example where continuous checking is required. It is critical that if a malfunction occurs, an appropriate action is taken immediately. Under a real-time multitasking operating system a task can be created for checking conditions which signify a malfunction. If a malfunction occurs then this monitoring task will respond by interrupting the operating system, and taking appropriate actions.

3.2 Design Philosphy for a Multitasking Control Environment

CIM systems have as their base a computer control network. Typically, most automation systems currently available will only allow integration of equipment, already designed with external communication interfaces. As well, this equipment must generally be from the same vendor. Therefore, companies looking to integrate existing stand alone, multi-vendor equipment face a serious problem.

The work done in this thesis demonstrates the ability to integrate typically stand alone incompatible devices into a cohesive manufacturing system. This integration of devices and software into a flexible manufacturing system is demonstrated utilizing readily available IPC's and control hardware.

Four primary design constraints were considered in the preliminary system design.

1. The system software and hardware development costs should be kept to a minimum.
2. The system should be based on readily available hardware and software.
3. The control software should be modular and structured to allow for future modification and expansion with provisions for integration of additional equipment.

4. The overall control environment should demonstrate the ability to integrate:

   - CAD/CAM software subsystems;
   - machines, such as mills and lathes;
   - mechanical/electrical retrofit systems, including pnuematic clamps, part positioning devices and assembly fixtures;
   - Programmable Logic Controller (PLC) based devices, such as pick and place robots; and
   - stand alone industrial robots.

Faced with these constraints and existing equipment, a networked IBM-PC bus based computer control system was selected. Justification for this selection is summarized below:

1. IBM-PC bus based computers are inexpensive and available in industrial versions designed for the industrial environment.
2. A wide variety of peripherals, operating systems and software are currently available and at relatively low cost in comparison to similar software available for SUN, Apollo, and McDonnell Douglas workstations.
3. A variety of operating systems with varying capabilities including networking are available.

The most critical component of the system is the operating system (OS). OS features or lack of, will directly affect the systems overall performance and capabilities. Because of the wide variety of control applications which require sensor monitoring, serial communications, and other types of control, many unique features were required. The primary requirements of the operating system included:

- the ability to run software developed under the PC-DOS operating system;
- high speed networking capabilities with provisions for integrating 8086, 80286, and 80386 CPU based computers;
- high speed task switching capabilities with "real-time" support functions; and
- the ability to directly access a wide variety of peripherals from the IBM-PC bus.

Some of these peripherals included:

- stepper motor controller boards;
- digital I/O boards;
- Vision boards; and
- various graphics display systems

The ability to run off the shelf PC-DOS based software packages was important. This facility allows PC-DOS based CAD/CAM packages to be integrated into the production system. A wide variety of these CAD/CAM packages are now available providing a very cost effective and flexible solution to a companies CAD/CAM needs.

Flexibility over the long term requires planning at the outset. Provisions for integrating new computers into the existing network is a must for insuring future compatibility and expansion capabilities. New Intel 80386 cpu based microcomputers offer high performance low-cost alternatives to SUN or Apollo workstations. The abilitiy to accurately time operations under any circumstance to within micro-seconds is a must for real-time process control. Many sequencing operations require short accurate delays to

insure proper system functioning. The inablility to time short delays may hamper later production hardware additions and should be considered at the outset of the system design.

A number of operating systems were investigated including ZENIX, QNX, and others. QNX was selected as the operating system for the prototype cell for a number of reasons. QNX was the only package evaluated which provided all the necessary features listed earlier. QNX is considered to be a "real-time" multitasking operating system. There are currently over 65,000 installations world wide, many of which are real-time process control applications. Corporate user's include Northern Telecom, Litton, and FORD Canada. QNX provides high speed networking, with data transfer rates of three megabits/second. Task switching speeds are in excess of 2800 task switches per second based on an IBM-PC/AT. Direct access to hardware on the IBM-PC bus, including hardware interrupts is provided, and task timing is accurate to under fifty milliseconds. Serial communication features include access up to eight serial ports on any network node.

A PC-DOS emulation mode is provided allowing PC-DOS based

software to operate as a task under the QNX system. File transfer capabilities between operating systems is also provided. QNX is a Canadian product making technical support more accessible than American based products. Some prior familiarity with the product by support staff at this University also influenced the selection.

Linked closely with the operating system is the software development language. QNX provides compilers for the "C language", Basic, and assembler. The following section describes the selection of the MCE software development language.

3.3 Control Software Development Language Selection

When evaluating the supporting hardware for the CIM cell, a number of specific language features were considered necessary:

- the ability to access through software, peripheral devices such as digital I/O boards, which are mounted to the IBM-PC/AT data bus;
- the ablility to do rapid bit manipulation in decoding values returning from peripherals off the bus.
- the language must be supported by the QNX operating system and support all of the task handling functions associated with the operating system.

When selecting a development language many considerations

must be given to the hardware being controlled. Most languages will allow direct access of the data bus or allow bit manipulation. The more important questions are which is the quickest, easiest to work with, and requires the least cpu overhead. The ability to access the IBM-PC/AT bus was also critical.

Following evaluation of these necessary features only one language was considered satisfactory, "C". "C" includes many features in the areas of bit manipulation and input / output, which were a necessity for this type of application. The QNX operating system itself is written in "C" and thus the control software would provide a natural extension to this base. The next section describes in more detail, the CIM cell's various hardware components and how the control interfaces were developed for each.

IV. A PROTOTYPE COMPUTER INTEGRATED MANUFACTURING WORKCELL BASED ON IPC CONTROL

4.1 Existing CIM Cell - Hardware Components

The developed Manufacturing Control Environment (MCE) is based on a prototype CIM Cell currently in place at the Automation / Manufacturing laboratory at the University of Manitoba . The cell contains an "Automated Storage and Retrieval System" (ASRS) for storing raw materials as well as finished goods. All raw materials as well as finished products can be retrieved and stored in the ASRS rack. The cell has two CNC machines for performing all the material removal operations. In addition to the CNC machines, the cell also contains two robots for material handling and assembly operations. This particular CIM cell is designed for manufacturing single or composite parts using the CNC-mill and the CNC-lathe. Components produced in the CIM cell can be assembled in the assembly cell. Figure 4.1 presents the layout of the CIM cell with its various components. The components of the cell are arranged in a hexagonal format to provide for easy removal and positioning of equipment. The layout also helps in providing easy access to the machines by the central material handling robot. A summarized description of each of the cells main components is provided in the following sections.

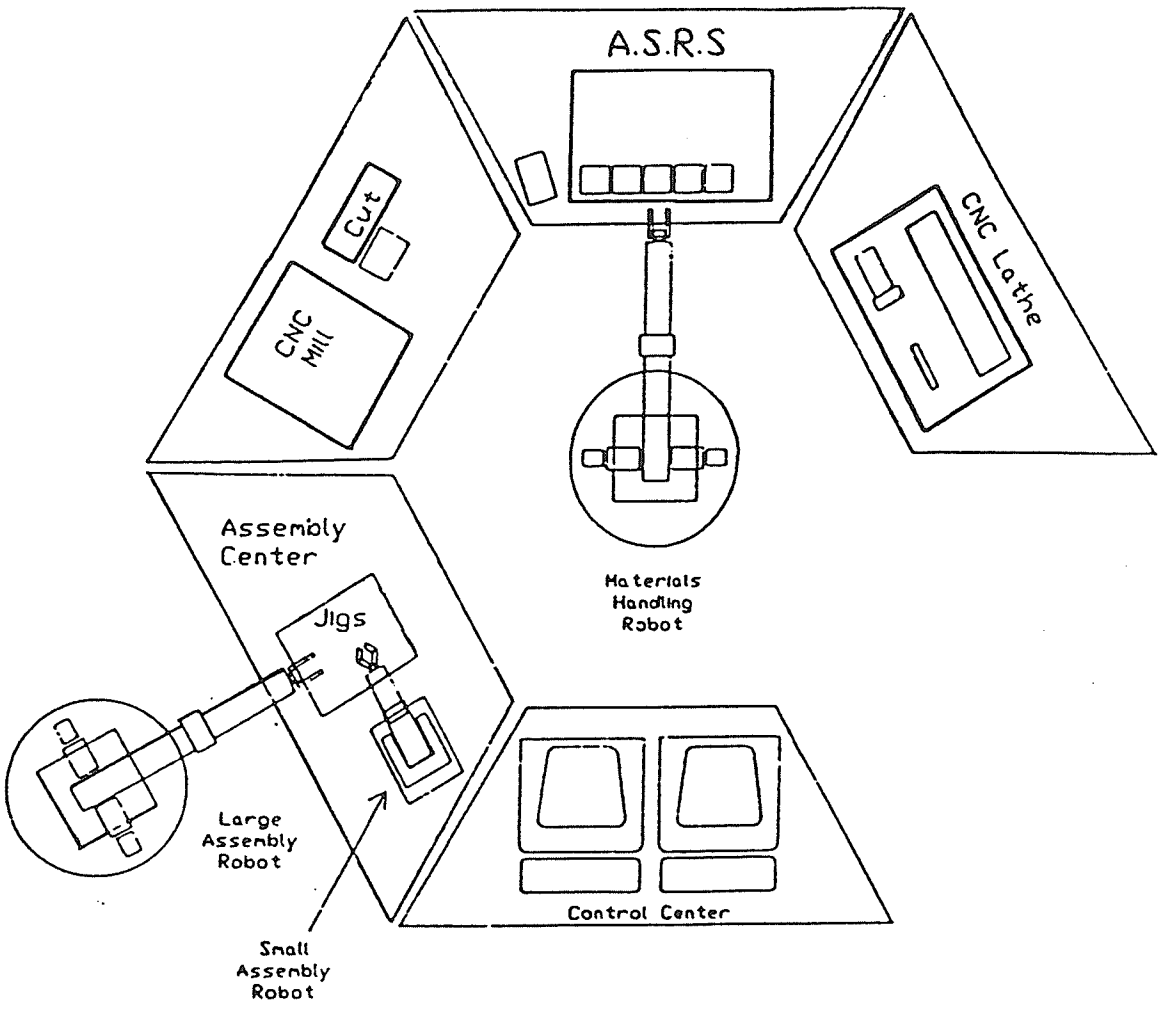


Figure 4.1 - CIM cell layout

4.11 CNC machines

The milling machine is a self-contained CNC machine manufacturedby 'TERCO'. It is a two and one half axis milling machine with interpolation capabilities in the XY plane and position control in the Z plane. Enhancements in the form of a pnuematic clamping system and part ejection mechanism have been added. These add on devices are controlled digitally. An RS-232C serial communication link allows downloading tool path files from the cell controller. Remote access to mill features are provided through digital I/O lines connected to the mill's keypad.

The lathe is a self contained CNC lathe made by 'EMCO'. A retrofit chucking system has been added to provide unattended machine loading and unloading. The retrofit chucking system, consists of a stepper motor, clutch, and a variety of sensors. Stepper motor control is provided by a PC-bus based stepper motor controller card by "Tecmar" and is also located in the cell control computer. The various sensors and clutch mechanism are controlled remotely via digital I/O lines. An RS-232C serial communication link has been utilized for downloading tool path files from the cell

controller.

4.12 Central Material Handling Robot

A five degree of freedom 6 kg capacity 'ASEA' robot provides material handling for the cell. Robot programming is performed directly through the controllers teach pendant and saved on a local floppy diskette. Communication with the cell controller is provided by a bank of eight digital input and output lines. Details concerning the communication interface between the cell controller and the central robot is presented later in section 6.56.

4.13 Automated Storage and Retrieval System (ASRS)

The ASRS is an indexing rack consisting of a number of gravity fed bins. The central material handling robot can load as well as retrieve parts from any bin. The rack motion is provided by a stepper motor controlled indexing mechanism. Figure 4.2 displays the ASRS Rack assembly.

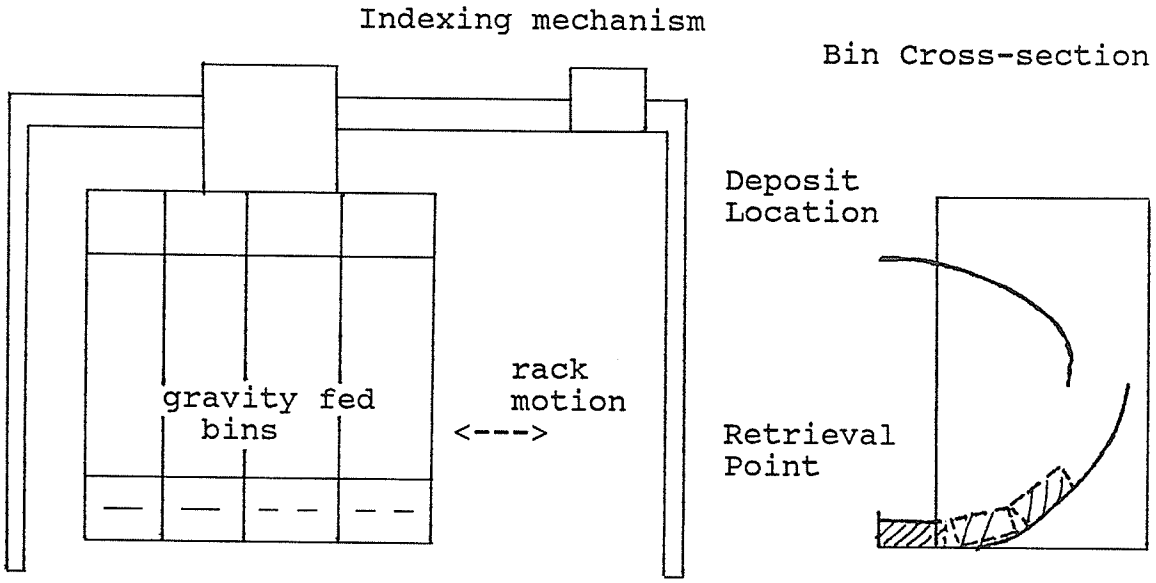


Figure 4.2 - ASRS schematic

Raw materials are contained in the first and third bins, with completed parts reserved for the second and fourth. The rack is modular and can be modified at any time. Provisions have been made for as many as two rows of ten bins per row. The rack is indexed so that the pick up point for the robot is at the same location at all times. This is a restriction imposed due to the lack of the sixth degree of freedom on the ASEA robot. It also reduces the amount of programming required for the central ASEA robot.

4.14 Cell Controller

The CIM cell controller is an 8MHz IBM-XT compatible MIND computer operating under the QNX operating system. Included within the cell controller is a variety of support hardware. These include:

1) Digital I/O board

An in-house developed 64 line digital I/O board with 32 inputs and 32 outputs, provides the base for controlling / monitoring of various sensors /devices. For example, pnuematic clamps are triggered by digital output lines, and sensors are monitored by various digital input lines.

2) Serial communications card

A serial port card by 'Intercom' containing four communication ports provides the main communication links to the CNC machines and assembly robot.

3) Stepper Motor Controller Card

A stepper motor controller card by 'Tecmar Inc.' is utilized for controlling two independent stepper motors. These stepper motors are used in two separate applications one for indexing the ASRS rack, and the other in the retrofit chucking system on the lathe.

The cell controller is operated using the QNX operating system and all control software was developed as part of this thesis.

4.2 System restrictions and integration difficulties

The CIM cell has been designed to be flexible but certain constraints on product size and machining processes do exist. The size and shape of products that can be produced are based on the capabilities of the CNC machines in the cell, as well as the capabilities of the material handling robots.

It is important to note that the CNC machines and ASEA robot described above are normally stand-alone devices. Each machine has its own controller and was intended to be programmed independently. They were not initially designed for remote control by an external computer. The CNC machines have serial communication interfaces but control of these interfaces required specific key strokes on the controllers keypad. To remotely control the mill and lathe, direct keypad manipulation was required. Prior to this project preliminary hardware interfaces were designed to handle this problem. Under this project optically isolated digital I/O interfaces were designed and implemented. More details on the CNC machine and robot interfaces are presented in sections 6.51 - 6.58. Having described the overall cell

structure the following chapter describes the development of the modular multitasking Manufacturing Control Environment (MCE).

V. A MULTITASKING, MANUFACTURING CONTROL ENVIRONMENT (MCE) - OVERVIEW AND OPERATION

5.1 Overview and Operation

The purpose of developing the Manufacturing Control Environment was to facilitate the integration of all major components of a manufacturing system. The primary components included in the currently developed system are:

- a Computer Aided Design and Manufacturing (CAD/CAM) system;
- inventory control of raw materials;
- remote robot and machine control; and
- production sequencing and simulation for efficient machine usage.

The MCE is composed of a series of software packages which provide the above features. Figure 5.1 displays the development flow of a product under the MCE. Modules are labeled based on a particular governing package also shown on the flowchart. The manufacturing process, from design to production as illustrated in Figure 5.1 is summarized in the following paragraphs.

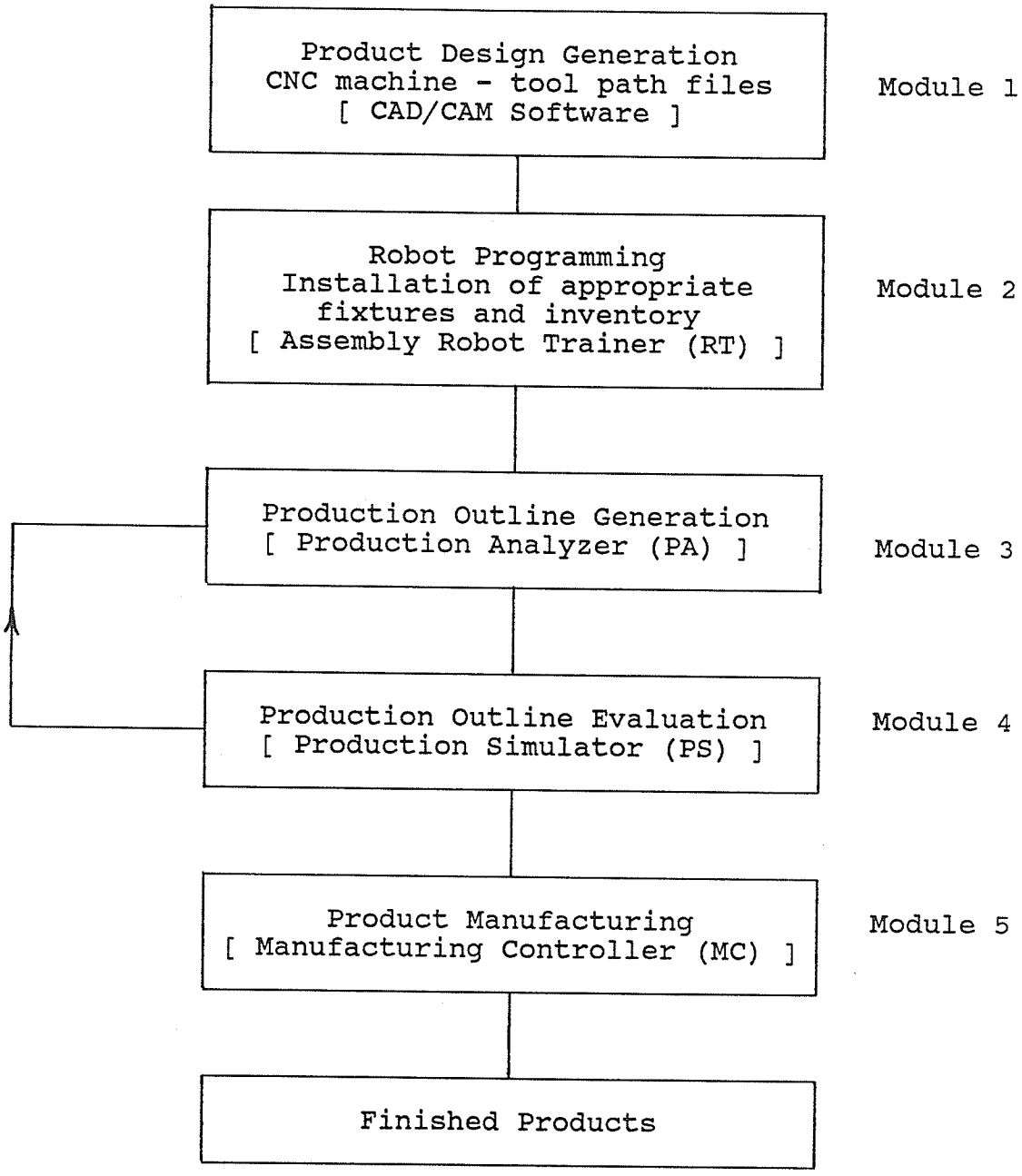


Figure 5.1 - Product Development Flowchart

The first phase involves creation of the tool path for the various parts to be machined within the cell. This is accomplished with the CAD system (Module 1 in Figure 5.1). Parts which are to be machined are developed in drawing "layers" with each layer representing a section to be machined by a particular tool. These layers are then imported into the CAM package and machine tools are assigned to each layer.

NOTE: It is not the intension of this thesis to explain in-depth the CAD system, rather the process of linking the CAD with the CAM system will be presented. Also, an outline of how all the phases of production are linked under this manufacturing environment, will be described.

Many CAD/CAM packages are currently available for IBM-PC/AT based personal computers. Because of their base being the IBM-PC/AT structure, these packages are generally less expensive compared to those based on larger computer systems such as Dec-VAX, Apollo and Sun workstations. It must be noted that, many high-performance features available on the larger systems may not be available on the PC based systems. However, many applications may not require these special features, such as shaded modeling, and 3-D real time on-screen object rotation etc. A PC based systems is also considerably less expensive.

The CAM system selected for the MCE was 'SmartCam' by Point Control Co.. The CAM system generates "Tool Path Files" which are later "downloaded" from the cell controller computer to the respective CNC machine. Once developed, these tool path files files are transferred, via the QNX/DOS file (QDF) system interface, from the DOS to QNX operating system environments.

The QNX operating system provides an MS-DOS operating environment which can operate concurrently with the QNX system on the same system (node). This feature allows MS-DOS based applications to be operated under the QNX environment. File handling between operating systems is virtually transparent. Data files available under either operating system, DOS or QNX, can be transferred back and forth.

After designing the product, raw materials are then selected. When this is completed, the "training" of the material handling robots for loading machines and assembling components can begin. It should be noted that training of the robots may not be required depending on the variation of the products from one production run to another. Once trained, similar production runs can be made

at any time thereafter without retraining the robots. In the case of the prototype CIM cell the central material handling robot is programmed through its own controller. A standard programming format has been established for this robot and is described in section 6.5.8.

The assembly robot is programmed using a developed package referred to as the "Assembly Robot Trainer (RT)" shown in the flowchart as module 2. The features and programming for this robot is described in greater detail in section 6.3. The output of the RT includes command files which are downloaded to the assembly robot as required throughout the production run. Upon completion of training the robots, and the positioning of appropriate fixtures in the system, the overall production outline can follow.

The production outline is the governing schedule for the production system. It is developed jointly using the Production Analyzer (PA), and the Production Simulator (PS) packages. These packages are modules 3 and 4 respectively in Figure 5.1. Development of the production outline, as well as the utilization of the pre-production simulation facility is described in sections 6.1 and 6.2 respectively. The PA system provides access to the CIM cell inventory database.

Any modifications to the inventory in terms of quantities, descriptions etc. are made through this package. The PA includes a facility for checking the production outline against the inventory database. This facility is included to insure that sufficient inventory is available, and all outlined procedures are valid before production procedes. Upon completion of the production outline, manufacturing can now procede.

The Manufacturing Controller (MC) subsystem, which is module 5 of Figure 5.1, analyzes the production outline and schedules the main operations. The MC, along with a number of "supervising tasks", control entire production process. All downloading of tool path files to CNC machines, signalling of devices, and controlling of robots are handled by the MC. Section 6.5 and all its sub-sections describe the MC along with its supervising tasks.

The MCE overlaps between the DOS and QNX operating system environments. A more detailed flowchart of the MCE is listed in Figure 5.2 and is intended to illustrate the primary data flow through the system. The next chapter presents the details/features of the various modules of the MCE.

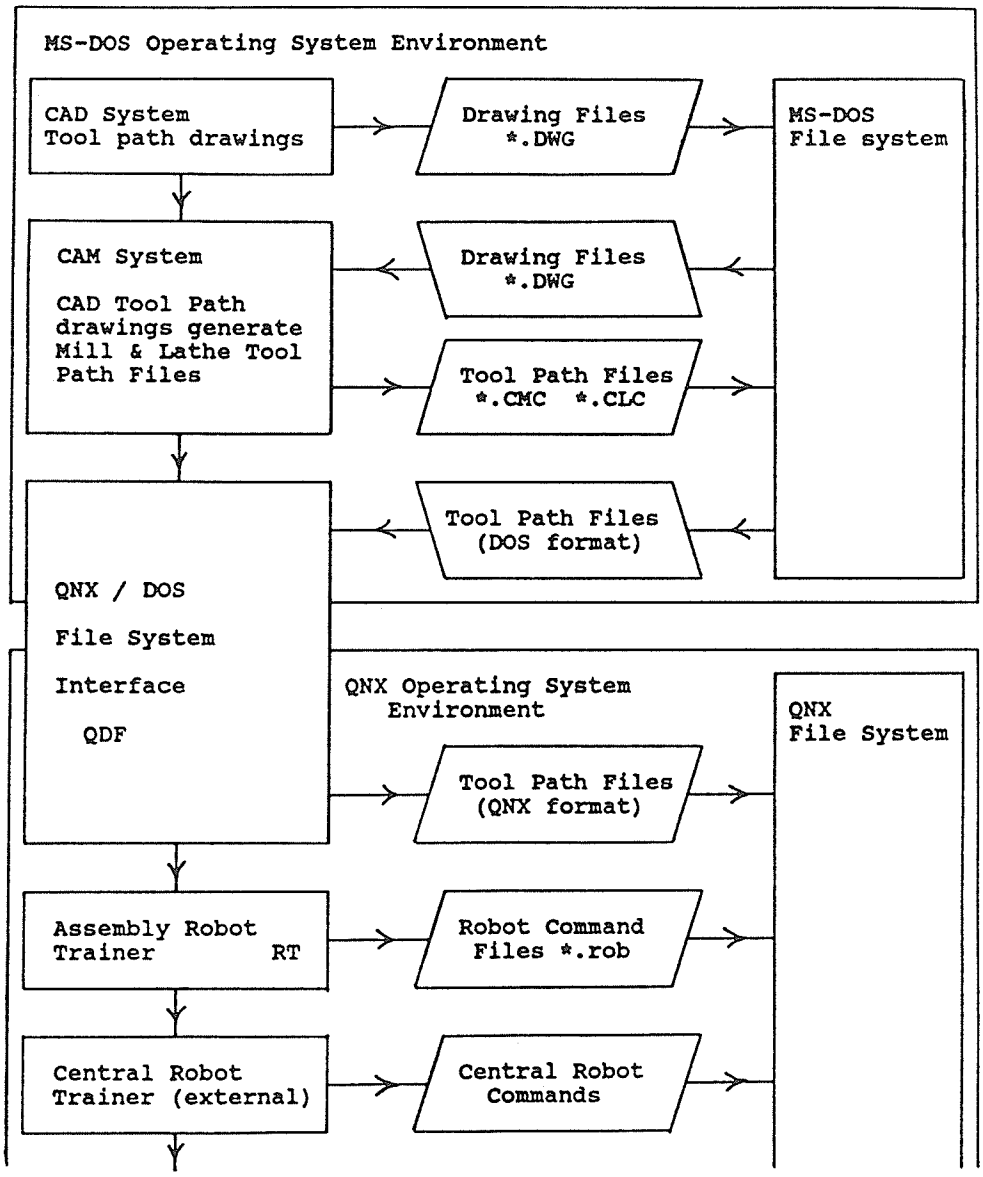


Figure 5.2 - MCE Information Flowchart

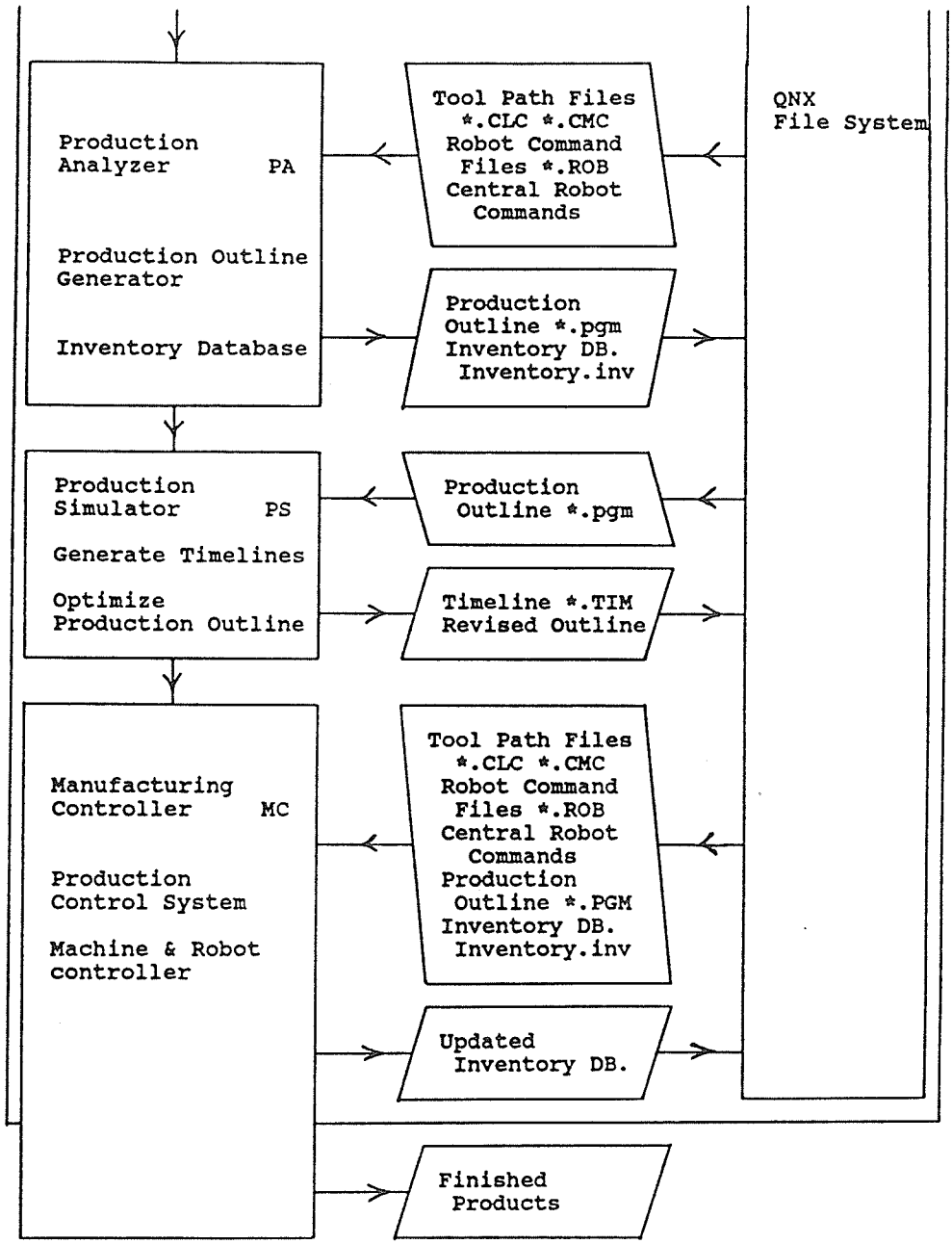


Figure 5.2 (continued)

VI. MCE SOFTWARE SUB-SYSTEMS

6.1 Production Analyzer (PA)

A complete production system should incorporate all the phases of product development from design through manufacturing. A software control system therefore, must be designed to take into account of these phases as well.

The Production Analyzer (PA) software sub-system ties the design phase to the manufacturing process. This single package is the link which brings together machining programs, (tool path files), database information (inventory), and robot motion programs. The purpose of the PA is to provide a base for generating production outlines and provide an access route to other MCE subsystems.

From the Production Analyzer main menu the user can access:

- the inventory database;
- assembly Robot Trainer (RT);
- Manufacturing Controller (MC); and
- the Production Simulator (PS).

6.11 Production Outline

The production outline is a structured description of a manufacturing run. This outline is one of the unique aspects of this manufacturing system. Activities to be performed in the cell are described in a general way, and are based on an "device-group" concept. The primary device groups include:

- the milling machine with its associated automation hardware;
- the lathe with its associated systems; and
- the assembly cell which includes the assembly robot and all associated assembly fixtures and mechanisms.

The device-group concept was developed in an effort to group cohesive sub-systems together. By grouping the mill with its associated automation hardware, control resources such as banks of digital I/O and serial communication ports could then be allocated. This group allocation meant that "supervisors" could be created to monitor these primary device groups with each one having its own resources without any overlap. The supervisor concept is presented in more detail under section 6.5.

When developing an outline, describing the sequencing of every operation in the cell is <u>not</u> required. Only an outline of the order in which parts are to be machined or assembled

is required. The sequence of part loading is determined internally by the system and is not defined by the user. Machine loading or unloading is based on a first ready first served basis. This allows efficient operation / preparation of a machine before the actual machining takes place.

Included within the PA is a text editor which is used to develop the production outline. The outline is composed of four sections:

- global definitions;
- mill operations;
- lathe operations; and
- assembly operations.

Included in the global definition section are:

- a list of tool path files to be used;
- the components of the system to be utilized such as the mill, lathe or assembly cell; and
- a list of the assembly robot command files to be used in this production run, if the assembly robot is being used.

Figure 6.1.1 displays a sample definition section, and includes explanatory comments. Comments such as those shown in Figure 6.1.1, may be included when generating an outline. Comments must be placed on separate lines and begin with a non-alphabetic character. Any line beginning with a non-alphabetic character is ignored when the program is executed.

The mill, lathe and assembly operation sections, are developed using similar constructs. Constructs include:

- a "loop - endloop" structure;
- "part#:" definition;
- "prod - stor" deposit identifiers; and
- section "begin - end" structures.

Figure 6.1.2 displays a sample mill machining sequence with its accompanying description. The looping construct can be used without limit but currently cannot be nested. The nesting feature could be added in the future. Machining sequences for the lathe are described in a similar format as shown in Figure 6.1.3.

Describing assembly operations, requires some understanding of the assembly task sequence. Parts required from storage during an assembly task must be listed in the order in which they enter the assembly. Figure 6.1.4 outlines a possible assembly operation sequence. The combination of all the sections described produces a complete production outline.

```
|*************************************************************|
|                                                             |
|  Global Definition Section                                  |
|                                                             |
|*************************************************************|
define
   begin
|*************************************************************|
|                                                             |
|  Production type 7 - mill, lathe and assembly cell          |
|                      being utilitized                       |
|                                                             |
|*************************************************************|
   prodtyp: 7
|*************************************************************|
|                                                             |
|  Mill global definitions                                    |
|                                                             |
|*************************************************************|
   mill
      nprogs: 1
|*************************************************************|
|                                                             |
|  hole1.cmc is the file name of the part program             |
|                                                             |
|*************************************************************|
      file: hole1.cmc
|*************************************************************|
|                                                             |
|  Serial communications port - four port serial card         |
|                               cable #1                      |
|*************************************************************|
      port: $term2

|*************************************************************|
|                                                             |
|  Lathe global definitions                                   |
|                                                             |
|*************************************************************|
   lathe
      nprogs: 1
```

Figure 6.1.1 Production Outline Defintion Section

```
|*************************************************************|
|                                                             |
|  qlathe1.clc is the file name of the part program           |
|                                                             |
|*************************************************************|
     file: qlathe1.clc
|*************************************************************|
|                                                             |
|  Serial communications port - four port serial card         |
|                               cable #2                      |
|*************************************************************|
    port: $term3

|*************************************************************|
|                                                             |
|  Assembly Cell global definitions                           |
|                                                             |
|*************************************************************|
   assemble
|*************************************************************|
|                                                             |
|  Robots: 3 - Assembly and central robots being utilized     |
|              in assembly activities                         |
|*************************************************************|
     robots: 3
     nprogs: 2
|*************************************************************|
|                                                             |
|  Assembly robot command files                               |
|                                                             |
|*************************************************************|
     file: slot1.rob
     file: slot2.rob
|*************************************************************|
|                                                             |
|  Serial communications port for assembly robot              |
|                                                             |
|*************************************************************|
     port: $mdm
  end
```

Figure 6.1.1 (continued) -
Production Outline - Definition Section

```
|*********************************************************|
|                                                         |
|  mill Production Sequences                              |
|                                                         |
|*********************************************************|
mill

   begin
      loop: 1

      nparts: 1

      stor
|*********************************************************|
|                                                         |
|  Part number 0001 - plain 35 mm diameter cylinder       |
|                                                         |
|*********************************************************|
         part#: 0001
|*********************************************************|
|                                                         |
|  Single sided milling operation sequence                |
|                                                         |
|*********************************************************|
         sgldbl: 1
|*********************************************************|
|                                                         |
|  mill loading command (load:) - 02                      |
|                                                         |
|*********************************************************|
         load: 2
|*********************************************************|
|                                                         |
|  Tool path file referenced above - eg. mill.cmc         |
|                                                         |
|*********************************************************|
         prog#: 1
|*********************************************************|
|                                                         |
|  mill unload command - 6                                |
|                                                         |
|*********************************************************|
         unload: 6
```

Figure 6.1.2 -
Production Outline - Mill machining sequence

```
|**********************************************************|
|                                                          |
|  Return machined part to original storage location       |
|                                                          |
|**********************************************************|
        stor#: 0001
     endloop
  end
```

Figure 6.1.2 (continued) -
Production Outline - Mill machining sequence

```
|*********************************************************|
|                                                         |
|  Lathe Production Sequences                             |
|                                                         |
|*********************************************************|
lathe
   begin
      loop: 1
      nparts: 1
      stor:
|*********************************************************|
|                                                         |
|  Part number 1000 - plain 35 mm diameter cylinder       |
|                                                         |
|*********************************************************|
     part#: 1000
|*********************************************************|
|                                                         |
|  Lathe loading command (load:) - 04                     |
|                                                         |
|*********************************************************|
     load: 04
|*********************************************************|
|                                                         |
|  Tool path file referenced above - eg. lathe.clc        |
|                                                         |
|*********************************************************|
     prog#: 1
|*********************************************************|
|                                                         |
|  Lathe unload to storage command - 12                   |
|                                                         |
|*********************************************************|
     unload: 12
|*********************************************************|
|                                                         |
|  Return machined part to original storage location      |
|                                                         |
|*********************************************************|
     stor#: 1000
  endloop:
end
```

Figure 6.1.3 -
Production Outline - Lathe machining sequence

```
|*********************************************************|
|                                                         |
|  Assembly cell operations                               |
|                                                         |
|*********************************************************|
assemble

   begin
  loop: 1
  ntasks: 1
|*********************************************************|
|                                                         |
|  assembly task - two components from storage (frstor)   |
|                                                         |
|*********************************************************|
  task
     nparts: 2
|*********************************************************|
|                                                         |
|  Central robot assembly command - 01                    |
|                                                         |
|*********************************************************|
     robot: 1
|*********************************************************|
|                                                         |
|  Assembly Robot - defined assembly program 01           |
|                                                         |
|*********************************************************|
     prog#: 1
     frstor: 0002
     frstor: 2000
  endloop
end
```

Figure 6.1.4 -
Production Outline - Assembly Description Section

Upon completion the production outline should be checked for procedural mistakes using the PA's internal pre-production analyzing facility. The analyzing facility checks for:

- typing or syntax errors;
- insufficient inventory levels; and
- invalid robot command definitions

The sequence of generating the outline and checking for errors, is an iterative process. Production can only proceed with a valid indication from the PA's analyzing facility. A valid production outline may be processed through the Production Simulator (PS). The PS will determine process completion times, product entry and completion times, and machine usage. From this information the user may decide to restructure the original outline to improve efficiency or proceed with production based on the current outline. A detailed description of the Production Simulator is provided in section 6.2. Outlines may be generated in many formats, and not all of the possible structures have been presented here. Further example structures are provided in Appendix A.

In summary, the purpose of the outlining facility is to provide a high level interface for describing the production process. This facility reduces the complexity in programming

the system and the amount of information required to operate it. The primary information required to operate the system reduces to:

- what needs to be machined on the mill;
- what has to be turned on the lathe;
- what components and in what sequence need to be assembled; and
- what is the required inventory.

6.12 Inventory Database

The inventory database is accessible only through the Production Analyzer. The database which is a direct access file, is composed of a series of records. Each record contains information about a defined part type. The part type, refers to a group of parts stored in the ASRS and is referenced as a four digit numeric value. The four digit value was arbitrarily selected based on the small number of inventory components. The part attributes are recorded with each part type and include;

- the quantity in storage;
- the part description;
- the part storage location (rack row and column); and
- the geometry, namely cylindrical or rectangular.

Additional fields inside each record have been allocated for future expansion. Only minor software modifications are required to access these additional fields. These additional

fields may be useful for describing material composition, size and/or shape. A menu driven facility is included in the PA for modifying any component of the inventory database and a typical screen display is shown in Figure 6.1.5.

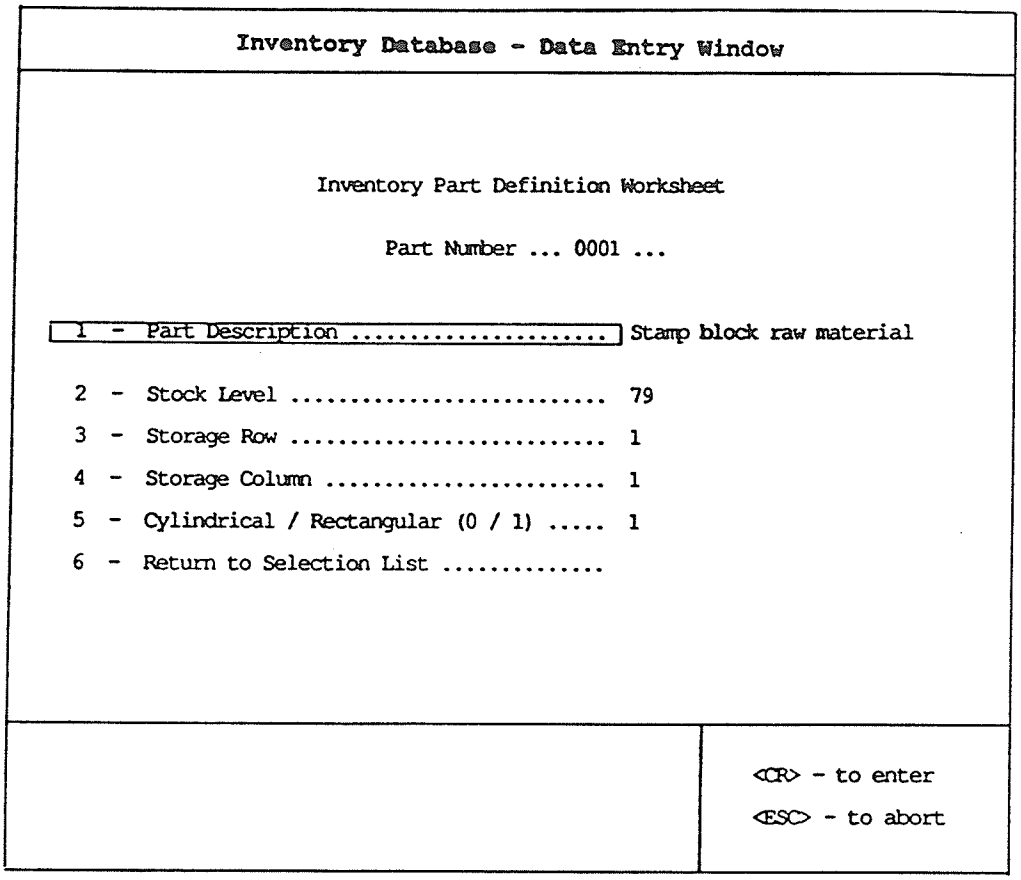


Figure 6.1.5 - Inventory Database

6.2 Production Simulator (PS)

The PS is a comprehensive simulation package for the QNX based Manufacturing Controller. It will simulate any production run that is possible in the actual production system. The simulator consists of two components, a data generator, and a display system. Simulation of a production outline is a two stage process. The production outline is first processed through the PS data generator. The data generator output is then displayed using the PS display facility.

6.21 The PS Data Generator

The PS data generator analyzes a given production outline using a simulation algorithm designed to duplicate the actual production system. The simulation algorithm is an actual subset of the main Manufacturing Controller (MC). The output from the simulation algorithm is a matrix of general production time information which is saved in a file. This file contains time and operation information about every product entering the system through to the completion of the production run. A sample output is listed in Appendix B.

The format of the data generated is on a part by part basis. Each line of information in the file is about a particular part which entered the system. Due to the complexity of the algorithm, the structure of the output data is not explained in detail here. A summarized description is provided in Appendix B.

6.22 The PS Display System

The display system is a component of the PS which provides a graphical portrayal of simulation information generated by the PS data generator. A sample display is shown in Figure 6.2.1. The display shows to scale, a time-line, in seconds of every part entering the system. Different shaded blocks indicate the various operations being performed on a given part. Dashed lines indicate idle times, and underscores indicate the part's existence in storage. The top line under the header, (labelled 'A' in Figure 6.2.1), displays the window time range the user is viewing.

The number 06260 in the top line header refers to the window beginning time in seconds from the start of the production run. The value 07060 refers to the time in

seconds at the end of the display window.

The column of numbers on the left side of the display window (labelled 'B' in Figure 6.2.1) describe the particular part time line below it. For example the number 10-0002-5 refers to the time-line below it. The series of numbers mean that the particular time-line is the tenth part to enter the production run. The raw material part number was 0002, and it is the fifth of the 0002 part type to enter the system. Additional features of the display system include the ability to zoom and pan over sections of the time line display. The display provides, at a glance, a visual portrayal of activities occurring in the cell at any given time.

The data generated from the production simulator may be manipulated to determine many important system parameters. Some of these include:

- machine utilization and idle time;
- overall system efficiency; and
- robot utilization and idle time.

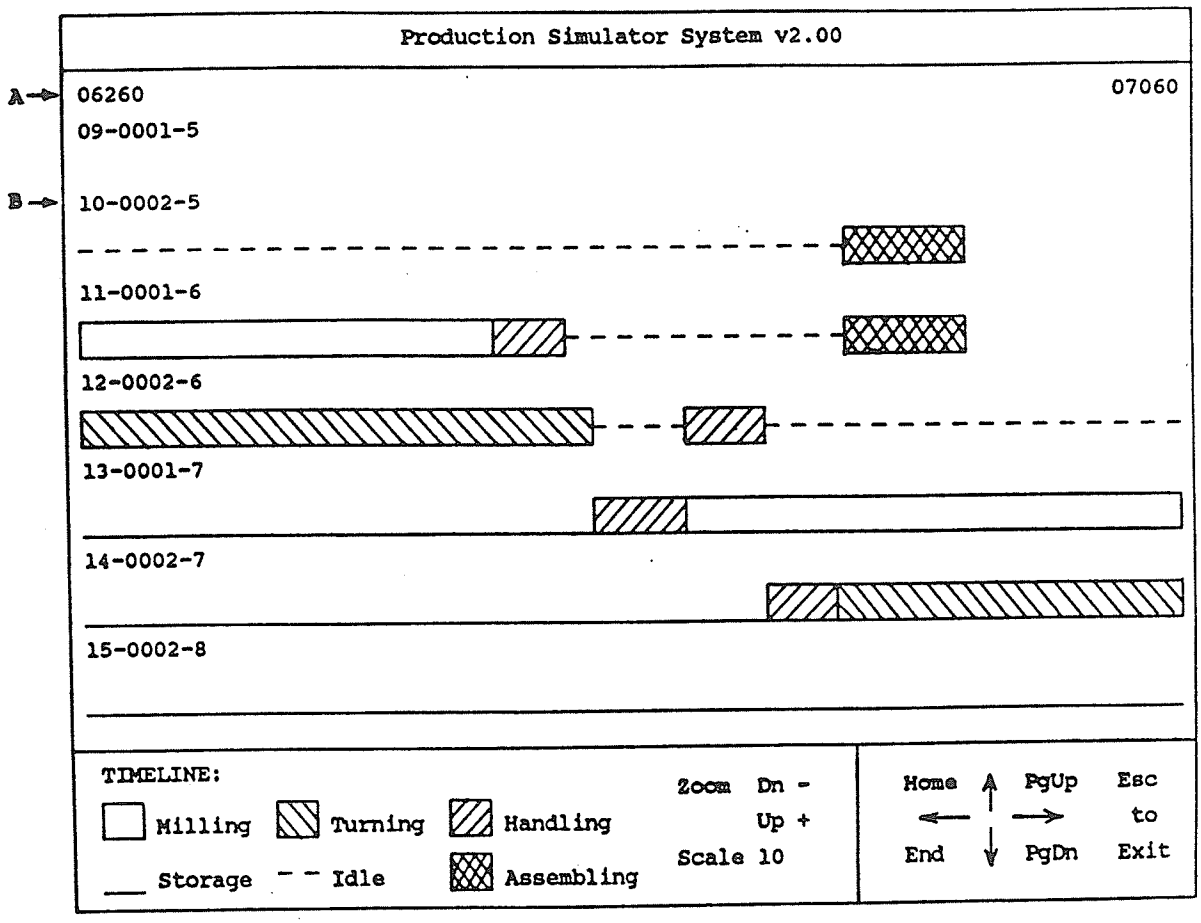


Figure 6.2.1 - Production Simulator Display System

6.3 Assembly Robot Trainer (RT)

The assembly Robot Trainer (RT) is an interactive robot training facility for developing assembly tasks. The assembly robot is linked with this module and can function under the MCE. Programs developed under this package are brought into the production system through the production outline as explained in section 6.1.

The assembly robot in the prototype CIM cell is a five degree of freedom 'Teachmover Microbot'. Some important features of the microbot include;

- a teach pendant for keypad programming;
- an RS-232C serial communications interface for communications to a host computer such as an IBM-PC/AT;
- a seven bit digital I/O output interface;
- a seven bit digital I/O input interface; and
- an internal command language interface.

Using the microbot's own command language, the robot can be controlled by a host computer. A complete description of the command language is not presented in this thesis. Further explanation of the microbot programming structure is provided in the microbot programming guide.

The RT provides an environment for developing an assembly

task program, which is subsequently incorporated into a production outline. The RT is used to record incremental robot motions using the robot teach pendant in conjuntion with the microbot's internal command language. Recorded motions, along with additonal features including, time delays, digital I/O input sampling etc. create the assembly task program. Conditional features, such as sampling a digital input, can be used to execute sub-tasks, which are generated similarly, and are later incorporated using the IF and EXECUTE commands. Figure 6.3.1 presents the available training options.

The general assembly task development process involves:

- defining a home position by selecting menu option A;
- selecting menu option B to begin recording an incremental step (Figure 6.3.2);
- utilize the teach pendant to move the microbot to a new position and recorded this movement in a file, in the format of an executable STEP command.
- To add more incremental motions and the same procedure followed.

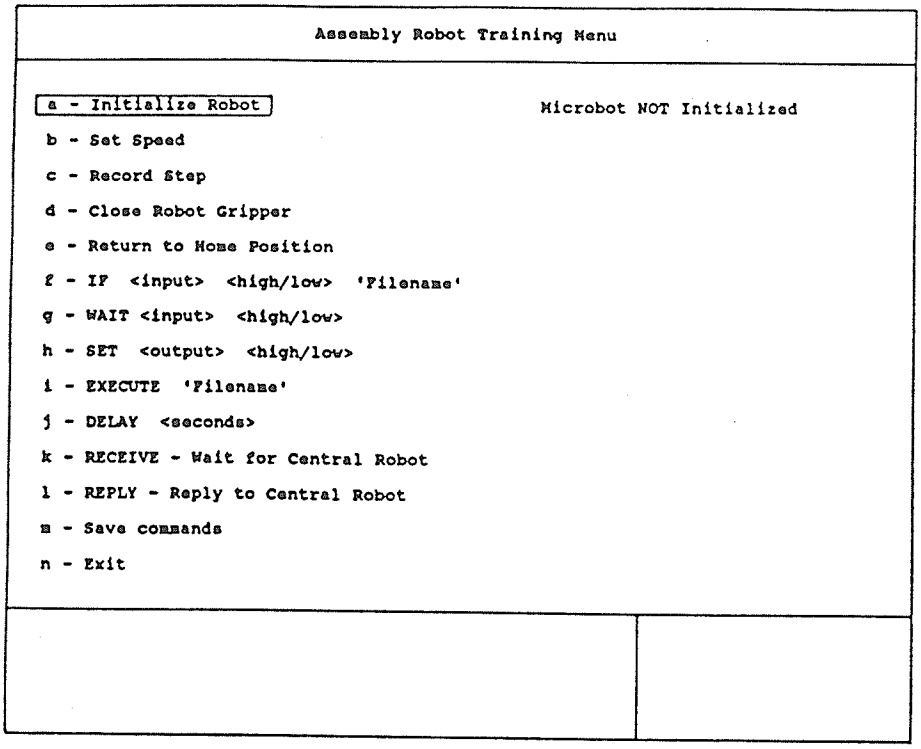


Figure 6.3.1 - Assembly Robot Training Menu

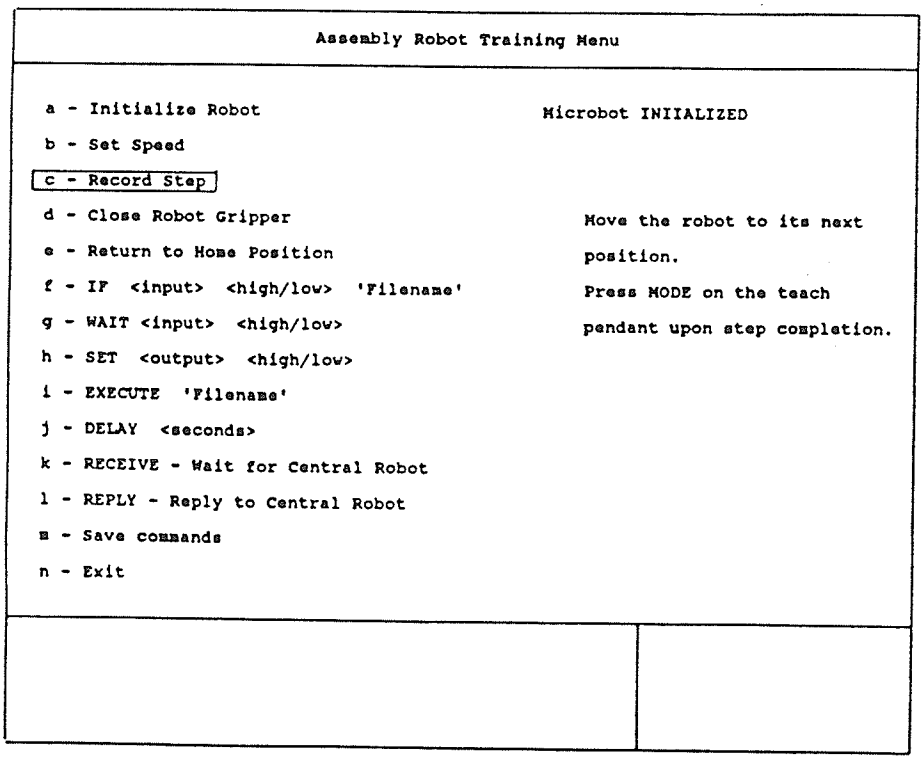


Figure 6.3.2 - Incremental Step Recording Process

Additional features which may be included in a robot task are:

- the ability to wait for an external input to be set high/low (menu option g);
- check if an input is high / low and based on this status execute another robot task (menu option f);
- set a digital I/O ouput line high / low (menu option h);
- pause for a specified period of time (menu option j);
- wait for the central material handling robot to signal the assembly robot to continue (menu option k); and
- respond to a central robots signal (reply). This usually is used as an acknowledgement to the central robot for a received message.

Other features available include:

- varying the robots speed during a task (SET SPEED);
- closing the robot gripper to a predefined pressure; and
- to return directly to the defined home position.

Features one through six are not intrinsic robot commands. These are interpreted and executed by the assembly robots supervising task (section 6.54) during a production run.

The RT also incorporates a testing / playback feature which can be used to replay a generated task outside a production run. This feature is necessary in final testing and timing of various phases of an assembly task. The playback facility will prompt the user to simulate or monitor the various external inputs affecting the task. For instance, if an assembly task requires a conditional read on an input, such as:

IF INPUT 1 IS HIGH EXECUTE "This sub-task".

The simulator will ask if the user wishes to simulate this true / false or actually sample the input.

Two important features which require further explanation are the RECEIVE and REPLY facilities. They are used as communication links between the assembly robot and the central material handling robot. The MC, described in section 6.4, monitors a number of device supervising tasks during a production run. Three of these tasks include:

- the assembly cell supervisor;
- the assembly robot supervisor; and
- the central robot supervisor.

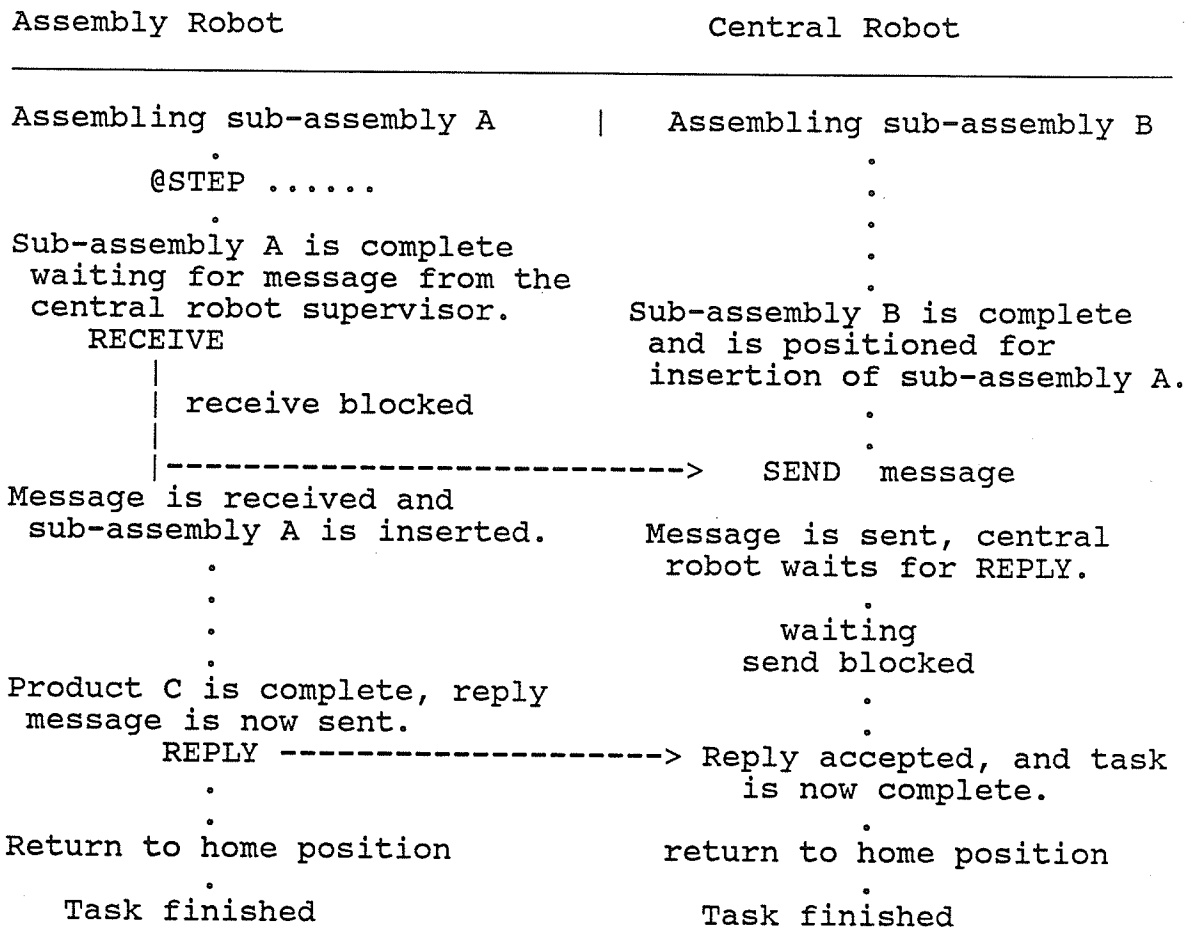
The assembly robot supervisor handles the on-line, or continuous communication, control of the microbot, as well as interpreting the additional command features provided by the RT. The RECEIVE command when interpreted by the assembly robot supervisor means, stop and wait for an incoming message from the central robot supervisor. This waiting for a message is known as being RECEIVE blocked. When a message is sent from the central robot supervisor, the assembly robot becomes UNBLOCKED and can continue its task. The central robot, however, does not continue until reception of

the message has been acknowledged via the REPLY command. The central robot is REPLY blocked until this reply is sent. The inter-robot communications link, via their supervising tasks is the basis for developing joint robot assembly applications.

For example, considering an application where two sub-assemblies, A and B are assembled separately, A by the assembly robot, and B by the central robot, and the product C is a combination of the two sub-assemblies.

sub-assembly		sub-assembly		product
	+		->	
A		B		C

The product, C, requires the central robot to position sub-assembly B and have sub-assembly A inserted into it by the assembly robot. For simplicity, all assembly components will be considered available within the assembly cell, and that no additional parts are required from storage. The assembly robot, and the central robot assembly tasks would be outlined as follows:



This describes the general features of the RT and its role in the manufacturing system. The process of integrating an assembly task into the production outline was presented in section 6.1.

NOTE: The central robot assembly task program is not outlined in detail here, only the process is shown. More information on the central robot programmming structure is provided in section 6.58

6.4 Manufacturing Controller (MC)

6.41 The MC Control Structure:

The MC is the hardware control hub for the manufacturing system. The responsibilities of this controller include:

- delegating duties to the mill, lathe, and assembly device groups; and
- scheduling all main production tasks.

A number of requirements were considered when designing the hardware control system:

- the control software had to be modular to provide flexibility for future modification, and provide ready access to control hardware;
- the production display system, in operation during manufacturing had to clearly portray activities occuring in the CIM cell at any given time; and
- the system was required to be self-sequencing, with machine serving being performed on a first ready first served basis.

The first priority when designing the software control system, was to make the system modular, flexible, and process independent. To achieve this goal, the "device group" sub-system concept outlined earlier was utilized. Major components and groups of related components, based on their activity relationships, were considered as device groupings. These groups include the devices outlined below.

1) Mill device group - CNC mill, pnuematic clamping, part ejection, and part orientation mechanisms.
2) lathe device group - CNC Lathe and retrofit chucking mechanism.
3) Assembly device group - Assembly robot, assembly fixtures, and sensors.
4) Material handling group - Central material handling robot.
5) ASRS device group - Inventory and indexing mechanisms.

The mill, lathe and assembly device groups are considered to be "primary" device groups. Activities in the cell are centered around these groups. The material handling and ASRS groups are considerd device servers because of their functions. For each of these functional device groups a "supervising" or group of tasks was developed. These supervising tasks control each of the functional group's devices. General purpose software / hardware interfaces had to be developed to provide communication between the various machines and sensors. All the primary system and sensor interfaces are computer controlled through peripherals attatched to the computer data bus. For this particular system the major computer / device interfaces included:

- a four port serial communications card by 'Digi-Comm' to provide RS232-C serial communications;
- a stepper motor control interface by 'Tecmar'; and
- an in-house developed digital input / output interface.

The device-group concept outlined earlier was developed as a means of allocating these various computer interface

resources. Each supervising task or group of tasks, and their associated device group, have their own interfaces allocated to them. For instance, The mill supervising tasks have the following resources allocated to them:

- one serial communication port from the 'Digi-Com' four port serial card;
- one bank of eight digital I/O input lines; and
- one bank of eight digital I/O output lines.

These resources are never addressed by any of the other device group supervising tasks. This breakdown of resources insures that different operating tasks do not conflict with one another during the normal operation of the system. Figure 6.4.1 is a schematic of the controller device interfaces and the resources allocated to each of the device groups.

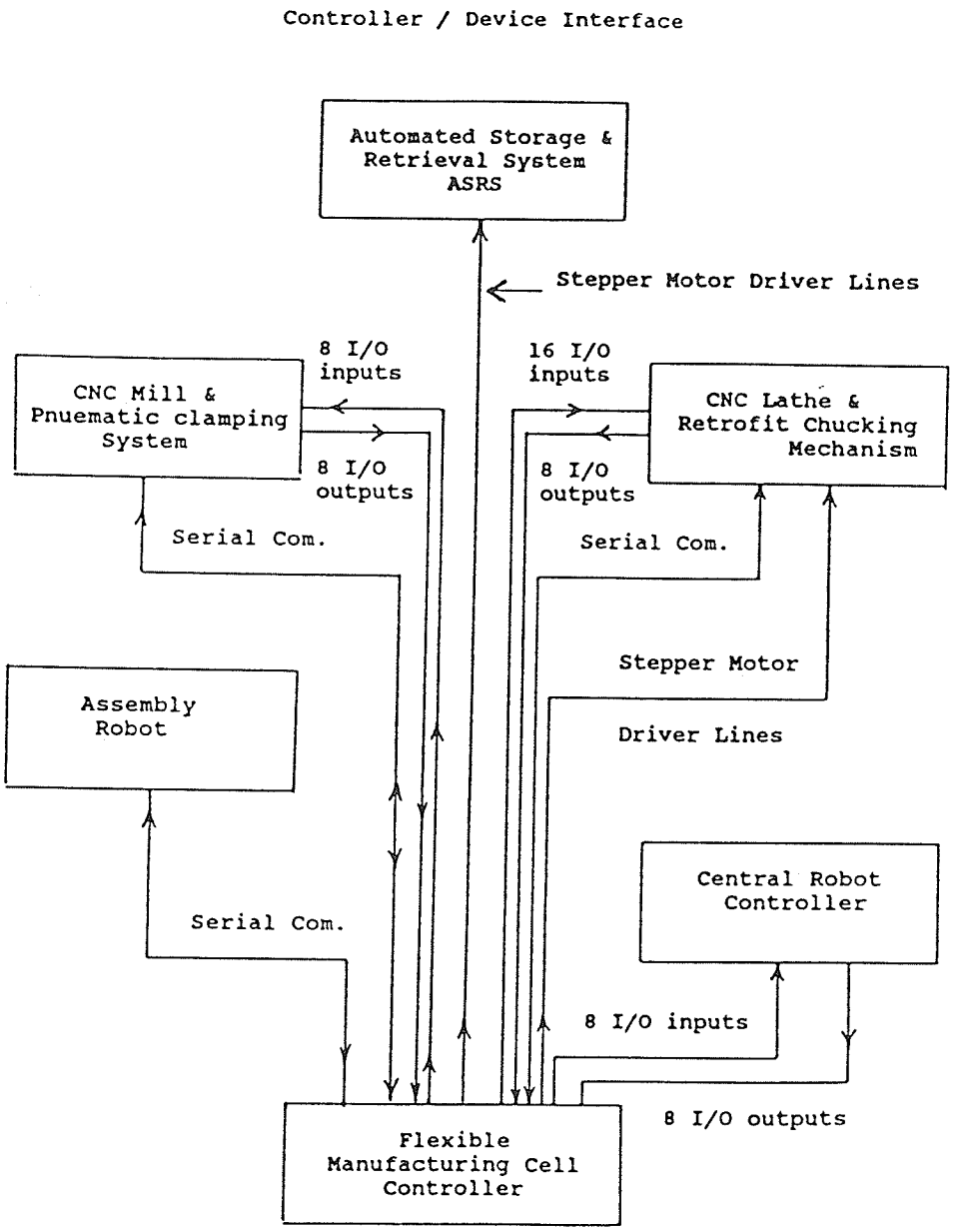


Figure 6.4.1 - Controller Device Interfaces

The MC has the responsibility of scheduling production tasks for the primary device groups. A production task consists of a series of related operations which are performed by a device group supervisor. for example, if the task requires machining a part with number 0001 on the mill and then returning it to storage as part number 0002, the task involves:

- uploading the approprate tool path file to the mill;
- retrieving the raw material from the ASRS;
- loading the milling machine with the retrieved part;
- machining the part;
- unloading the finished part;
- placing the finished part back in storage; and
- updating the inventory database.

The primary device group for this example is the mill group. Primary device groups and device group servers always overlap. In this particular case the ASRS, and central robot device group servers are involved. The MC breaks down a production outline, and formulates tables of production tasks, one table for each of the primary controllers. As a primary controller's task is completed another one is submitted until its duties are completed. Internal monitoring of inventory requirements for each task is performed during the production process, in real-time. An example of this type of inventory monitoring, is in monitoring temporary part storage levels in the assembly

cell. Assembly task scheduling does not occur until sufficient inventory levels are available.

Figure 6.4.2 shows schematically the task communication flow diagram for the supervising tasks. The manufacturing controller "creates" the approriate supervising tasks during the software initialization phase. The manufacturing controller sends messages to these tasks when a certain series of operations are required. For example, the MC sends a message to the mill download supervisor, indicating that a particular tool path file should be downloaded to the CNC mill. This supervising task then takes the appropriate actions including preparing the serial communications port and initiating the file downloading process. Each supervising task is resposible for serving a particular device group.

The mill operating supervisor has the responsibility for monitoring the mill during machining operations, as well as requesting the services of the material handling robot to load and unload it. A service request is issued in the form of a message sent from one supervisor to another. In the case of the mill being loaded, the mill operating supervisor sends a message to the central robot supervisor indicating

that a particular part is to be loaded. When the central robot is free, it sends a message to the ASRS supervisor indicating that a particular part should be positioned for pickup. Once the part is positioned the ASRS supervisor signals the central robot supervisor to retrieve the part. The central robot acknowledges, retrieves the part, loads the mill, and finally signals the mill supervisor that loading is complete. It is this inter-supervisory task messaging which links the cell controller to the various device group supervisors. Further details on the device group servers, and primary controllers is presented in the following sections.

6.42 The MC Display System:

The display system for the MC is based on an overlapping window design. Each supervising task has its own unique "window" in which information is written. Overlapping windows display communication links between supervising tasks. A series of screen displays, presents the task organization, for a typical situation. Figure 6.4.3 displays two supervisors downloading tool path files to their respective device groups. Figure 6.4.4 shows a progressive stage where the lathe is turning the insert piece and the

mill is being loaded. Note the relationship between the overlapping windows on the mill supervisor. Figure 6.4.5 shows a subsequent stage in production where parts are being machined on both the mill and lathe, and an assembly task is being performed using both the central and assembly robots. In the display shown in Figure 6.4.6, a part is being removed from storage for direct usage in the assembly cell. The display system presents the state of the manufacturing system at any point in time in a form which is clear and understandable.

The windowing system was developed under this project as well as all higher level graphics tools. Graphics primitives were provided with the QNX 'C' compiler.

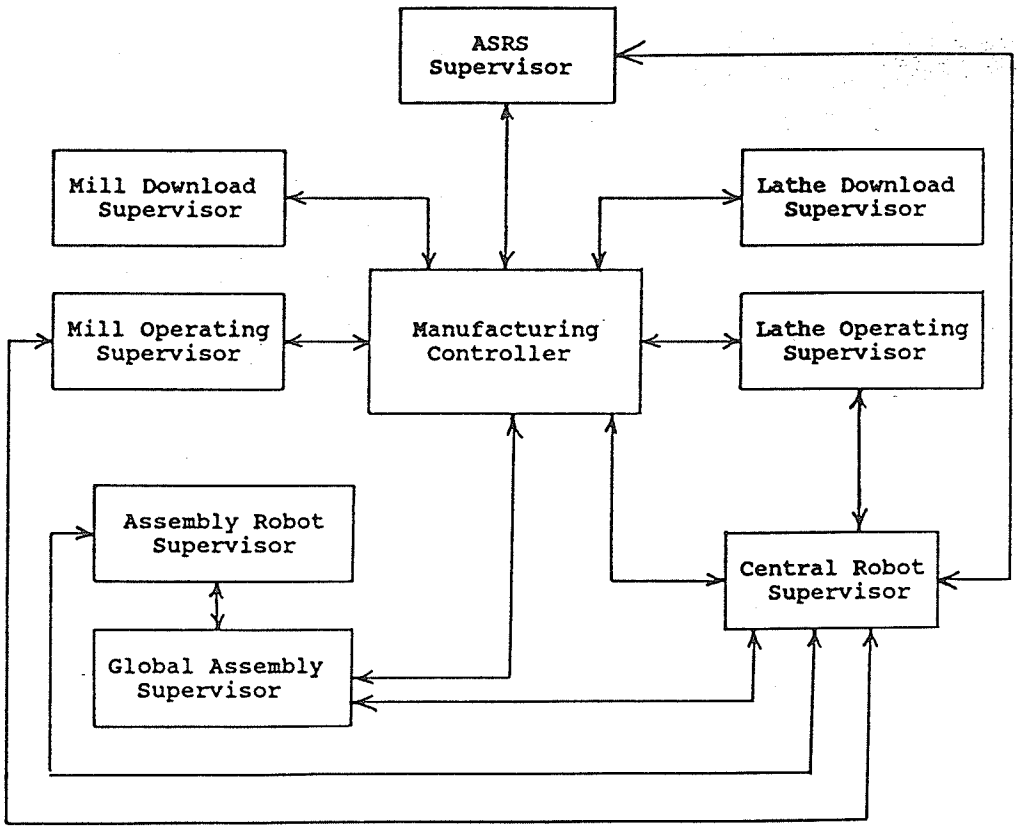


Figure 6.4.2 -

Manufacturing Controller - Task Communication Flow Diagram

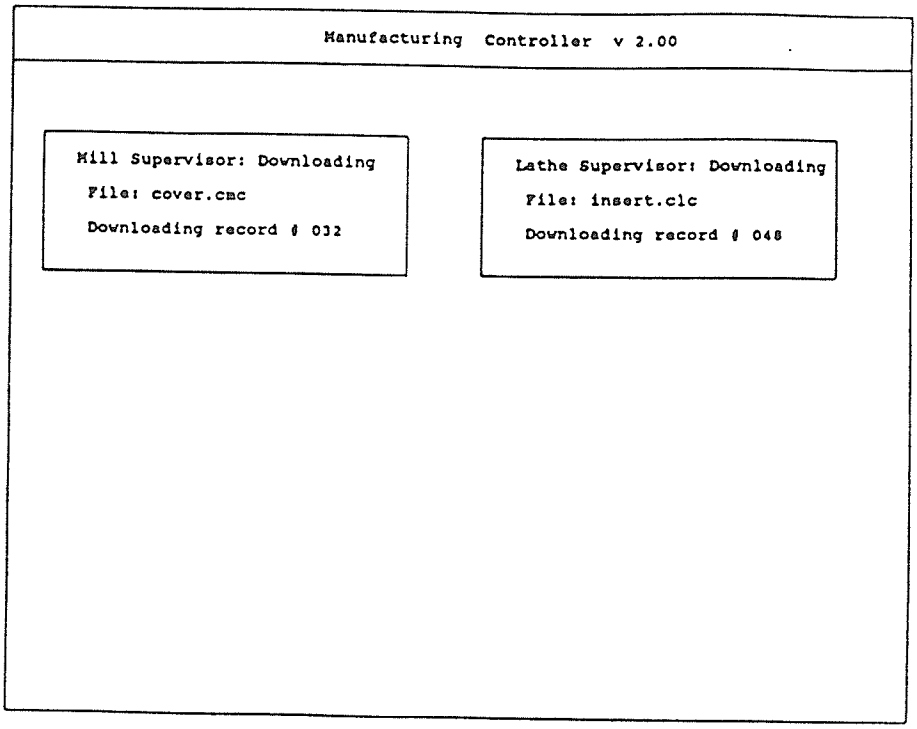


Figure 6.4.3 - Information Downloading Display

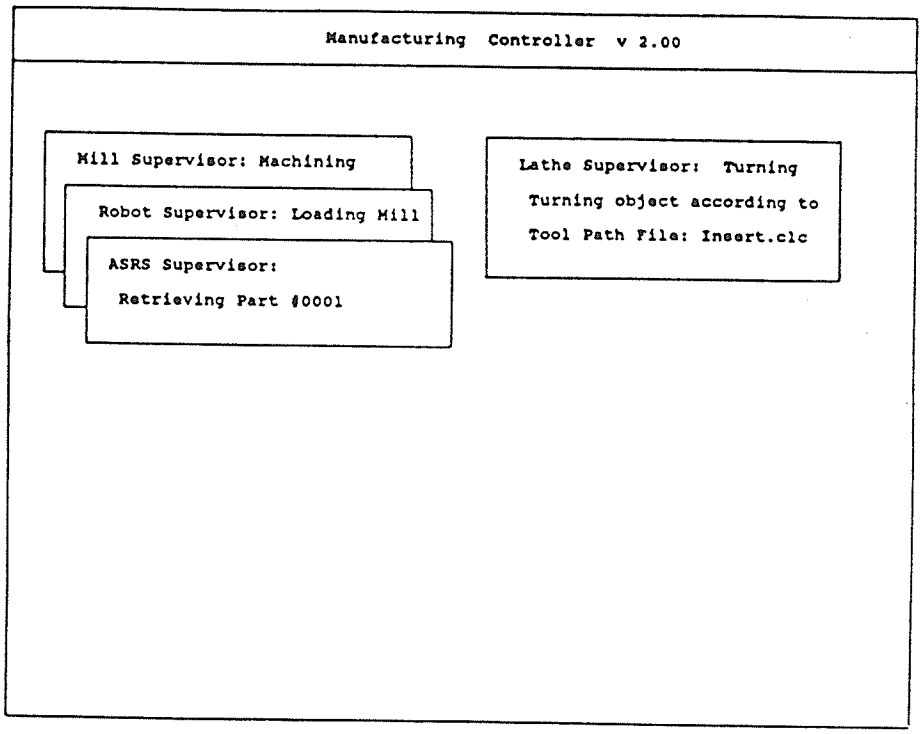


Figure 6.4.4 - Milling machine loading

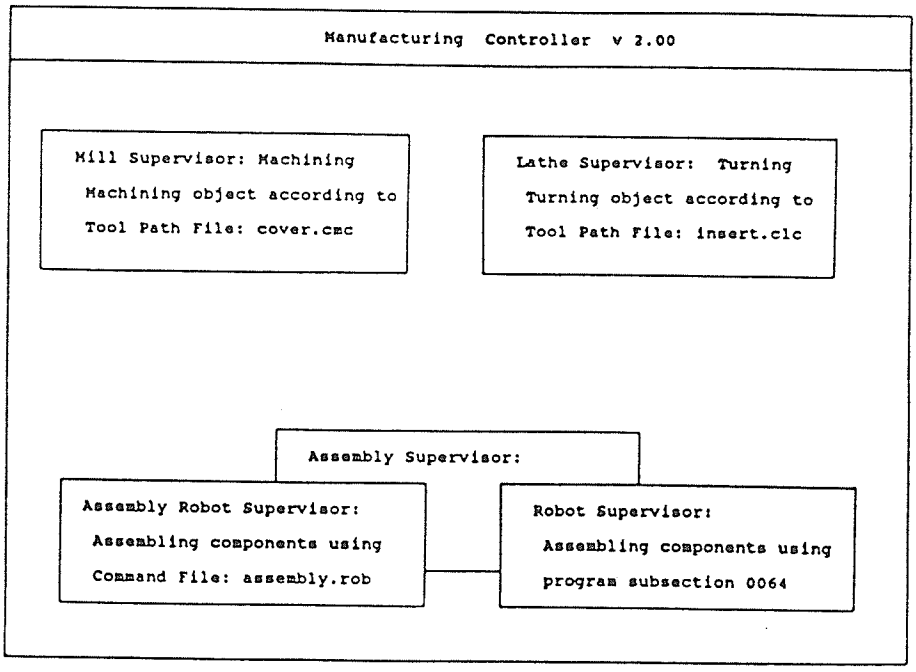


Figure 6.4.5 -
Mill, lathe and assembly tasks being performed

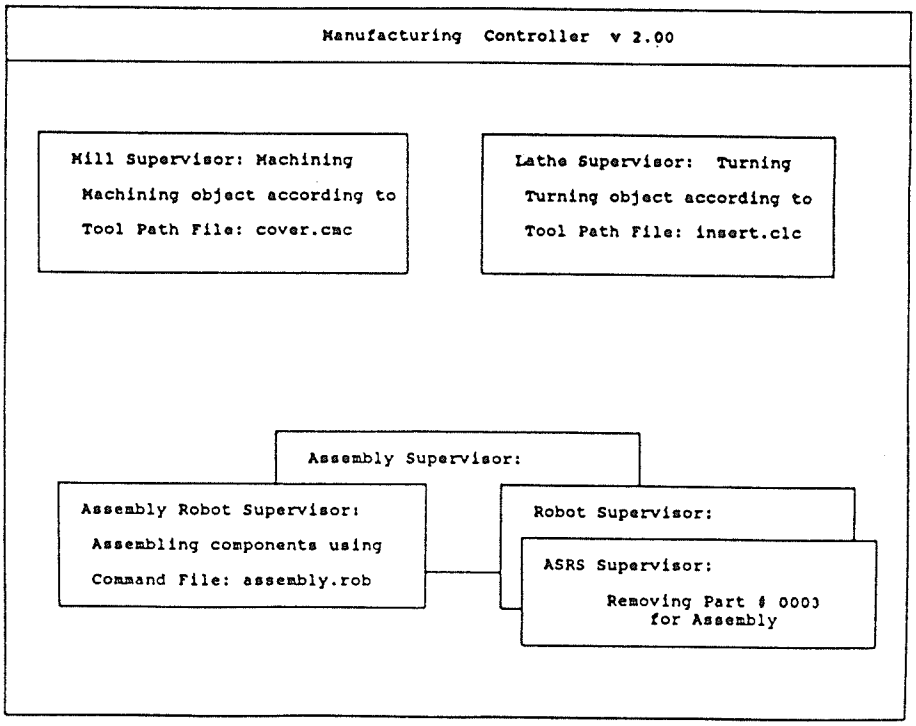


Figure 6.4.6 -
Retrieval of a pre-machined part form storage

6.5 Manufacturing Controller Supervising Tasks

The Manufacturing controller supervising tasks are used as links to the CIM cell hardware. The MC itself does not access any of the cell's sub-system components, such as digital I/O lines. Only the device group supervisors perform this function. Each supervising task is allocated certain independent resources such as serial communication ports, and banks of digital I/O. No other task operating on the system has access to these particular resources. In the following sections, each supervising task is briefly outlined and special features described.

6.51 CNC Mill Downloading Supervisory Task

The purpose of this task is to control downloading of tool path files to the CNC mill. Depending on the file size, up to four tool path programs may be resident in the CNC mill's memory at any time. This was a restriction imposed by the 'Terco' mill. This supervising task will "mount" any program in a particular part of the machines memory for later reference.

The computer hardware interfaces utilized by this task include:

- an RS-232C Serial communications interface required for downloading tool path files to the mill; and
- eight digital I/O output lines, required for remote operation of the mill's keypad, engaging pnuematic clamps and part ejection mechanisms

Serial communications between the cell controller and the mill are performed at 300 baud. This again was a constraint imposed by the mill's controller. To configure or start the milling machine, various keys on the mill's keypad had to be pressed. To engage these keys in the proper sequence a number of computer digital I/O output lines were tied into the mill controller. Keypad reseting, program starting, and initialization are all initiated remotely via this interface.

6.52 CNC Mill Operating Supervisory Task

The purpose of the mill operating supervising task is to monitor machining on the CNC mill, and to control the processes of loading and unloading. Given a resident part program within the mill controller memory, this supervisor will:

- engage the services of the central robot and ASRS, to load, unload and reload the mill when appropriate;
- monitor various conditions including part program completion; and
- trigger all part clamping and ejection mechanisms.

The hardware interfaces utilized by this task include:

- eight digital I/O output lines, used as an interface to the mill keypad and for engaging the pnuematic clamping mechanisms; and
- eight digital I/O input lines, for monitoring program completion and proper part ejection.

The digital I/O output lines are the same ones as those utilized by the mill downloading supervisory task. These tasks are never operating concurrently and hence do not pose any conflict.

The mill operating supervisor can perform a variety of tasks. Single or double sided milling can be performed at any time. Parts may also be machined, deposited into storage or transferred directly to the assembly cell.

6.53 CNC Lathe Downloading Supervisory Task

The purpose of this task is to control downloading of tool path files to the CNC lathe. Unlike the CNC mill, the lathe may only have a single resident tool path program. When a different program is required this task will configure the lathe and download the appropriate program to it.

The computer hardware interfaces utilized by this task include:

- an RS-232C Serial communications interface required for downloading tool path files to the lathe; and
- eight digital I/O output lines, required for remote operation of the lathe keypad.

Similar to the mill's keypad interface, a number of digital I/O output lines have been tied into the lathe controller keypad, allowing lathe keypad keystrokes to be initiated remotely. Keypad reseting, program starting, and initialization are all initiated remotely via this interface. Serial communication between the cell controller and the lathe is also performed at 300 baud, as a result of restrictions imposed on the system by the lathe controller's hardware.

6.54 CNC Lathe Operating Supervisory Task

The purpose of this task is to control and monitor part production on the CNC lathe, including machine loading and unloading. In addition to regular machine serving operations such as part loading etc., a retrofit chucking system is also controlled. Pieces inserted into the lathe chuck by the central robot are remotely clamped using a variety of sensors and control hardware. Some of these components include a stepper motor, a proximity sensor and various switches.

The computer hardware interfaces utilized by this task include:

- sixteen digital I/O output lines which are used as an interface to the lathe keypad, for engaging pnuematic clamping devices, and triggering components of the retrofit chucking system;
- eight digital I/O input lines linked to switches for monitoring part program completion and the retrofit chucking mechanism; and
- a stepper motor interface controller board required for retrofit chucking mechanism.

Eight of the digital I/O output lines are identical to those utilized by the lathe downloading supervisory task. These tasks are never operating concurrently and hence they do not pose a conflict.

The lathe operating supervisor can perform a variety of tasks. Parts may be turned on the lathe, returned to storage, or placed directly in the assembly cell.

6.55 Assembly Cell Supervisor

The purpose of the assembly cell supervisor is to coordinate activities in the assembly cell, including instructing the assembly robot, via the assembly robot supervisor.

Some of the software features this task provides include:

- simultaneous operation of multiple robots in assembly

activities; and
- an inter-supervisor communications interface between the assembly and material handling robots.

Assembly tasks may be performed individually by either the assembly robot or, the central material handling robot. It is also possible to have both robots working together. The assembly cell supervisor distributes commands to the two lower level device suspervisors (central material handling robot supervisor, and/or assembly robot supervisor). The assembly robot supervisor is instructed by the assembly cell supervisor only.

Assembly instructions are passed from the Manufacturing Controller to the assembly cell supervisor. These messages are then interpreted and delegation of duties then occurs. The assembly supervisor then requests the services of the central and assembly robots based on their respective duties.

6.56 Assembly Robot Supervisor

The purpose of the Assembly robot supervisor is to provide on-line control of the assembly robot, while performing an assembly task.

The assembly robot supervisor interprets robot command files generated by the assembly robot trainer (RT) and passes messages to the teachmover microbot. Command file functions including inter-robot messaging and device signalling, using the assembly robot's input/output capabilities, are all performed in real-time. The only communication interface utilized by this task is an RS-232C Serial communications interface. This interface provides a communication link from the cell control computer to the microbot robot.

The assembly robot is the only device in the CIM cell which requires extensive on-line, cpu intensive control. The microbot receives and executes commands sent to it one at a time over the serial communications link at 9600 baud. Every time a robot movement is required, anywhere from ten to fifty bytes of information must be transmitted from the cell controller to the assembly robot. In the future it is recommended that a single, networked computer be attached to the microbot for handling this continuous information passing. This will result in improved cell controller performance.

6.57 Automated Storage and Retrieval System (ASRS) Supervisor

The purpose of this task is to monitor ASRS inventory levels, and to control the accurate part presentation mechanism. The ASRS consists of a rack with a series of gravity fed bins. This rack is indexed to appropriate positions so that the central robot may remove / add parts. Updating of the inventory database as parts are added or removed from storage is automatic. The positioning of a particular part is based on part number. This rack may be modified and expanded as necessary. No restriction has been placed on the number of storage bins. The rack indexing is provided by a stepper motor, controlled via a controller board mounted in the cell control computer.

6.58 Central Robot Supervisor

The purpose of this task is to monitor the status of the Central Material Handling Robot (CMHR) and to schedule appropriate associated activities. Because the central robot is a serving device, all the primary device groups (mill, lathe, and assembly) will request its services. This task has selectable priority levels for each of the tasks it is

requested to perform. Requests for service may occur near the same time. These requests are queued by the operating system, and are selected and executed by the central robot supervisor at the earliest appropriate time.

The CMHR is a 5 degree of freedom ASEA robot which had one particular restriction which directly influenced the interface developed for this device. The ASEA robot has a serial communications interface, but only propriety software can access it. Due to the proprietary nature, the information on the communication format was unavailable. As a result a direct communication link was not possible. One of the primary design considerations of the entire system was to allow each device to do as much as its own controller would handle and remove as much of the burden from the cell controller as possible. The central robot is programmed independently from the rest of the system using its own teach pendant.

A communication interface using digital I/O lines was developed to link the cell controller to the robot controller. By setting combinations of these lines messages are sent between controllers. The Central robot supervisor utilizes eight digital I/O output lines as a communication

link for passing messages from the cell controller to the central robot. Eight digital I/O input lines are similarly employed for message passing from the central robot to the cell controller. A SEND, RECEIVE, REPLY format was developed using these lines. Figure 6.5.1 displays a table of the suggested commands for the central robot, and Appendix C contains a structured program outline written in the format of the robot controller.

The general communication format between the CMHR supervisor and the ASEA robot is as follows:

NOTE: cell controller outputs 0 through 5 are robot controller inputs 1 through 6. Cell controller inputs 0 through 3 are robot controller outputs 1 through 4.

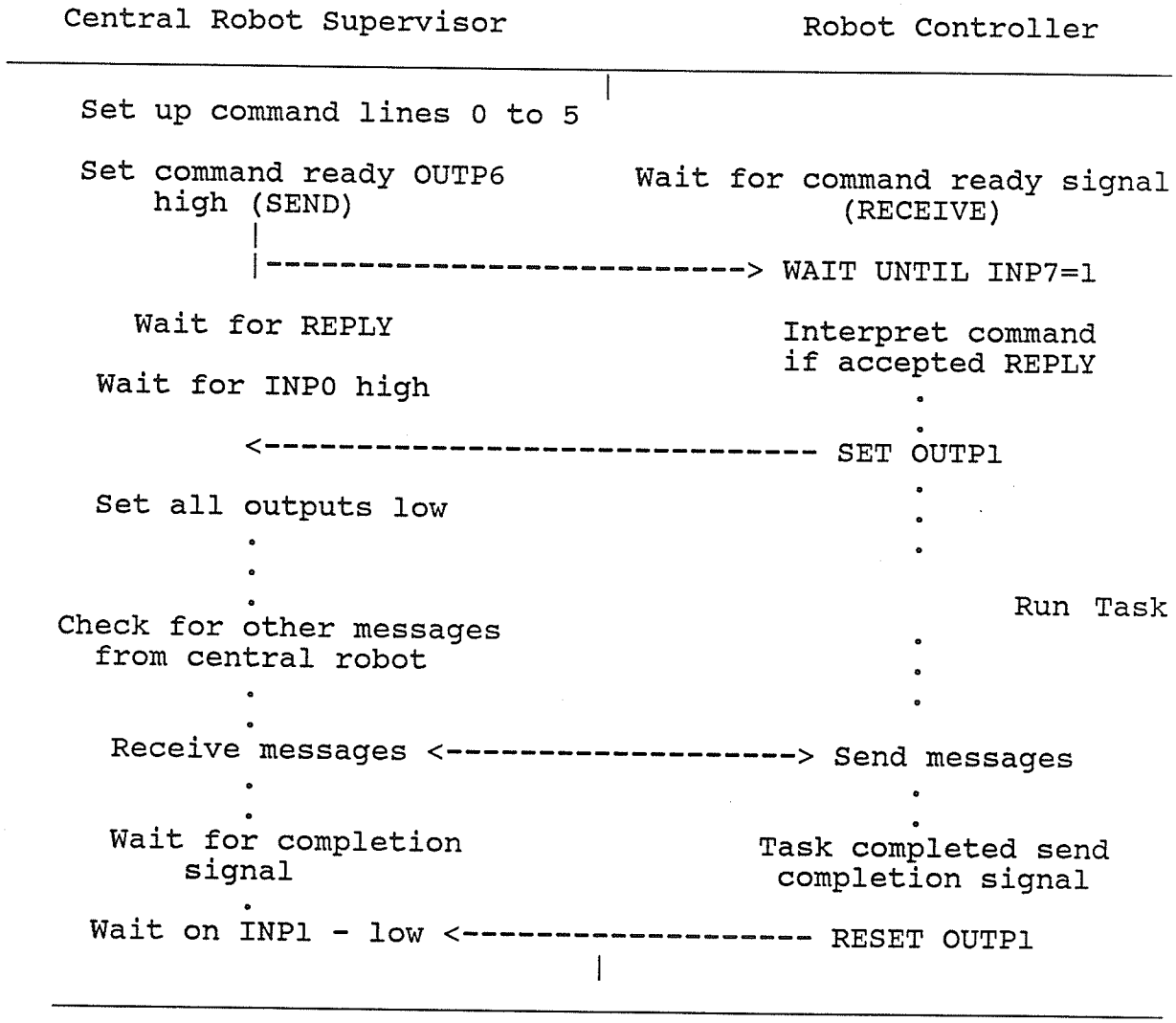


Figure 6.5.1 - ASEA / controller Communications Format

Intermediate messages do occur and a number of examples are presented in Appendix C. For each of the ASEA commands an execution priority status can be set. This priority status is used to determine which task should be executed if

more than one request for the robot is waiting.

For instance assume that the central robot was assembling components in the assembly cell, and both the mill and lathe finished machining their respective products during that time. Two requests for the central robots services would be waiting when the robot completed its assembly task. At that point, the command with the highest priority would be executed first. This feature allows important tasks to be given priority in a tie situation. These priority levels may be modified prior to any production run.

VII. EMERGENCY MONITORING SYSTEM

One component which is very important but not implemented in this system to this point is an extensive emergency monitoring system. The control system developed, was not designed to be a completely unattended system at this stage. Tool breakage and fixture breakdowns etc. are not currently monitored. In the future this area must be addressed.

Currently it is the operators responsiblity to watch for calamities and interrupt the sytem if something fails. Typing control C on the controller keyboard will interrupt the system and hold all running tasks. The production run can be simply aborted at this time without saving the current production status, or the production run may be restarted from near the same production point. If the system is interrupted and restarted, the parts in the machines at the time of the shut down are considered failed and must be removed from the system.

An intricate emergency monitoring system was not implemented at this time due to time constraints. In the following paragraphs an outline of a possible emergency

monitoring system is presented.

The production system currently consists of a central controller (MC) with seven device group supervising tasks. An emergency system would almost mirror this design (Figure 7.1). Each group of devices would have an emergency monitoring task associated with it. This task would monitor the condition of the device group and inform the main emergency controller task of any inconsistencies. In the case of an emergency, the monitoring task which identified a problem would interrupt the main emergency controller with a message. The controller would interpret the message and interrupt the manufacturing controller (MC) and dictate the appropriate action to be taken, such as a power shut down.

The emergency monitoring system could be isolated on a different node of the computer control network. The system could have a separate power supply and provide greater reliability and autonomy to the actual control system. Monitoring of the computer control network could also be performed using this method.

Manufacturing Controller

Emergency Monitoring System EMS (Proposed)

Task Communication Flow Diagram

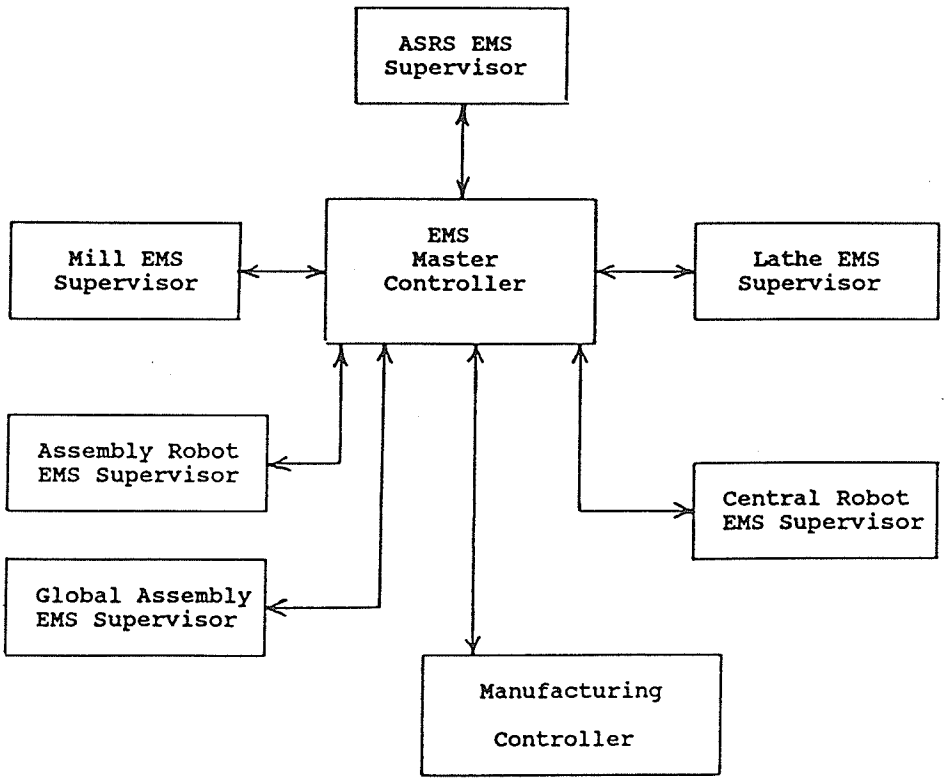


Figure 7.1 - Emergency Monitoring System

VIII. CONCLUSIONS

Personal computers based on the IBM-PC bus are becoming very popular as controllers for manufacturing processes. Industrial versions of the PC are being placed on the factory floor as local system monitoring equipment. The cost and performance of the personal computer make it an ideal component in a computer integrated maufacturing enviroment.

The Manufacturing Control Environment (MCE) presented in this thesis demonstrated the ability to utilize industrial personal computers for real-time process control. The prototype system presented was able to integrate existing stand-alone machinery, such as CNC mills and lathes, into an integrated cohesive manufacturing system.

The Manufacturing Control Environment (MCE) demonstrated a feasible and flexible approach to linking the various components of a manufacturing system from design though part production. The utilization of readily available PC-based CAD applications in the control system, presented a cost effective solution for smaller industries interested in entering the CAD/CAM arena.

The various software components of the MCE presented a unique approach to presenting technical manufacturing information. The Production Simulator (PS) provides accurate estimates of production times and operation sequencing, which is necessary for capacity planning and scheduling. As well, the PS can be utilized as a base for research in developing expert systems for optimizing production sequencing.

The Manufacturing Controller (MC) production outline system was developed as a simple programming interface for describing production plans, without the necessity of requiring extensive knowledge of machine loading and timing. The MC display system is a unique simple format for presenting activities occuring within the CIM cell at any given time. It is an effective method of presenting technical control information in an "obvious" task / control relationship format.

The central robot control scheme utilized a simple input/output interface with combinations of digital input and output lines. Although this may not be optimal, it does demonstrate one method to link devices with the simplest of

communication interfaces. A more appropriate interface would be a serial RS232C communication interface between the robot and the cell controller. It should be noted that many companies provide serial communication interfaces with their robots or CNC machines, but very few will provide information on their communication protocol. This results in an inflexible system, that is very difficult to integrate. It is recommended, that with the purchase of any computer controlled equipment that all the components of the commuications interface namely the software and hardware be included at the time of purchase. In the case of the ASEA robot described in this project, the ability to utilize the serial communications interface included by ASEA, would have simplified the project greatly.

The task of making software or hardware additions to process control systems is greatly simplified when the system is based on networked industrial personal computers. More computer processing power can be added at any time. The ability to reallocate computer resources to different areas of the control system make software and hardware modifications easier. The implementation of the Emergency Monitoring System is one example of an addition which may be made at any time without causing a major impact on the

existing manufacturing system.

The approach taken in this project to integrating existing manufacturing equipment by using a flexible computer control base is one cost effective alternative for plant automation.

IX. RECOMMENDATIONS

The MCE provides a versatile base for research into other areas of CIM such as vision applications, emergency monitoring, tool breakage monitoring etc. Outlined below are a few recommendations for follow-up work.

1. An attempt should be made in developing an emergency monitoring system for the given prototype MCE. Such a system would allow complete unattended machining and would provide greater level of safety during manufacturing.

2. Further production testing should be performed on the system to insure that all features in place are tested for reliability. Also more complicated production runs must be evaluated.

3. A small network should be implemented to demonstrate the overall concept of "networked" industrial PC control. A primary system could consist of a CAD workstation networked to the cell controller and an application system. An application system would provide a base for the development of an EMS and for testing other concepts.

4. Further work in the development of jigs and fixtures for assembly applications should be encouraged. More flexible jigs and fixtures would provide greater assembly accuracy and repeatability.

5. Work in the area of appropriate modelling of the entire system should be made. Software then may be developed to optimize production sequencing.

X. References

1. Hemmond, R.F., "Making CIM work on the shop floor", Production Engineering, Sept. 1986, pp.44-47.
2. Hughes, J., "PCs and distributed control: Partners in productivity", I&CS, Sept. 1985, pp.45-47.
3. Merritt, R., "Personal computers gain ground on factory/plant floor", I&CS, Sept. 1985, pp.73-79.
4. MacAloney, B.G., "Just what is a cell controller?", I&CS, June 1987, pp.47-49.
5. "An FMS without solitary traffic cop", AMERICAN MACHINIST & Automated Manufacturing, July 1986, pp.79-83.
6. Rubin, S.E., "PCs in control: Are our expectations too great?", I&CS, May 1987, pp.33-38.
7. Janasy, L.C., "PCs and the Factory: Symbiosis in action", Production Engineering, April 1985, pp.44-54.
8. "MORE POWER to the FLOOR", Production Engineering, July 1986, pp.30-32.
9. Costa, A., "Design of a control system for an FMS", Journal of Man. Systems, Vol. 4 No. 1. pp.65-84.
10. Smith, R.S. and Meyerhofer, W.L., "Toughening up PCs for industrial use", I&CS, Sept. 1985, pp.91-92.
11. Pinto, J., "A Real-time, multitasking operating system for the IBM-PC", I&CS, Sept. 1985, pp.102-103.
12. Wiatrowski, C.A., "Using PC-Dos for real-time control", I&CS, Aug. 1986, pp.51-54.
13. Cleaveland, P., "Personal computers: A technology update", I&CS, Oct. 1986, pp.53-62.
14. Elfring, G., "A Message-Passing Executive", PC TECH Journal, Jan. 1987, Vol. 5 No. 1.
15. Readman, G., "Is it a controller or an industrial computer?", C&I, Jan. 1987, pp.39-40.
16. Tinham, B., "The changing face of programmable controllers", C&I, Jan. 1987, pp.29-33.
17. Schoenberg, S., "Industrial I/O enhances PC power", I&CS, Sept. 1985, pp.81.
18. Finucane, J.C., "PCs gear up for manufacturing", PC World, Dec. 1986, pp.196-211.
19. Greenberg, K., "The self-made PC", PC World, Dec. 1986, pp.202-211.
20. Cleaveland, P., "New industrial computers are XT, AT-compatible", I&CS, April 1987, pp.57-59.
21. Sole, C.J., "New software concepts for distributed control", I&CS, Sept. 1987, pp.50-54.

22. Cox, W.C., "A case for NPOSs in real-time applications", I&CS, Oct. 1987, pp.43-45.
23. Cutkosky, M.R., et. al, "The Design of a Flexible Machining Cell for Small Batch Production", Journal of Man. Systems, Vol. 3 No. 1, pp.39-59.
24. Babb, M., "Riding the PC Bus: New Avenues to Industrial Automation", Control Engineering, Oct. 1986, pp. 64-65.
25. Laduzinsky, A.J., "Software Expands Industrial Control", Control Engineering, July 1986, pp.42-43.
26. Jones, L., "IBM PC Bus - the industrial control connection", I&CS, Sept. 1986, pp.33-35.

XI. APPENDIX A:

Production Outline Sample

```
|-------------------------------------------------------|
|   The definition section of a production program      |
|   is used for outlining general production            |
|   information. This includes what areas of the cell   |
|   are being used (defined using the production type)  |
|   the part program files used by the mill and lathe   |
|   (if applicable), and the robot command files        |
|   which are required by the assembly robot.           |
|-------------------------------------------------------|
```

```
define
```

```
|-------------------------------------------------------|
| The production type refers to the equipment being     |
| utilized in the cell for this production run.         |
|                                                       |
|       1 - assembly cell                               |
|       2 - lathe                                       |
|       3 - lathe and assembly cell                     |
|       4 - mill                                        |
|       5 - mill and assembly cell                      |
|       6 - mill and lathe                              |
|       7 - mill, lathe and assembly cell               |
|-------------------------------------------------------|
```

```
   begin
   prodtyp: 7
   mill
      nprogs: 1
      file: mill.cmc
      port: $mdm
   lathe
      nprogs: 1
      file: lathe.clc
      port: $term1
   assemble
```

```
|-------------------------------------------------------|
| The robots command defines which robots are being     |
| utilized under this production program, for the       |
| purpose of assembling products                        |
|                                                       |
|       1 - asea (central robot)                        |
|       2 - microbot robot (assembly robot)             |
|       3 - both the central and assembly robots        |
|                                                       |
|-------------------------------------------------------|
```

```
      robots: 1
```

```
        end
mill
   begin
      loop: 12
      nparts: 1
      prod
         part#: 0001
         sgldbl: 1
         load: 2
         prog#: 1
         unload: 14
      endloop
   end
lathe
   begin
      loop: 12
      nparts: 1
      prod
         part#: 0002
         load: 4
         prog#: 1
         unload: 28
      endloop
   end
assemble
   begin
      loop: 12
      ntasks: 1
      task
         nparts: 2
         robot: 1
         frprod: 0001
         frprod: 0002
      endloop
   end
```

XII. APPENDIX B:

Production Simulator Data Generator Output and Description

Data Generator Output

```
0001 1 00030 3 0030 5 0300 2 0000 3 0030 2 0160 8 0045 9 0000 0 0000 0 0000
0002 1 00060 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0001 1 00390 3 0030 5 0300 2 0000 3 0030 2 0260 8 0045 9 0000 0 0000 0 0000
0002 1 00520 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0001 1 00750 3 0030 5 0300 2 0000 3 0030 2 0420 8 0045 9 0000 0 0000 0 0000
0002 1 00980 3 0030 7 0400 2 0000 3 0030 2 0090 8 0045 9 0000 0 0000 0 0000
0001 1 01110 3 0030 5 0300 2 0000 3 0030 2 0490 8 0045 9 0000 0 0000 0 0000
0002 1 01470 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0001 1 01500 3 0030 5 0300 2 0000 3 0030 2 0560 8 0045 9 0000 0 0000 0 0000
0001 1 01860 3 0030 5 0300 2 0000 3 0030 2 0660 8 0045 9 0000 0 0000 0 0000
0002 1 01930 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0001 1 02220 3 0030 5 0300 2 0000 3 0030 2 0820 8 0045 9 0000 0 0000 0 0000
0002 1 02390 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0001 1 02580 3 0030 5 0300 2 0015 3 0030 2 0875 8 0045 9 0000 0 0000 0 0000
0002 1 02850 3 0030 7 0400 2 0000 3 0030 2 0090 8 0045 9 0000 0 0000 0 0000
0001 1 02955 3 0030 5 0300 2 0025 3 0030 2 0950 8 0045 9 0000 0 0000 0 0000
0002 1 03340 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0001 1 03370 3 0030 5 0300 2 0000 3 0030 2 1020 8 0045 9 0000 0 0000 0 0000
0001 1 03730 3 0030 5 0300 2 0000 3 0030 2 1120 8 0045 9 0000 0 0000 0 0000
0002 1 03800 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0001 1 04090 3 0030 5 0300 2 0000 3 0030 2 1190 8 0045 9 0000 0 0000 0 0000
0002 1 04260 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0002 1 04720 3 0030 7 0400 2 0000 3 0030 2 0030 8 0045 9 0000 0 0000 0 0000
0002 1 05180 3 0030 7 0400 2 0000 3 0030 2 0000 8 0045 9 0000 0 0000 0 0000
```

PS Data Generator Format Description (Summarized)

```
1111  2  03333  4  5555 .......
 (a) (b)  (c)  (d) (e)
```

part number
(b) operation id
(c) duration of operation in seconds
(d) and (e) - subsequent operation / time pairs.

Below are the operation id numbers and their description:

Operation id#	Description
1	inventory
2	idle
3	handling
5	milling
7	turning
8	assembly
9	completion, removed from system

Below is a sample "time-line" and description:

```
0001 1 00050 3 0030 5 0400 2 0045 3 0035 2 0015
                                    8 0070 9 0000 0 0000
```

0001 1 00050 - part number 0001 being removed from storage 50 seconds into the production run.
3 0003 - The part is loaded into the mill, taking 30 seconds.
5 0400 - Part 0001 is milled for 400 seconds.
2 0045 - The part sits idle in the mill for 45 seconds until being removed by the CR.
3 0035 - Part is transported to the assembly cell, taking 35 seconds.
2 0015 - Part sits idle in the assembly cell for 15 seconds prior to being assembled.
8 0070 9 0000 - Part is being assembled for 70 seconds and then leaves the system.

13. APPENDIX C:

Material Handling Robot - Programming Outline

Asea Commands - Defined in file asea_lines

The asea robot allows sequential checking of digital input lines through its programming language. The asea robot communicates with the cell controller and asrs via seven digital input lines, and four digital ouput lines.

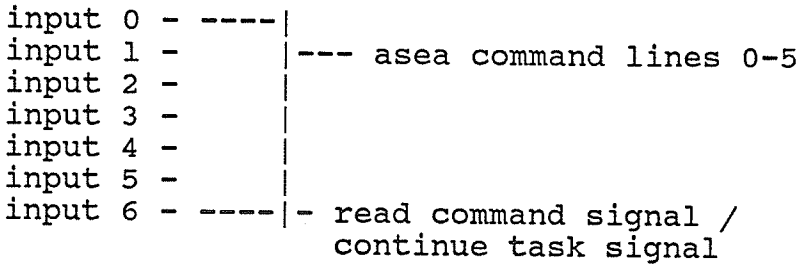
A line by line description is given below.

File: asea_lines

- 4 - output lines
- 7 - input lines

- The asea_lines file is read by the asea task

output 0 - command acknowledge, and task completion line
output 1 - index / return storage rack
output 2 - wait for continuation acknowledgement
output 3 - command ready signal



The asea task requires command numbers which are the binary equivalent of the six command lines 0-5. Listed below are the suggested commands and optional commands.

lines 0 1 2 3 4 5	Command description	Command #
1 0 0 0 0 0	Assembly command #1	1
1 1 0 0 0 0	Assembly command #2	3
1 1 1 0 0 0	Assembly command #3	7
1 1 1 1 0 0	Assembly command #4	15
1 1 1 1 1 0	Assembly command #5	31
1 1 1 1 1 1	Assembly command #6	63
0 1 0 0 0 0	load mill from storage	2
0 1 1 0 0 0	unload mill to storage	6
0 1 1 1 0 0	unload mill to assembly	14
0 1 1 1 1 0	extra command	30
0 1 1 1 1 1	extra command	62
0 0 1 0 0 0	load lathe from storage	4
0 0 1 1 0 0	unload lathe to storage	12
0 0 1 1 1 0	unload lathe to assembly	28
0 0 1 1 1 1	extra command	60

The suggested programming structure for the asea robot is given below.

Asea Robot General Control Program Outline

```
------------------------------------------------------------
|                                                          |
|    ASEA Main Line Control Program                        |
|                                                          |
------------------------------------------------------------

xxxx  RECT COORD                          coordinate system
xxxx  V = 500 mm/s  MAX = 1000 mm/s       velocity max
xxxx  POSITION                            set home position
0040  POSITION                            set home position
xxxx  WAIT UNTIL INP7=1                   command ready line
xxxx  JUMP TO 0200 IF INP1 = 1            assembly command
xxxx  JUMP TO 1000 IF INP2 = 1            mill command
xxxx  JUMP TO 2000 IF INP3 = 1            lathe command
xxxx  JUMP TO 40                          command unknown

------------------------------------------------------------
|                                                          |
|    Check for appropriate assembly command                |
|                                                          |
------------------------------------------------------------

0200  JUMP TO 0300 IF INP2 = 1    check next assembly
                                  command
xxxx  CALL PROG1                  assembly program - cmd 01
xxxx  JUMP TO 40                  return to home position
0300  JUMP TO 0400 IF INP3 = 1    check next assembly command


xxxx  CALL PROG2                  assembly program - cmd 03
xxxx  JUMP TO 40                  return to home position
```

```
0400  JUMP TO 0500 IF INP4 = 1  check next assembly command
xxxx  CALL PROG3                assembly program - cmd 07
xxxx  JUMP TO 40                return to home position
0500  JUMP TO 0600 IF INP5 = 1  check next assembly command
xxxx  CALL PROG4                assembly program - cmd 15
xxxx  JUMP TO 40                return to home position
0600  JUMP TO 0700 IF INP6 = 1  check next assembly command
xxxx  CALL PROG5                assembly program - cmd 31
xxxx  JUMP TO 40                return to home position
0700  CALL PROG6                assembly program - cmd 63
xxxx  JUMP TO 40                return to home position
```

Check for appropriate mill command

```
1000  JUMP TO 1100 IF INP3 = 1  check next mill command
xxxx  CALL PROG7                mill program - cmd 02
xxxx  JUMP TO 40                return to home position
1100  JUMP TO 1200 IF INP4 = 1  check next mill command
xxxx  CALL PROG8                mill program - cmd 06
xxxx  JUMP TO 40                return to home position
xxxx  JUMP TO 1300 IF INP5 = 1  check next mill command
xxxx  CALL PROG9                mill program - cmd 14
xxxx  JUMP TO 40                return to home position
1300  JUMP TO 1400 IF INP6 = 1  check next mill command
```

```
xxxx  CALL PROG10                    mill program - cmd 30
xxxx  JUMP TO 40                     return to home position
1400  CALL PROG11                    mill program - cmd 62
xxxx  JUMP TO 40                     return to home position
```

Check for appropriate lathe command

```
2000  JUMP TO 2100 IF INP4 = 1       check next lathe command
xxxx  CALL PROG12                    lathe program - cmd 04
xxxx  JUMP TO 40                     return to home position
2100  JUMP TO 2200 IF INP5 = 1       check next lathe command
xxxx  CALL PROG13                    lathe program - cmd 12
xxxx  JUMP TO 40                     return to home position
2200  JUMP TO 2300 IF INP6 = 1       check next lathe command
xxxx  CALL PROG14                    lathe program - cmd 28
xxxx  JUMP TO 40                     return to home position
2300  CALL PROG15                    lathe program - cmd 60
xxxx  JUMP TO 40                     return to home position
xxxx  PROGRAM END
```

Assembly programs

ASSEMBLY PPROGRAM #1

Below is an example of an assembly task with no parts coming from storage and no interaction with the assembly robot.

```
xxxx SET OUTP 1                    signal message reception
xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement
   .
   .
   .  Execute the task
   .
   .
xxxx RESET OUTP 1                  signal end of task
xxxx JUMP TO 10                    wait for new command
xxxx RETURN                        return to mainline
```

ASSEMBLY PROGRAM #2

An assembly task which requires a part from storage

```
xxxx SET OUTP 1                    signal message reception

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .  begin task execution
   .

xxxx SET OUTP 1                    signal assembly supervisor
                                   for requirement of asrs

xxxx SET OUTP 3                    send command

xxxx WAIT UNTIL INP7 = 1           wait for indication that
                                  the asrs has part positioned
   .
   .  remove part from asrs
   .

xxxx RESET OUTP 3                  signal end of use of asrs

   .
   .  continue assembly task
   .

xxxx WAIT UNTIL INP7 = 0           check to make sure asrs has
                                   returned

   .
   .
   .  continue assembly task
   .
   .

xxxx RESET OUTP 1                  signal end of task

xxxx JUMP TO 10                    wait for new command

xxxx RETURN                        return to mainline
```

```
-------------------------------------------------------------
|                                                           |
|   ASSEMBLY PROGRAM #3                                     |
|                                                           |
|   A joint assembly task with the assembly robot           |
|                                                           |
-------------------------------------------------------------

xxxx SET OUTP 1                    signal message reception

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .
   .  begin task execution
   .
   .

xxxx SET OUTP 3                    signal assembly supervisor
                                   for req. of assembly robot

xxxx SET OUTP 4                    send command

xxxx WAIT UNTIL INP7 = 1           wait for indication that
                                 the message has been received

xxxx RESET OUTP 4                  acknowledge

   .
   .  continue with task if not in the same working area
   .

xxxx WAIT UNTIL INP7 = 0           wait for indication that
                                   the asea can continue

xxxx RESET OUTP 3                  acknowledge

   .
   .
   .  continue task
   .
   .

xxxx RESET OUTP 1                  signal end of task

xxxx RETURN                        return to mainline
```

```
-------------------------------------------------------
|                                                     |
|    Mill associated commands                         |
|                                                     |
-------------------------------------------------------

-------------------------------------------------------
|                                                     |
|    MILL PROGRAM #1                                  |
|                                                     |
|    load mill from storage                           |
|                                                     |
-------------------------------------------------------

xxx SET OUTP 1                    signal message reception

xxxx WAIT UNTIL INP7 = 0          wait for acknowledgement

   .
   .
   .  prepare to retrieve part from storage
   .
   .

xxxx SET OUTP 1                   signal to set up rack

xxxx WAIT UNTIL INP7 = 1          wait for acknowledgement

   .
   .
   .  retrieve part
   .
   .

xxxx RESET OUTP 1                 return rack signal

xxxx WAIT UNTIL INP7 = 0          wait for acknowledgement

   .
   .  load milling machine
   .
   .

xxxx RESET OUTP 1                 signal end of task

xxxx RETURN                       return to mainline
```

```
-------------------------------------------------------------
|                                                           |
|    MILL PROGRAM #2                                        |
|                                                           |
|    unload mill to storage                                 |
|                                                           |
-------------------------------------------------------------

xxxx SET OUTP 1                    signal message reception

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .
   .  prepare to remove part from mill
   .
   .

xxxx SET OUTP 1                    signal to set up rack

xxxx WAIT UNTIL INP7 = 1           wait for acknowledgement

   .
   .
   .  drop part in storage rack
   .
   .

xxxx RESET OUTP 1                  return rack signal

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .  return robot to home position
   .
   .

xxxx RESET OUTP 1                  signal end of task

xxxx RETURN                        return to mainline
```

```
MILL PROGRAM #3

unload mill to assembly
```

```
xxxx SET OUTP 1                    signal message reception

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .
   .  remove part from mill, place in assembly cell
   .  and return to home position
   .

xxxx RESET OUTP 1                  signal end of task

xxxx GOTO 10                       wait for new command

xxxx RETURN                        return to mainline
```

```
-------------------------------------------------------------
|                                                           |
|     Lathe associated commands                             |
|                                                           |
-------------------------------------------------------------

-------------------------------------------------------------
|                                                           |
|     LATHE PROGRAM #1                                      |
|                                                           |
|     load lathe from storage                               |
|                                                           |
-------------------------------------------------------------

xxxx SET OUTP 1                    signal message reception

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .
   .  prepare to retrieve part from storage
   .
   .

xxxx SET OUTP 2                    signal to set up rack

xxxx WAIT UNTIL INP7 = 1           wait for acknowledgement

   .
   .
   .  retrieve part and insert it in lathe chuck
   .
   .

xxxx SET OUTP 3                    signal ready clamp part

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

xxxx WAIT UNTIL INP7 = 1           wait for release signal

   .
   . release part and back away
   .
   .

xxxx RESET OUTP 3                  robot out of range message
```

```
xxxx WAIT UNTIL INP7 = 0          wait for acknowledgement
xxxx RESET OUTP 2                 return rack signal
xxxx WAIT UNTIL INP7 = 1          wait for acknowledgement
   .
   .  return to home position
   .
   .
xxxx RESET OUTP 1                 signal end of task
xxxx RETURN                       return to mainline
```

```
LATHE PROGRAM #2

unload lathe to storage

xxxx SET OUTP 1                    signal message reception
xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

  .
  .
  .  prepare to retrieve part lathe, and prepare rack
  .
  .

xxxx SET OUTP 2                    signal to set up rack
xxxx WAIT UNTIL INP7 = 1           wait for acknowledgement

  .
  .
  .  grasp part in lathe chuck
  .
  .

xxxx SET OUTP 3                    signal to release chuck
xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement
xxxx WAIT UNTIL INP7 = 1         wait for chuck release signal

  .
  . remove part part and back away
  .
  .

xxxx RESET OUTP 3                  robot out of range message
xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

  .
  .
  . place part in storage rack
  .
```

```
   .
xxxx RESET OUTP 2                  return rack signal
xxxx WAIT UNTIL INP7 = 1           wait for acknowledgement
   .
   .  return to home position
   .
   .
xxxx RESET OUTP 1                  signal end of task
xxxx RETURN                        return to mainline
```

```
-------------------------------------------------------------
|                                                           |
|    LATHE PROGRAM #3                                       |
|                                                           |
|    unload lathe to assembly                               |
|                                                           |
-------------------------------------------------------------

xxxx SET OUTP 1                    signal message reception

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .
   .  prepare to retrieve part from lathe,
   .  grasp part in chuck
   .

xxxx SET OUTPUT 3                  signal to release chuck

xxxx WAIT UNTIL INP7 = 1           wait for acknowledgement

   .
   . remove part part and back away
   .
   .

xxxx RESET OUTP 3                  robot out of range message

xxxx WAIT UNTIL INP7 = 0           wait for acknowledgement

   .
   .
   . place part in assembly cell,
   . and return to home position
   .

xxxx RESET OUTP 1                  signal end of task

xxxx RETURN                        return to mainline
```