NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Previously copyrighted materials (journal articles, published tests, etc.) are not filmed.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

Les documents qui font déjà l'objet d'un droit d'auteur (articles de revue, tests publiés, etc.) ne sont pas microfilmés.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30.

An Approach to Flexible Manufacturing System Design Based on Process Activity Diagrams

by

Philippe Rémi Piché

A Thesis
Presented to the University of Ottawa
In Partial Fulfillment of the
Requirements for the Degree of
Master of Applied Science
in
Electrical Engineering

Ottawa, Ontario, May 1987

ISBN 0-315-40733-6

UNIVERSITÉ D'OTTAWA
UNIVERSITY OF OTTAWA

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

ABSTRACT

Advances in computing technology, automated and programmable manufacturing elements such as computer-numerical machines, advanced sensor systems, and robot manipulators, have made Flexible Manufacturing Systems (FMS) a reality. This thesis presents a methodology for integrating manufacturing elements into a coordinated Flexible Manufacturing System. Manufacturing elements are first integrated into modular and programmable Flexible Manufacturing Cells (FMC) that execute manufacturing activities. Based upon the mechanical equivalence of a FMC to a computer, the core building blocks of FMS's, a FMS can be considered as a multiprocessor system. Process Activity Diagrams (PAD), a tool developed for the design of multiprocessor systems, are applied to the specification of the manufacturing process in terms of the set of required activities executed by FMC's, and are used to organize and coordinate these activities in the corresponding FMS. In this way the FMC's are integrated into a FMS.

Acknowledgements

'It was the best of times, it was the worst of times...' It took a long time, but it was time well spent! Time spent in my 'dungeon' hacking away at the ideas. Now that it is completed, and I can reflect back over the years, I can now say that 'It is a far, far better thing I do, than I have ever done...'

With this in mind, I would like to express may sincere thanks to Moshe Krieger, my mentor, for his patient advice and guidance during the course of my research. The 'sorcerer's apprentice' can now say he is ready to conduct this 'black magic' which is engineering. I would also like to thank Moshe for being such a good friend and I hope that we will remain friends.

J'aimerais aussi remercier mon épouse Renée Langlois, qui a souffert avec moi pendant ce précieux chef-d'oeuvre. Elle, qui m'a encouragé et soutenu, et, à sa façon unique, m'a porté conseil pendant ces dernières années. Maintenant, nous sommes tous deux UN!

De plus, j'aimerais remercier mes familles Piché et Langlois, qui m'ont aussi encouragé lors de mon travail de thèse. Parmi eux, je dois souligner mon père et ami Roland, qui m'a aidé avec la rédaction.

J'aimerais aussi souligner l'encouragement collectif de tous mes amis, entre autres Joseph Given et Ronald Pelot, qui ont déjà franchi cette étape.

Acknowledgements

En dernier lieu, j'aimerais remercier tous mes collegues à l'Université d'Ottawa, ainsi que ceux avec lesquels j'ai travaillé pendant ces dernières années.

Table of Contents

Table of Contents

Table of Contents

Table of Contents

Table of Contents

List of Figures

List of Figures

Chapter 1

Introduction

The purpose of this thesis is to present the applicability of Process Activity Diagrams (PAD) to the specification of the manufacturing process and the implementation of an integrated Flexible Manufacturing System. With the advances in computing technology, automated and programmable manufacturing elements — such as computer-numerical machines, advanced sensor systems, and robot manipulators, Flexible Manufacturing Systems (FMS) are becoming a reality.

Much of the past research effort in automated manufacturing has been related to the direct control of individual elements such as robot manipulators, sensor systems, end effectors, etc. Today, many of these elements are available from vendors in ready-to-use form and are supplied with their own programmable controllers. The major effort of users must now be directed toward the integration of the various elements into a coordinated Flexible Manufacturing System which can be reconfigured at acceptable costs.

To indicate the scope and complexity of modern automated manufacturing, Chapter 2 presents the history of automated manufacturing from its beginnings in the Industrial Revolution to the modern era of Computer Integrated Manufacturing. The important changes in manufacturing that led to the notion of Flexible

Manufacturing are highlighted.

In Chapter 3, a short description of Flexible Manufacturing Systems and its elements is presented. To show the ad-hoc nature of FMS's, different organizational approaches taken to control and coordinate its elements are also presented. The core building blocks of FMS's are shown to be Flexible Manufacturing Cells.

In Chapter 4, the functional elements of FMC's are presented and the view that a FMC is the mechanical equivalent to a computer is introduced. Based upon this equivalence, a general purpose coordinator architecture and corresponding cell-level programming language, used to implement the FMC's activity, are proposed. Together, they allow the integration of manufacturing elements from different sources into a programmable FMC.

In Chapter 5, Process Activity Diagrams (PAD), a tool used in the design of multiprocessor systems, are presented. Since a FMS can be considered as multiprocessor system consisting of FMC's, it is shown that PAD's can be used to specify the manufacturing process in terms of a coordinated set of manufacturing activities executed by FMC's. To demonstrate the applicability of the methodology, the manufacturing process of Printed Circuit Boards (PCB) is specified in terms of PAD's and approaches to the integration the FMC's into a FMS are discussed.

In Appendix A can be found a copy of three publications which touch on some of the points presented in the thesis and which indicate the evolution of the reported research. The first publication, 'Oligarchy: A Control Scheme For Flexible Manufacturing

Systems,' relates to the overall coordination aspects of FMS's. The second publication, 'Segmented Microprogramming: A Technique For Designing Special Purpose VLSI Processors,' explores some of the ideas that were applied to the design of various controllers. The last publication, 'PAD: A New Tool For FMS Design,' considers the major ideas concerning the design and programming of FMC's and integrating them into a coordinated FMS using PAD's.

Chapter 2

Automated Manufacturing

2.1 Introduction

In this chapter, the various aspects of automated manufacturing are presented beginning with a short account of its history. The continued and expanded use of computers in manufacturing is shown to result in Computer Integrated Manufacturing (CIM). The Flexible Manufacturing System (FMS) is shown to be the next step to fully automated manufacturing at the factory level.

2.2 The History of Automated Manufacturing

The introduction of automation in every day life begins in the period preceding industrial automation. Although automation techniques and schemes had been applied in various areas of human endeavor, from toy making, clock, watch and musical box making, and even life mimicking automatons (the Scribe and the Musician mechanized dolls from Pierre and Henri-Louis Jacquet-Droz in 1774 [ALBU81]), most of these applications were not directed towards the automation of industrial activities.

2.2.1 Historical Factors Leading to Automated Manufacturing

Before 1800, manufacturing of goods was largely done by hand, with tremendous dependence on craftsmen, their hand tools, and their skills. In the manufacturing of goods, although the product quality could be very high, the involvement of craftsmen meant a limited number of products, and rather high production costs.

After 1800, manufacturing of goods was increasingly done by machines. The production of goods is made more practical and attractive for two main factors: the harnessing of mechanical power and tool specialization and automation.

2.2.1.1 Harnessing of Mechanical Power

Industrial automation first took the form of the application of mechanical power to industrial activities. The application of power to industrial activity is the distinguishing factor marking the Industrial Revolution.

Power was first harnessed from water streams and the wind. Power obtained this way allowed for better soil irrigation in the agricultural industry, powered the saws of wood mills in the lumber industry, and drove the large looms in the textile industry [PALM65].

The next major development in the harnessing of power arose with the invention of the steam engine. It was first applied in the mining industry to pump out water and eventually to pull carts loaded with the extracted minerals out of the mines along track ways, or rail ways, by use of pulleys and takeup wheels [PALM65]. Soon after, with the invention of the governor, control over the speed of the engine was possible, and provided the industry with a constant, continuous and more reliable driving power. The steam engine could then be applied to the transportation sector and immensely improved communications, the distribution of people, products and materials, and market flow.

With the application of steam power to industrial machinery, the Industrial Revolution had firmly begun. Later, other forms of power were discovered and used in the industrial community, such as the power provided by the internal combustion gasoline engine and the power of electricity.

2.2.1.2 Tool Specialization and Automation

The process of shifting from hand tools to power machinery is what is meant by the Industrial Revolution. Therefore tools and automation are directly linked when considering tool impact on manufacturing. An increase in tool complexity, precision, and specialization directly results in an increase in manufacturing capability and automation.

Tools are capability enhancement devices permitting users to perform manufacturing operations more easily, with better precision, and more quickly while still providing effective control over the manufacturing steps. When the tools are properly selected, they combine their benefits to give an increased production rate as well as an increase in quality of the product. There are two major types of tools: general purpose tools, and specialized tools.

General purpose tools are usually simple in nature, require, for most applications, only a basic background on their use and a minimum amount of skill to use them in making simple products. To make more complex products, the user must possess more advanced skills.

Specialized tools on the other hand, are designed to satisfy needs in the manufacturing of products that cannot be met effectively with general tools. To use specialized tools, the user must be adequately trained or possess appropriate skills. The size of the production run and the complexity of the product to manufacture are the main factors affecting the choice of tool and the demand on user skills.

In terms of automation, one of the earliest and most widely acknowledged form of practical industrial automation in the true sense of the word, was the invention of an automated loom for the silk industry by the French weaver Joseph-Marie Charles Jacquard in 1801. Not only was it powered but it also operated independently of human labour and control, and most notably, was controlled in the same way numerically controlled machines are controlled today, by means of a "program" encoded in an "array of holes punched in a series of cards" [GUNN82] (see Figure 2-1).

In modern manufacturing, automated tools are used almost everywhere and most extensively in machining applications. With such tools, the production process depends less upon user skills and more upon the user's ability to control the tool in shaping the product. The user's tasks mainly have to do with feeding the tool and supervising the tool's progress.

2.2.2 Programmable Tools

The next step in the evolution of tools is programmability. There are in general two approaches to programmability: data driven and algorithm (computer) controlled.

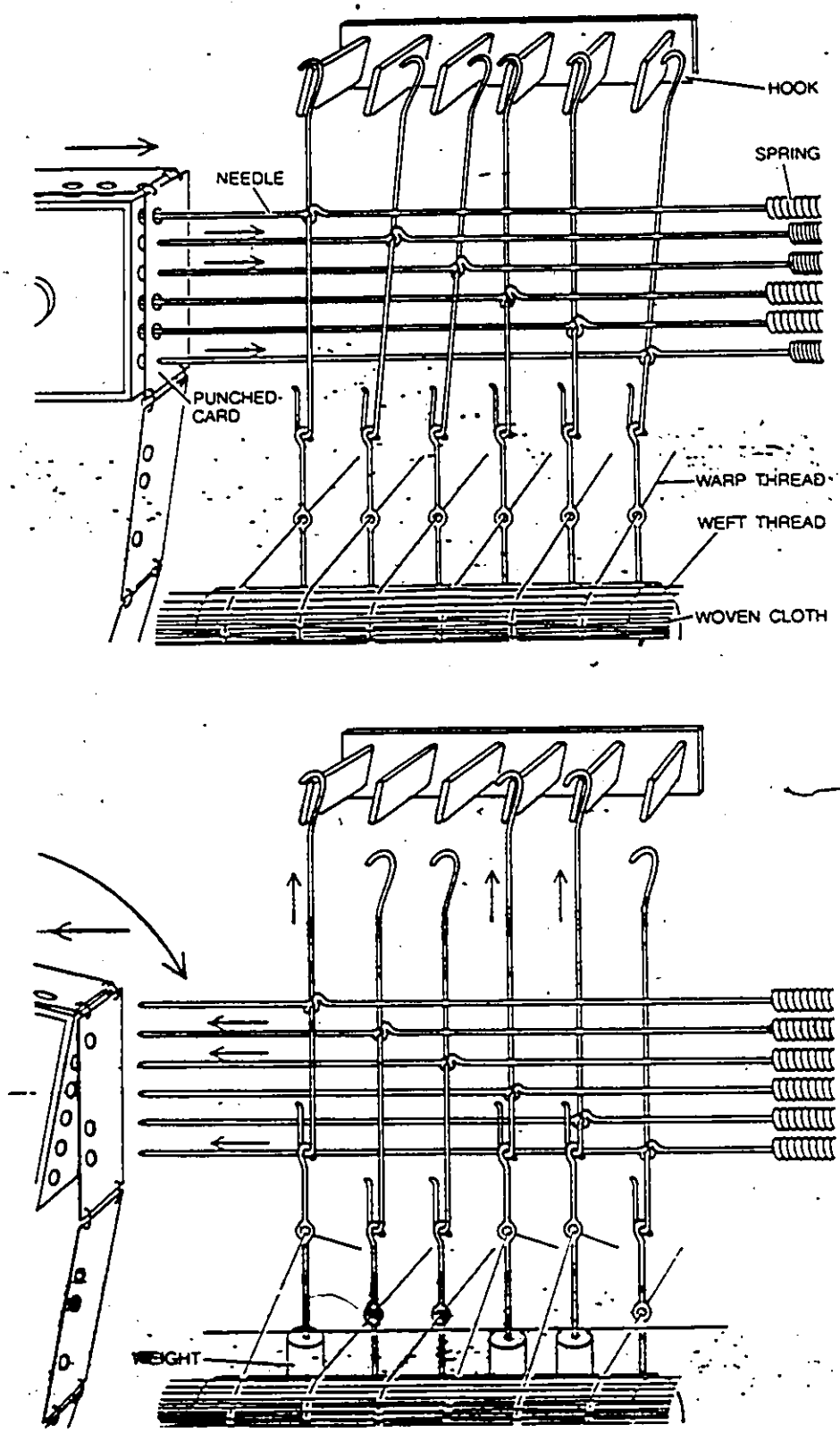


Figure 2-1. Jacquard's Loom.
(taken from [GUNN82])

2.2.2.1 Data Driven Tools

Data driven tools are largely machining devices that transform raw materials into usable parts which are either finished products, or are used later in the assembly of other products. Machining devices are largely programmable Numerically-Controlled (NC) machine tools, whose actions are governed by numerical data and various product specification parameters for each step in the production sequence. More simply, NC machines are a form of programmable tool in which the process is controlled by numbers, letters, and symbols [GR0080][GR0084]. The earliest independently running NC machine tools were controlled by means of a hard-wired control unit that read the part programming data directly from a punched tape.

Before computers were used to control NC machines, a manual approach was used. The manual NC method of control in manufacturing [GR0080][GR0084] consists of the following steps:

1: The engineering drawing of the workpart to be manufactured is interpreted in terms of machining instructions on a special form called a part programming manuscript; the part programming manuscript, unique to the tool being used, is a listing of the relative tool and workpiece locations, and includes other data such as preparatory commands, needed to operate the machine under tape control;

2: A punched tape is manually prepared directly from the part programming manuscript;

3: The tape is checked and verified;

4: The NC machine performs the job under tape control.

The manual method requires that the part programmer be knowledgeable about the machining process and have also been trained to program NC machines. The responsibilities to plan the

sequence of driving data used by the NC machine and to write these down in a special format belong to the part programmer himself. Once a job is initiated, there is less need for continuous supervision by the machinist. Improvement in quality control are made because production becomes more systematic and less dependent on worker skills.

Since detailed part information must be fed to the machine, a more complex setup is required for NC machines. This type of programmable tool is referred to as data driven automation [JURG83], because the incoming data represents the "control" of the machine. It is still the most widely used form of automation in industry today.

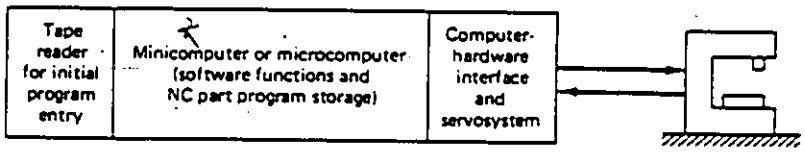
Computers were first used to do computer-assisted part programming where the part programmer uses an NC part programming language to describe a part. In this way, the the tedious computing work is transferred to the computer. The computer produces, from the set of geometrical descriptions, numerical data required by the NC machine. The data are transferred to punched tape used by the NC machine. In this arrangement, the computer is not directly in charge of the NC machine. Several examples of such languages are based on MIT's Automatically Programmed Tools (APT) such as: IBM's ADaptation of APT (ADAPT), EXtended subset of APT (EXAPT) from Germany, UNited Computing Corp.'s ATP (UNIAPT), Sundstrand Processing Language Internally Translated (SPLIT), Manufacturing Data Systems, Inc.'s COMPACT II, Weber N/C System, Inc.'s PROMPT, and Cincinnati Milacron's CINTURN.

The first attempt to use computers to directly control NC machines was in a large job shop where several NC machines were used to perform a number of machining steps. Several NC machine tools were directly linked to a single central computer. This is what is meant by Direct-Numerically Controlled (DNC) machine tools (see Figure 2-2). In DNC, a single large central computer controls many devices directly and concurrently.

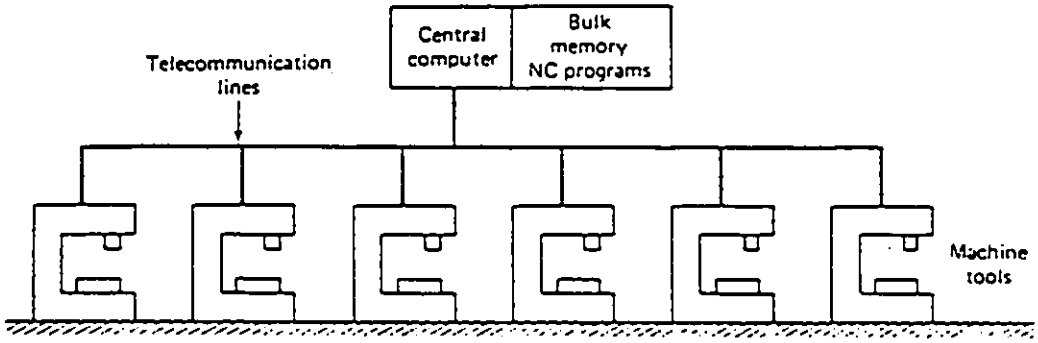
In a DNC system the only connection between the tools is electronic, still leaving the workpiece to be physically moved. By linking the DNC system with a transfer system, the central computer can be programmed to operate the tools in a specified sequence. In this type or arrangement, the main problem of implementation is to provide the proper coordination between the machining elements in order to achieve reliable, high quality, and cost effective production.

Computer-Numerically Controlled (CNC) machines are machining tools where a dedicated computer is directly in control of the device. For many applications, this method relieves the processing burden found in centrally controlled computer based DNC systems. Input data is processed locally by the dedicated computer resulting in controlled actions by the NC machine (see Figure 2-2).

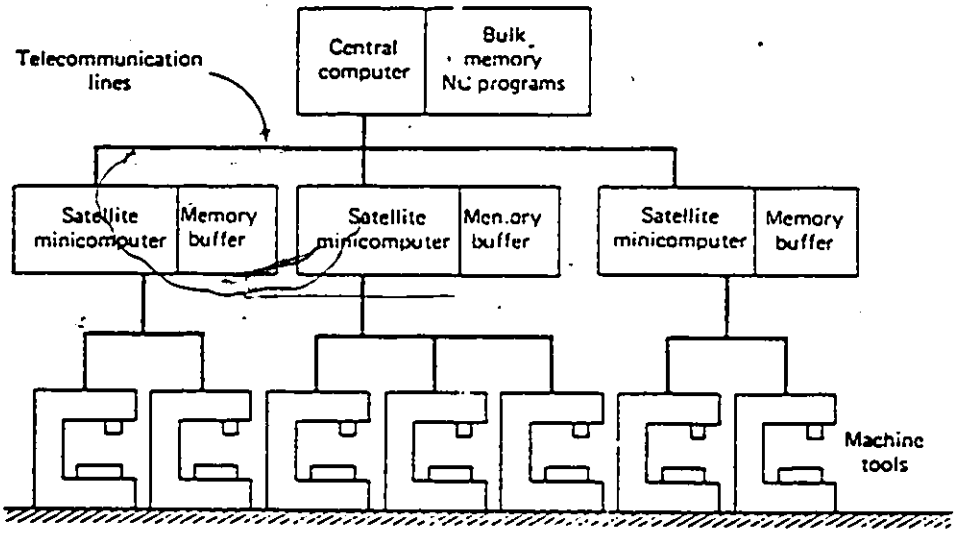
Today's CNC machines are controlled by microcomputers. In large job shops, several machines are often linked via a minicomputer, and several minicomputers are linked in turn via a large mainframe computer. Programs for the manufacture of parts



General configuration of computer numerical control (CNC) system.



General configuration of direct numerical control (DNC) system.



DNC with satellite minicomputers.

Figure 2-2. DNC and CNC Machine Configurations.
(Taken from [GRO084])

are stored in a central data base and are down-loaded to the specific machine tools in the network. Status information, production volume and quality control data are collected from the peripheral controllers (shop floor) and flow back up the hierarchy to the main computer and data base. Coordination and control of all the CNC machines in the plant still represents a major problem.

CNC machines can be controlled by using feedback elements to measure one or more process variables such as simple positional control. In this way a compensating action can be taken to counteract, for example, undesired changes in these variables. Thus the machining operation and precision could be optimized, and could react to the environment, aspects that simple NC could not accomplish.

CNC machines can be very large, and can incorporate a cutting head capable of independent motion around several axes at the same time. The dedicated computer control enables the machine to cut metal automatically to tolerances of a ten-thousandth of an inch using feedback sensors. The program is designed to prevent the machine from cutting too deeply into the workpiece and so ruining the part. In some cases, the computer signals the machine operator to change or sharpen the cutting tool when sensors indicate that the torque required to make the cut is outside the proper range of values.

2.2.2.2 Computer Algorithm Controlled Tools: Robot Manipulators

DNC and CNC arose from the marriage between the computer and

numerical control technology. Most applications of DNC and CNC are in the area of machining. Robot manipulators evolved from NC machine technology by following the path using DNC, CNC, and feedback control. Robotics is the application of computers in the area of product and tool manipulation, or object manipulation.

A robot manipulator is an enhanced CNC machine having material handling capabilities, i.e. the ability to execute object and tool manipulation operations (movement, trajectory, and tool operation). Even the most modern robot manipulators are in reality only automated servo manipulators. They are, in simple terms, nothing more than sophisticated pincers that replace muscle power, and allow actions to be repeated reliably in predetermined ways. By using sensory devices, feedback control systems are incorporated into robot manipulators. A robot manipulator can be used to do a variety of tasks from spot-welding and spray painting, to finished product assembly.

A robot manipulator is an algorithm controlled rather than strictly data driven programmable tool. Not only is the data used by the robot manipulator changeable, but the algorithm or control mechanism it uses to perform tasks using this data is also alterable (programmable). This reprogrammability characteristic is what makes it more flexible, at the cost of a more complex controller. To make effective use robot manipulators, it is necessary for them to be easily reprogrammable.

Because a robot manipulator is flexible in that it can be applied to a wide variety of manufacturing tasks, and because it

is also a programmable device, it will be the most prominent device used in the manufacturing of finished assembled products in the near future. Its scope of application and flexibility allows it to be used economically not only in large production runs, but also in smaller production runs under appropriate circumstances. In fact, in the future it may become the single most important factor to affect low-volume applications. It can perform mundane tasks such as loading material handling palettes and machine tools, or, because of its flexibility, it can be a better alternative to fixed tooling. This flexibility makes the robot manipulator the first true machine with the potential to perform generalized assembly tasks. Robot manipulators will be discussed more fully in this light in a later chapter to identify the kind of control and coordination needed to take advantage of the technology. One advantage of robots is that their performance never varies: robot manipulators may be slower than human workers but they never tire and are often more reliable.

Robot Manipulator Classification

To establish the present state of robotics in manufacturing, and give an indication of the complexity of the task at hand when using manipulators in manufacturing and assembly, the following generic classification scheme of robots is presented. Generic classification is mainly concerned with robot behavior and performance. Some characteristics upon which this classification is based are: load capacity, speed of movement, built-in sensors (such as actuator sensors), working conditions, manipulative

capability, ease of programming, complexity and sophistication of the controller, and reliability.

Requirements of present day robot manipulators vary immensely but in general they must have the following traits: they must be fully programmable, not simply pick-and-place; and they must be able to perform many complex series of motor actions over a variety of processes. The provision of sensory facilities and the type of programming used defines in part the robot's generation [SIMO8Ø].

First Generation Robots

First generation robots are of the fixed sequence type which can repeat a sequence of operations tirelessly and reliably, once having been "programmed" to do so. Programming of these machines is usually done by "training" or "teaching" (also referred to as guiding [LOZA83]), where an operator literally leads the device by the hand from one step to the next. This is done for each new job. The robot "remembers" the sequence of moves and repeats the task indefinitely, until a new task or new training is required. The sequence of tasks may be recorded and stored for future use.

The main advantages of this class of robot are:

1: guiding is a simple technique to implement and use that does not need a general purpose computer;

2: this type of robot is good for such tasks as spot-welding, spay painting, and simple materials handling (parts transfer).

The main drawbacks of this class of robot are:

1: objects being worked on must be in the right place at the right time;

2: no feedback is available; for example in a painting application, a robot will happily continue painting thin air if the object to paint is misplaced;

3: these robot are unable to pick up randomly positioned parts from a moving conveyor belt, or to pick-out unoriented parts from a bin;

4: these robot systems have a single execution sequence with no looping capability, no conditional operations, and no computing capability.

First generation robots are mostly simple pick-and-place devices. They have some measure of programmability, can be memory controlled, have several degrees of motion, and have several attachments to operate tools such as paint-spray guns, welding heads, etc. They are well suited to handle materials in various parts transfer situations. More advanced first generation robots are provided with simple sensory control, such as stop switches used in error detection, for enhanced capabilities.

The first generation of robots greatly resemble NC machines in the way they are "set up" (the recording and playback of stored sequences of numerical control codes that drive robot actuators). In advanced situations, where a minicomputer is used, the robot may be able to select from a variety of fixed stored sequences, depending on the state and character of components arriving at the workstation. In general, recognition can only be done if the robot can discern and interpret variations in the environment.

Second Generation Robots

The main characteristics that distinguish second generation robot manipulators from first generation robots are:

a) the provision of sensory facilities,
b) data processing facilities and,
c) the use of robot-level programming languages.

The need for the robot to react to environmental changes requires the use of sensors and greater processing power. The processing requirements are dictated by the type of sensors used, their number, the actual physical structure of the robot (dimensions, number of joints, etc), the manner in which the parts are presented, and the speed and complexity of the tasks to be done. Although robots may have had some sensory capability before, largely primitive ones, in general, second generation robots use more advanced sensors such as tactile, sonar, and vision.

The main advantages of second generation robots are:

1: ability to react to the environment through the application of sophisticated sensory systems; randomly positioned parts on a moving conveyor belt or in a bin can be identified and picked up;

2: programmable; a controller can implement complex algorithms used to perform advanced tasks such as assembly;

3: ability to analyze data through powerful local computational facilities;

4: failure handling can be done by detecting abnormal situations;

By processing higher level mathematical representations, object recognition can be performed. Different strategies such as "edge finding" and "region growing" are commonly used. Shape of

outline, region shading, and the objects' relationships to other regions in the scene are found to perform property determination. Recognition of an object can rely upon comparing the outline approximation and the calculated parameters with those stored in memory. When parts are recognized, their positions can be determined and any necessary positional actions can be undertaken.

Robot-level programming (also called explicit programming) uses computer languages with commands to access sensors and to specify robot motions. Data from external sensors, such as vision and force, are used to modify robot motions. Therefore with sensors and appropriate programming, robots can cope with greater degrees of uncertainty in the position of objects, and so increase their range of application. The main drawback of robot-level programming requires programmers to be experts in computer programming and in design of sensor based motion strategies (real-time programming). Therefore programming cannot be done by factory workers [LOZA83].

Second-generation robots can be instructed to follow a predetermined pattern, with the attainment of a specified target monitored through local sensory feedback. Coupled with greater accuracy of the mechanical structure of the robot, much higher repeatability can be achieved. The robots become "interactive", by fitting in the gripper a miniature camera used to gather scene data for recognizing an object's position. The gripper can then be moved to pick up objects from a rotating table. In this way robots can have "hand-eye coordination". Second generation robots

are therefore moderately adaptable to a changing environment. Other sensory systems of second generation robots incorporate sonar-based sensing such as in the "acoustic phase monitoring" technique used to find the position of arbitrarily shaped objects.

Some of the disadvantages of second generation robots are:

1: complex control algorithms may potentially have to be implemented for every new application;

2: robot controllers are difficult to integrate with equally complex sensory controllers unless they are "made for each other" by the same manufacturer;

3: expert programmers are required to program these robots increasing the set up costs;

4: robots and their support systems are expensive;

5: general purpose end effectors are difficult to design; often new grippers must be invented for new applications;

Next Generation Trends

With the advent of powerful micro-processing elements, localized processing will lead to more advanced capabilities for robot manipulators. This computing requirement is fundamental for processing in real-time the data from feedback sensors. This computing capability will allow a very high degree of interaction between robots and their environment that will be part of the next generation. Already some of the features are beginning to appear.

A robot of the next generation will use a number of processing elements to cope with the amount of data that will be required to be processed. Also, the complexity of the computations

demands that data be kept locally at the robot to keep communication requirements low and to provide for real-time execution of complex tasks. Tactile feedback used in performing exacting tasks such as fastening, insertion and other assembly tasks will play a major role in the next generation. Tactile feedback coupled with vision and sonar require that high speed data processing be made available in order to fully automate assembly. Recognition by visual means simply is not sufficient for the variety of tasks robots will be doing. Complementing vision with tactile and sound feedback will provide to the controller(s) with the information needed to compensate for weight and shape differences in objects. These objects may in turn require special manipulation for assembly.

A key problem to be resolved in robot control and application in assembly is that of part identification. By preserving the orientation of the parts from station to station with an adequate transfer system in a multi-station manufacturing system, robots can compete economically with other machines. With the aid of a sophisticated sensory system such as vision, it may be able to recognize and pick a randomly oriented assembly part out of a bin.

Other major developments are coming from the software aspect of control. New specialized languages are being developed to ease the programming task. The new-language approaches may not be the best long term solution because of the support they must have. Therefore research is also being conducted in adapting

modern high-level programming languages such as C, Pascal or Ada to the programming of robot systems. The programming environment provided with these high level languages as well as their support will be major considerations for their use, as for any software language. Task-level programming could also be developed such that programming can be made to be completely robot independent [LOZA83]. Furthermore, artificial intelligence techniques are being investigated for industrial robot control.

2.2.3 Transfer Systems

The word "automation" itself first appeared in the 1930's, and was used to refer to "the automatic transfer of parts and materials from one production operation to the next." Today, "automation" is not reserved strictly to automatic transfer. In this section, devices used in transferring parts are discussed.

2.2.3.1 Methods of Transfer

There are three major methods of transfer now found in manufacturing systems:

1: continuous transfer where objects move continuously at constant speed; it is the simplest and oldest form of transfer system;

2: intermittent (synchronous) transfer where objects move discontinuously (that is all the objects move at the same time) usually between work stations; this form of transfer first impacted the automobile industry;

3: non-synchronous transfer (power-and-free) is a modern method of transfer where each workpart moves to the next station independently when processing at the current station has been completed (when it is ready to move); plants that use non-synchronous transfer systems have the characteristic that some parts may be moving while others are being processed; this type of transfer offers greater flexibility than the

other types but usually operate at cycle rates slower than the other types of transfer.

Transfer systems have many forms depending upon the objects they transfer. In the chemical industry, they consist of piping, control valves and the like and are on the continuous type. In product manufacturing, most of the existing transfer systems are of the intermittent type. There are several kinds of intermittent transfer systems: linear transfer systems such as the walking beam transfer bar system, the powered roller conveyor system, and the chain-drive conveyor system; rotary transfer systems such as the rack and pinion mechanism, the ratchet and pawl mechanism used in indexing machines, the Geneva mechanism used to convert linear motion into a rotational motion, and cam mechanisms used to provide accurate dial indexing [GR0080]. Also, it should be noted that in many cases, robot manipulators are used in the the transfer system.

2.2.3.2 Transfer System Enhancements

In the field of manufacturing enhancements to the transfers systems, were made to ease the control and the flow of objects in the plant from one station to another. These enhancements involve the use of pallets, fixtures and buffer storage.

Some transfer systems accommodate pallet fixtures to which parts are attached and the pallets are transferred between work stations. Pallets are designed to be conveniently moved, located, and clamped into position at successive work stations. When the pallet is correctly positioned, the parts are accurately located

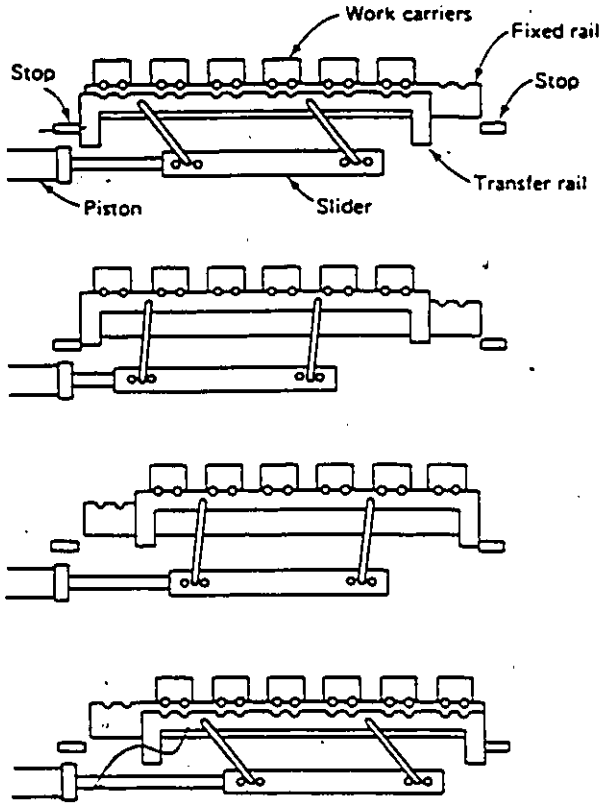


Figure 2-3. Walking Beam Transfer System.
(Taken from [GROO8Ø])

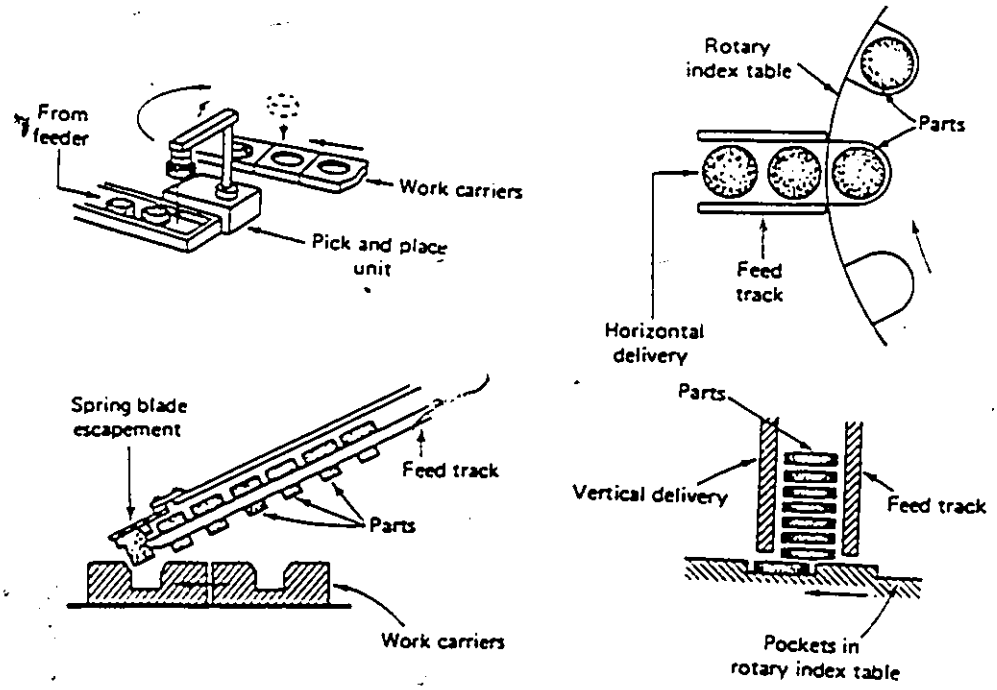


Figure 2-4. Part Feeders.
(Taken from [GROO8Ø])

in the fixture for the operation. Pallets can be designed to be used for a variety of similar parts. The disadvantage of pallets is that they must also be built and this is costly. To avoid this cost, the workpart itself is often designed to also be the fixture for operation from workstation to workstation.

Automatic storage and retrieval systems transfer pallets of material into or out of storage racks up to 100 feet high. Smaller systems called mini-loaders hold drawers of small parts. In both cases a part can be selected by number or by location, and an automatic warehouse in which the shuttle takes the place of the fork-lift truck and its human operator. Similarly, automatic guided-vehicle systems replace conveyors and hand trucks for transporting materials to and from the warehouse and throughout the factory. The driverless shuttle cars can be guided by signals sent through a wire embedded in the floor.

Another aspect of transfer systems that is used to advantage is the buffer storage capacity. Storage zones are used to collect workparts along the manufacturing line to permit mating different intermitted transfer systems, or mating non-synchronous transfer systems with work stations in the line. They allow the connection of one flow line with several others to reduce the effect of individual workstation breakdowns in the production line. The buffer storage capacity also helps in smoothing out the effect of variations in cycle times that may occur between work stations or between work stations and manual stations. Some of the disadvantages of buffer storage are increased factory floor space, higher in-process inventory, more materials handling equipment and grea-

ter complexity of the overall flow line system.

To correctly use advanced transfer systems, appropriate control mechanisms must be provided. This often requires some elementary feedback sensors and controller to detect proper sequencing and hazards, and even to monitor quality of the workparts.

In automated assembly systems where a number of robot manipulators are used, a method of efficiently transferring parts, components and partially finished products is required. This is normally done using an automated flow line. It consists of several work stations linked together by work handling devices that transfer parts between stations. Product transfer systems can be viewed as part feeders but at a higher level and are mostly general purpose conveyor systems. In many situations, product transfer systems may not require very complex resources because each workstation can place its output in a known or desired position. Thus transfer is done without losing orientation and so reduces extra processing that would be necessary to determine positioning information.

Transfer systems are optimized to reduce labour costs, to increase production rates, and to reduces the work in progress with the associated reductions in inventory. Optimization is done by minimizing distances between work stations, by minimizing the number of movements made in the workstation, by allowing for specialization of production operations in some cases and by integrating production operations in others. In many instances, robot manipulators form an integral part of the transfer system.

2.2.3.3 Control of Transfer Systems

There are two types of control mechanism found in transfer systems: passively and actively controlled transfer systems.

Passively controlled transfer systems operate in completely deterministic environments where all information on the parts is known or is assumed known. Information such as position, orientation (using "chocolate-box" approaches) and condition of the part is are contained implicitly in the workstation or are assumed to be correct. In these transfer systems, no processing is done to react to changes since changes are not allowed. Transfers are done in a blind way. Examples of this class of transfer system are conveyor belts and chain systems where simple pick-and-place operations are done. These systems are not usually found in this elementary form in automated assembly environments.

In actively controlled transfer systems an active role is taken in the movement of objects. Actions are taken based on environmental changes detected by a sensory system. Production systems that use active transfer systems often have bins where information on part position and orientation is lost. The transfer system can be designed to determine part position and orientation while transferring the part from one workstation to another. The transfer system can be designed to perform other tasks such as parts inspection while the part is in transit. Based on the information collected, the parts could be rerouted. These systems aid in off-loading the processing that would be

done by work stations or a master control center in order to reduce system complexity and improve system performance and productivity.

2.2.4 History of Automated Manufacturing Summary

The main driving forces in automated manufacturing have mainly been through increasing tool automation and improving product transfer in the manufacturing system.

Complete tool automation is achieved through computer programmability. The production program is entered into the controller part (often a mini- or micro-computer) of programmable tools at the start of the operation. The same part program is repeated to produce many copies of the part being machined. With programmability, manufacturing flexibility is achieved whereby new and different products can be made without resorting to a new tool, but by simply changing the program or set of algorithms controlling the tool. In an automated factory, the flexibility of the tools is essential in accommodating product changes [GRO08Ø].

Programmable tools used in an automated factory do not always operate in an open-loop control fashion, but are often augmented with feedback sensors permitting them to operate in a closed-loop fashion. This augmentation permits automated tools to perform properly in the face of disturbances. By providing sensors and decision capability to the controllers of automated programmable tools, complex control and sequencing strategies can be used to react to sensory information, and to ensure correct execution of the jobs. This allows the tools to operate largely

unattended. In a factory with several programmable tools with sensors, there must exist a control and coordination system between the tools to achieve continuous reliable operation or production.

As tools become more complex, more complex controllers are needed to control them. This task can only be performed by using computers. As these tools progress from a collection of simple machines, to networks of complex machines, coordination between various machines in a production system also requires the intelligent use of computers.

Transfer systems form a key function in a manufacturing system. They permit the orderly movement of parts, components, and subassemblies in the plant. The ability of the transfer system to move objects throughout the plant directly impacts the throughput of the manufacturing system.

With advances in mechanics, in computer control, in data communications, in computing technology (hardware and software required for controlling more advanced uses of robot manipulators such as those found in the assembly of finished products), full automation of the manufacturing plant is now within the grasp of industry. The most pressing requirement is the determination of a strategy that best integrates these technologies.

2.3 Computers In Manufacturing: Computer Integrated Manufacturing

In 1977, the following definition Computer Aided Manufacturing was proposed by CAM-I (International), as:

> **Computer-Aided-Manufacturing is the effective utilization of computer technology in the management, control, and operations of the manufacturing facility through either direct or indirect computer interface with the physical and human resources of the company.**

By direct, it is meant that the computer is used either to monitor or to control the manufacturing operations. By indirect, it is meant that the computer is used in support of the manufacturing activities in the plant, but there is no direct connection between the computer and the production process (production and inventory control, order scheduling, development of work standards, etc.).

Clearly, in the years since this definition was put forward, a veritable revolution in information technology has occured driven by both hardware cost/performance improvements in computing hardware and software technology. Computer technology has evolved to the point where its use is not only concerned with aiding and managing the manufacturing process, but is also concerned with the conception of the product itself right through design, to full automation of the manufacturing plant.

From [DWOR85] and [MERC85], Computer Integrated Manufacturing (CIM) is defined as:

> **a closed-loop feedback system that uses product concepts and requirements (needs and creativity) as input and produces finished goods (fully assembled, inspected, and ready for use) as output.**

CIM covers many aspects of the manufacturing process including Planning, Computer Aided Engineering (CAE), Computer Aided Design and Manufacturing (CAD/CAM), Computer Aided Testing and Inspection (CAT/CAI), Robotics, Flexible Manufacturing Systems (FMS's), Flexible Assembly Systems (FAS's), and other elements such as a common data base and non-geometric product information. To achieve full CIM, a consistent combination must be made of the software and hardware making up these elements whose aspects include product design (for production), production planning (programming), production control (feedback, supervisory and adaptive optimization), production equipment (including machine tools and robots) and production processes (removal, forming, assembly). The trend and impact in automated manufacturing can best be identified by examining the main aspects of CIM.

Computer Integrated Manufacturing is achieved by linking three main aspects of manufacturing: planning, product design, and manufacturing. All three aspects depend on their proper linking to allow the flow of information to be managed throughout the factory [NEVI82]. To integrate planning, product design and manufacture, very different types of information, data communication and processing must be accommodated.

2.3.2 Planning

When a product is to be manufactured, two types of planning are identified. The first is economic planning which involves making decisions to determine the economic feasibility and soundness of the product to manufacture. The second type of

planning is manufacturing resource planning.

Economic planning consists of market forecasting and production costing (often referred as the preplanning phase). In market forecasting, computers are used to process data from marketing surveys to determine product demand, market sizes, and other information upon which to make marketing decisions with respect to the viability of the product (marketing research phase). Costing is the economic process that determines the cost of raw materials and parts needed for production. Costing includes the cost of performing the production tasks (by machines and people: job costing aspect), and the cost of storage before, during and after the production cycle. Costing gives an estimate of the production cost of a new product. These data are processed by computers to help planners verify the market and manufacturing feasibility of the product, given the present and projected demands, inventory sizes and component availability. For proper market analysis and forecasting, planners must have access to large data bases of production data and accurate models to help analyze the data. The analysis determines what should be produced, and if it can be produced economically (go-no go decision). Computers are therefore used in a supporting role in the operations of the plant and are usually not linked directly to the manufacturing process. Computers are used mostly in an "off-line" manner to provide data and information for the effective management of production activities.

Manufacturing Resource Planning (MRP) is the management and control of available resources such as labour, machines, and

materials, according to changing demands. The main purpose of manufacturing resource planning is to maximize profits and productivity. Often, production systems are very complex making it impractical to obtain an optimal solution to resource planning. A number of suboptimal techniques are found instead that give acceptable results in terms of productivity and profits. Relatively simple methods of planning and control can cut down immensely on long waiting times and can eliminate most of the costs associated with inventory.

Material requirements planning is done to obtain the precise determination of the raw materials needed at different stages of production. Material requirements planning is a set of logically related procedures, decision rules and records. They are designed to translate a "master production schedule" into time-phased net requirements and the planned coverage of these requirements for each component inventory item needed to implement the schedule. In this aspect alone, large amounts of data demand the heavy use of computers.

MRP is the precise determination of the manufacturing line capacity, of production tools requirements, and the human and computing resources needed to carry out the design and manufacturing steps. Manufacturing resource planning is used in predicting the demand for each element in the manufacturing process at a given time. The scheduling of labour, materials, machine time and other resource elements that go into the manufacture of the product can be estimated by extrapolating backward from the

delivery date for the assembled product. If the scheduling is done accurately, there is no need to maintain parts inventory because of the uncertainties in the demand for parts; instead each part can be manufactured just before it is needed.

To have successful manufacturing resource planning, accurate information is required on the parts needed for each stage in the assembly of a product, on the time needed to manufacture each part (work time, setup time, transit time, waiting time), on the lead time needed for purchasing parts from suppliers and on the plant's own inventory. This scheme works quite well in job shops where many parts are manufactured in varying quantities. This type of resource planning system depends on detailed, centralized planning of all subassemblies, components and raw materials and on efficient feedback from every station.

Resource and material planning help in establishing the job manufacturing cost.

2.3.2 Product Design: Computer-Aided Engineering

Another aspect of CIM that is undertaken is the design and development of the product itself. Using appropriate tools, the design of the product can be carried out. The use of computers for this aspect of manufacturing is commonly referred to as Computer-Aided Engineering (CAE). CAE includes Computer-Aided Design (CAD), and Computer-Aided Manufacturing (CAM) and Computer-Aided Testing and Inspection (CAT/CAI).

2.3.2.1 Computer-Aided Design

Computer-Aided Design (CAD) systems allow the designer to describe, manipulate, and simulate parts and products to be manufactured. The type of processing involved in CAD is in general engineering computing for design and analysis functions. CAD systems permit the rapid prototyping of designs. Advanced CAD systems can even eliminate the need to make real hardware prototypes. With the aid of such design tools, the designer can perform an engineering design analysis quickly for several alternative design solutions. Modern CAD systems are highly interactive man-machine systems that aid in increasing user productivity and efficiency. The designer can even check for the proper design of parts that will be mated or assembled during simulation by moving the part images on a screen. Accurate specification and verification through simulation helps to qualify the final product for real production.

There is a large amount of design and production information generated in CAD. To achieve CIM, information, such as images and product data, are stored in a data-base. Design information becomes highly accessible to other manufacturing aspects and can be easily modified several times without large amounts of effort and time. The information stored allows the planning and scheduling of manufacturing functions to be started earlier.

One approach to data-base for CAD information storage is group technology. Group technology is a storage and retrieval system that permits those parts that are similar in design and

use the same set of manufacturing steps. Parts of similar design and manufacturing can be designed and processed in the same way an existing part has been designed. The same production procedure can be used in the manufacturing of a family of products by simply providing the data set defining the specific product attributes to the process machines.

Group technology allows the use of a data-base system used in the planning stage of manufacturing from which part information can be extracted and sorted according to various part characteristics. Part information can be retrieved according to characteristics such as size, shape, volume, weight, and materials used, and for manufacturing-process characteristics such as machinery setup time, machining sequence, and the usual lot sizes. By indicating the differences to the an existing manufacturing process of a part, the new part manufacturing process can be specified (variant processing). Group technology results in large labour savings since knowledge of similar parts can be reused in the manufacturing of the new part. Grouping production machines in stations or cells together according to the parts they make, allows higher production rates and efficient use of machinery.

In machining, CAD systems help generate the NC programs for parts manufacturing. This task is known as computer-assisted NC part programming, where computers are used in the design and specification of the parts to be manufactured, and involves sophisticated methods and techniques using computers to ease the task. Computer-assisted part programming is often performed by using an interactive parts design capture system also based on

computers (using a special part programming language or graphics editors). A scheme based upon such a system is often used to help the parts designers to control the design process and to save time. A part programming language is used by a part programmer to describe the necessary cutting motions in an English-like notation which is then "compiled" (translated) into a standardized "cutter location file" (CLFILE). This CLFILE is then processed by special "post-processors" for each machine into explicit instructions for that machine, usually on punched tape.

Although these systems are described as CAD systems, they are really application specific automation tools. The main thrusts of these systems are to provide assistance for design needs to users by applying the strengths and capabilities of automation (computers). Such systems permit the integration of specialized information for the application through data-base services and processing. With such integration, the user can make intelligent and informed decisions with respect to the application. Some of these systems are also designed to aid in project management to accurately track a project's progress, minimize design time, reduce the amount of information that must be processed or deciphered by the user during design, and accurately document the work with reduced effort. These systems are presently being created for Computer Integrated Manufacturing. Using sophisticated CAD tools, remarkable improvements of productivity are obtained.

2.3.2.2 Computer-Aided Manufacturing

A major function of CAM is the matching of the production process to the available resources. In this aspect of product design, a careful study of the tasks must be done in order to determine the partitioning of these tasks, and the routing scheme (sheets) or the flow of the product through the production line. Also, other aspects are studied such as the development of work standards, a time study of the direct labour involved in a production system without complete automated production.

Computers are used in the actual implementation of the process steps using data on the available resources and the allocation of work on the production line. Computers are used to manipulate data on tool availability, materials requirements, flow of parts being manufactured, work-in-progress inventory, and finished goods inventory, among others. This is known as line balancing. Line balancing involves the minimization of unit cost of operation or maximization of the production rate by determining what can be done by automated machinery, and what kinds and amount of data is required for these devices. The technique used to help in manufacturing scheduling is that of group technology, where different parts with similar geometric features and manufacturing characteristics are classified and are scheduled based on this categorization.

Using group technology, families of parts can be selected for machining. Workpieces move on pallets automatically from tool to tool for machining in the proper sequence. High efficiency can be

achieved in such systems by keeping the tools working as much as possible. In an automated system a machine could be kept cutting 50 to 90 percent of its time while in computer-numerical controlled systems, cutting time may be as low as 10 to 30 percent of the total shift time [GUNN82].

Another main function of CAM is to achieve active control of processes and timely dissemination of production data that narrows the response time required for the entire manufacturing operation to make decisions and cure production problems. This active involvement of the computer in the manufacturing process and the control of production is the basis for CAM.

2.3.2.3 Computer-Aided Testing and Inspection

With a set of sensors and testing equipment, automated testing and inspection can be done to avoid ruining an entire batch of parts because of mechanical failure or programming error. The information from sensors can be used to accept or reject individual parts. This information can be kept for later statistical analysis in a statistical data base use to verify quality of production. With even more integration, this data can be analyzed immediately and result in control adjustments during production.

In CAE, computers are used extensively to generate product specifications. The specifications can be captured using various design capture and simulation schemes and tools. These allow for the formal description of parts using high level part description languages and verified through simulation. The amount of data

generated can be enormous. In many cases, the data generated are translated into control commands and are used directly in production. In order to take advantage of computing technology in manufacturing, a CIM system must integrate all areas of CAE.

2.3.3 Modern Automated Manufacturing Areas

The only true on-line use of computers in modern automated manufacturing is in the production phase where raw materials and parts are transformed into finished products. Computers are used both to directly control specific machines, and to coordinate the machines in the production line. Even though these are very distinct activities, in most existing systems, these activities are lumped together into a single overall control activity.

The overall control function can be implemented either on a single centralized computer, or distributed among several computers. The control achieved may be preplanned, where computers control a process through a predetermined series of steps like in direct numerical control (DNC), or adaptive, where computers monitor the performance of the process and continually redetermine how optimum performance can be achieved in the face of disturbances.

In modern automated manufacturing, two manufacturing types are identified: aggregate product and structural product manufacturing.

2.3.3.1 Aggregate Product Manufacturing

Process control, also known as hard automation, refers to the manufacturing of continuous or aggregate products. It can be further subdivided into continuous product processing and monitoring and regulation control. Most process control systems must operate in real-time, within well defined limits. Considering the speed of operation of present day computer hardware and the process control times that must be met, the speed of response is not too demanding. Many of these systems require uninterrupted operation, often demanding the implementation of complex failure tolerance (in hardware and software) strategies.

i) Continuous Product Processing

Continuous product processing generally refers to the refinement of a product or the controlled generation of raw materials. In continuous processing there are a relatively limited number of product choices. Good examples of this type of process control applications can be found in many industries such as the petrochemical, pulp and paper and other chemical industries. In such installations, the coarse refinement of many products is carried out: oil, chemicals such as soaps, detergents, plastics, solvents, and papers of different grades and quality. In general, the required manufacturing plants are elaborate and costly producing only a previously defined class of products. Also, in such systems, there is not a great variety of products that can be manufactured economically, even in large batches. The control in such systems is quite well defined and their event sequence is

well known, with very little, if any, changes in both. This automation method is often referred to as hard or coarse automation.

ii) Monitoring and Regulation Processing

Under monitoring and regulation processing, both the regulation of product flow and the distribution of the product is considered. Regulation of product flow involves the monitoring and maintenance of the flow at the required levels. Examples of such systems range from electric power plants, to various traffic control systems.

In distribution systems, control is required for the proper allocation of the product or resources in response to varying demand. Examples of distribution systems are electric power distribution (on the power grid), oil pipeline control, and different routing systems. More recent examples are the automated materials handling systems.

In both monitoring and regulation, computers are needed to continuously collect manufacturing and performance data from underlying system, and make well defined decisions based on a predetermined strategy.

2.3.3.2 Structural Product Manufacturing

Structural product manufacturing refers to the generation of finished discrete and finished assembled products.

i) Discrete Parts Manufacturing

Discrete parts manufacturing is similar to process control, can be separated into two broad kinds of discrete parts manufacturing: large and small production runs.

In modern large production run systems, many aspects of manufacturing must be considered. For the most part, they include the manufacture of goods, starting from the raw materials to the final product to be used or sold. Since the large numbers justify the investment, the manufacturing system can be as complex as needed. Discrete parts made are often used to build other products manufactured in large quantities that may require assembly. An example of this type of manufacturing can be found in screw manufacturing. Even though the manufacturing process can be very complex and quite flexible to allow for different types, there is no assembly. That is the products of such manufacturing systems are not composite objects. These screws are then used in the manufacture of other products such as automobiles involving different and more complex devices and setup. The combination of both manufacturing of smaller atomic parts and successive manufacturing of a large number of finished products is also known as Detroit type manufacturing because of its roots in automobile mass production.

The main characteristic of such systems is the number discrete products being manufactured. Large production runs usually have more than 300,000 issues [NEVI78] of the same product [GROO80]. The actual figures depend mainly on the demand of the

product and the cost of producing them. For instance, a particular consumer product may only be produced by a specific special purpose set-up if the number of products exceeds a predetermined threshold, dictated by the economies of scale in its manufacture.

In small production runs, a much smaller number of products are made, usually less than fifty issues [NEVI78]. The products are usually simple to manufacture, such as special screws, using similar automated machining devices used in large production runs. The production cost per part manufactured is greater since the same amount (or more) of set up and control must be developed for a small number of parts manufactured, compared to large production runs.

ii) Automated Assembly of Products

The task that sets this automation manufacturing type apart from the other type is assembly. Automated assembly involves the use of discrete parts, composite objects, subassemblies consisting of assembled parts, and highly advanced automated devices in the manufacturing of finished products. Automated assembly types are also distinguished by their production run sizes.

In large production runs, literally hundreds of automated devices are used in an assembly plant to manufacture a large number of the same product. There is usually very little variation in the product being made, and very little variations in the actual control of all the devices. The tasks have been split into very simple jobs to a point that they are done very quickly using very advanced machinery and with very high reliability and quali-

ty of product. Different models of a particular product can be made since the product's underlying structure is usually the same for a variety of finished products. Often, the variety is simply cosmetic. Examples of such automation is found in some aspects of automobile assembly.

In small production runs, the goal is the economic assembly of a relatively small number of finished products, or to produce many different models of a basic product in small numbers, all at the lowest possible cost. The different product models are assembled by automated devices that can be dynamically reconfigured to do many complex tasks (such as part insertion and fastenning). To accomplish this, the manufacturing plant must be very flexible, and be easily and economically reconfigured or "retooled" to produce the product as economically and as quickly as possible.

2.3.3.3 Modern Automated Manufacturing Summary

Modern automated manufacturing involves the manufacturing of aggregate and structural products.

Aggregate product manufacturing is concerned with continuous production of refined or generated materials and monitoring. Flexibility in the manufacturing systems for aggregate product manufacturing and monitoring is very limited, although a wide range of products can be refined or generated using the same process mechanism.

Structural product manufacturing is concerned with the manu-

facturing of discrete and assembled products, both with large and small production runs. Discrete product manufacturing involves only the production of separate units from "raw" materials, usually use more fixed automation techniques, with little or no variety in products being manufactured. Discrete part manufacturing does not involve any assembly task. Flexibility in the manufacturing system is paramount for assembled product manufacturing especially in small production runs since it must be easily and quickly reconfigured to produce potentially completely different products. Large production runs can also benefit from any flexibility even if it is only to reduce the reconfiguration time.

2.4 Concluding Remarks

In this chapter, it was shown that in the evolution of manufacturing there is a natural progression from hand tools to powered tools, to automated and programmable tools. As tools become more complex, more complex controllers are needed to control them. This task can only be performed by using computers.

In modern automated manufacturing the human worker is being replaced at various levels in the production system, by first replacing the tool operators, and then tool supervisors and finally, production management. Automation is used to obtain more efficient and higher quality production (increased productivity, quality, flexibility, etc.). As tools become more automated, they become less dependent upon a worker's control, and more under the control of production planners.

In the modern manufacturing systems, the robot manipulator augmented with advanced sensory systems and high level programmability is becoming the most prominent device in the manufacturing of finished assembled products in the near future. To identify the kind of control and coordination needed to take advantage of this technology, robot manipulators will be analyzed in greater detail in a later chapter.

Programmability and feedback allow the tools to operate largely unattended. In a factory with several programmable tools with sensors, such as robot manipulators, there must exist a manufacturing control and coordination system linking the tools to achieve continuous reliable operation or production.

With advances in mechanics, robotics, sensor systems, computer control, computing technology (both hardware and software required for controlling more advanced uses of robot manipulators such as those found in the assembly of finished products), and data communications, full automation in manufacturing is now within the grasp of industry. To achieve full automation in manufacturing several areas of design and manufacturing must be well integrated. The most pressing requirement is the determination of a strategy that best integrates these technologies resulting in a Computer Integrated Manufacturing (CIM) system.

To realize full integration, demands in advanced data processing must be met before benefits of linking them can become significant. CIM relies in the need for a carefully designed hierarchy of information flow from the planning and design levels

down to the low level control primitives of the manufacturing elements. This information flow must allow quick and easy reconfiguration or retooling of the manufacturing system under changing production conditions and product changes. The factory floor realization of the manufacturing level of CIM is the Flexible Manufacturing System (FMS).

The only way to have full automated manufacturing flexibility is to design a more efficient automated facility with the properties of easy reprogrammability and adaptability through reconfiguration in hardware, and in software. For this kind of manufacturing, many advanced automated production devices must be linked. As tools progress from simple machines to complex programmable machines, coordination between various machines becomes a dominant requirement in a manufacturing system.

To link automation devices requires that proper control and coordination be achieved between the devices operating on the product. Proper control is needed in the sense of partitioning manufacturing tasks among a limited number of devices, taking into account the individual flexibility of these devices. Coordination is necessary in the cases where a pipe-lining of tasks is done in multi-workstation-based FMS's. In the case of cooperating machines executing a single complex task, even more coupled control and coordination is needed. In small production runs, especially in assembly, tasks require an automated setup to make them economical.

A manufacturing system for small production runs must be very

flexible, be able to undertake a very wide range of manufacturing and assembly tasks for a large number of different products, and at the same time, must be easily reconfigured for such tasks. This means that such manufacturing systems must consist of highly flexible elements. Meaningful schemes and tools must be provided to make FMS's for CIM a reality.

Chapter 3

Flexible Manufacturing Systems

3.1 Introduction

Manufacturing lines are in general the result of a production method based on modularity where a product is manufactured from a set of interchangeable components. The materials and components are fabricated in batches and are fed in the proper order to the appropriate work stations for the manufacturing of the product. The manufacturing process is systematically subdivided into a set of distinct activities in order to pipeline production. Large production runs justify the use of specialized machinery in the automated line. The use of specialized machinery results in large investment costs and in very rigid or inflexible manufacturing lines. In practice, production lines have additional requirements to allow for minor product customization. Customization is realized either by the matching of components or by increasing the complexity and flexibility of the production machines. When a new product is to be manufactured, large costs are incurred because complete retooling of the line may be necessary.

Increased world competition, labour costs, the need for reliable products, reliable production, and the availability of advanced automation technology, are factors that have forced manufacturers to redesign their plants based on flexible manufacturing systems.

In general, for small production run manufacturers, flexible automation is the only way to survive. Plants with small production runs must be able to produce a variety of similar products (e.g. different electric motors, similar consumer appliances, etc.). Continuous investment in new special purpose machines is not feasible in small production run systems.

The main feature of Flexible Manufacturing Systems (FMS's) is that they are easily reconfigurable to manufacture different products, both physically and logically. This easy reconfigurability allows FMS's to manufacture new products in a cost effective way with low set up cost. To allow for this adaptability without excessive cost, a complex manufacturing process must be partitioned into manufacturing activities and assigned to a set of general purpose reprogrammable manufacturing machines or elements. In the case of major product changes, a timely, cost effective, and complete reorganization of the manufacturing line must be possible and should not require new specialized manufacturing machines.

In this chapter, design, computer control and coordination requirements of integrated FMS's are presented. The general FMS elements and their characteristics are first discussed. Computer organization approaches to computer control and coordination of elements in FMS's are then presented. Based on previous work [PICH83], a generalized control scheme encompassing the different control approaches presented is then proposed. This scheme can be considered as a general methodology and notation for describing or specifying computer control organizations.

3.2 Flexible Manufacturing Systems

Ideally, the elements in a manufacturing system should be flexible enough to undertake an extremely wide variety of activities including assembly. It must be governed by a control mechanism that permits vastly different manufacturing activities to be executed while still being easily reprogrammed.

3.2.1 FMS Characteristics

Specifically, a Flexible Manufacturing System must have the following characteristics:

1: The manufacturing process must be partitionable:

a) it must be functionally and physically divisible into independent activities for the different production steps;

b) the activities must be subdivided into a set of primitives directly executed by adaptable machines for different products;

2: The FMS must:

a) be implemented from a number of flexible elements called Flexible Manufacturing Cells (FMC) interconnected by a flexible and efficient Automated Transfer System (ATS);

b) be coordinated to provide:

i) proper sequencing of FMC activities to perform the needed manufacturing process on the given product;

ii) proper flow of the product and components between the different FMC's;

iii) continued operation despite local failures or minor modifications in the system;

3: To effectively use flexibility, the FMS organization must allow for customization of products being manufactured; the FMS organization must permit:

a) the partitioning of the affected manufacturing process into separate activities encompassing the new activities;

i) to provide a straight forward means of specifying the new coordination between FMC's in the FMS;

ii) to coordinate the old and new activities done by FMC's on the objects, with the movement of objects using the ATS;

ii) to identify potential failures and define a contingency plan allowing continued operation of the line;

b) the reorganization and reprogramming of the FMC's to execute the new or modified activities;

4: Ideally, the effort in reconfiguring the FMS completely for different classes of products must be low; this can be is done automatically provided that reconfiguration:

a) is part of the new product design process;

b) is also part of the manufacturing system up design process;

A Flexible Manufacturing System (FMS) is therefore a flexible, integrated and reconfigurable automated computer controlled manufacturing system consisting of coordinated Flexible Manufac-

turing Cells (FMC's) linked by an Automated Transfer System (ATS) which work together to produce finished products from raw materials, parts, and components. A FMC is an element of the FMS composed of flexible controlled entities that together are used to execute manufacturing activities. The coordinated set of activities form a manufacturing process.

3.2.2 FMS Subsystems

There are three classes of subsystems in FMS: Automated Machining Systems (AMS), Flexible Assembly Systems (FAS), and Automated Transfer Systems (ATS). There are two types of Flexible Manufacturing Cells (FMC): Automated Machining Cells (AMC), and Flexible Assembly Cells (FAC). The Automated Transfer System (ATS) forms a integral part of an FMS and is imbedded into AMS's and FAS's. The ATS binds the the cells in a subsystem together to form a working whole. It consists of passive or active (controllable) transfer elements such as those mentioned in the previous chapter: conveyor systems, pallets and fixtures, bins, mobile vehicles, etc.

An Automated Machining System (AMS) is a FMS subsystem consisting of Automated Machining Cells (AMC) linked by a ATS. In the case of AMS's, the ATS is often referred to as a Materials Handling System (MHS). The ATS transfers raw materials and machined parts under work between AMC's in the manufacturing of finished parts. An AMC is itself an automated machining subsystem designed to convert raw materials into usable parts by performing molding or shaping activities on materials. An AMC may include

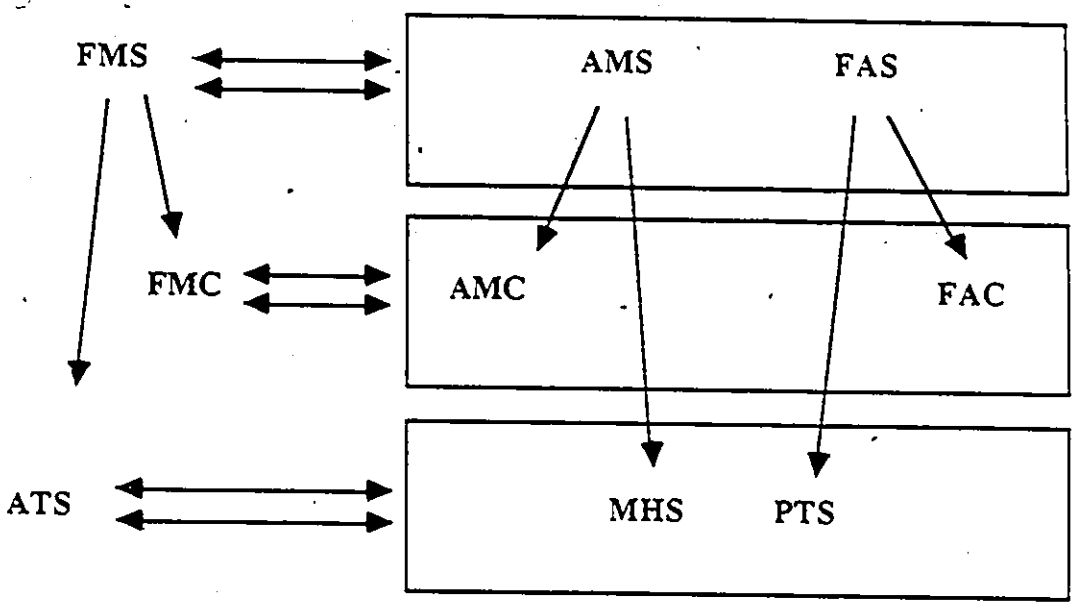


Figure 3-1. Flexible Manufacturing System and Subsystems.

robots, sensory systems, and materials feeders in addition to CNC machines and the like. Although manipulation of materials and parts may be performed, no assembly activity is done in an AMC's. An AMS is the functional binding of AMC's with an ATS, that transforms raw materials and machined parts into a variety of finished parts.

A Flexible Assembly System (FAS) is a FMS subsystem consisting of Flexible Assembly Cells (FAC) linked by a form of ATS called a Product Transfer System (PTS). A FAC is a computer controlled assembly subsystem designed to perform assembly activities or tasks by combining or mating components and parts to form composite or assembled products. The main elements of a FAC usually consist of robot manipulators, part and component feeders, assembly fixtures and complex sensory systems, occasionally special purpose assembly devices. Although some machining type of operations may be performed (e.g. drilling, bending), the activities that distinguish FAC's from AMC's are assembly activities. The PTS moves parts and components to FAC's, as well as unfinished assemblies or subassemblies between FAC's in the manufacturing of assembled products. A FAS is the functional binding of FAC's with a PTS that together can accept parts, components and subassemblies and assemble them into a variety of finished products.

3.3 Approaches to FMS Control and Coordination

In order to integrate the above elements of a FMS, an appropriate strategy must be taken to control and coordinate all the

subsystem element activities (in FAC's and AMC's) with the ATS such that manufacturing actually takes place. Also, this strategy must conserve the desired flexibility of the system, otherwise all potential gains provided by using the FAC's and AMC's may be lost.

In this section, an overview of conventional organizational control and coordination strategies used in manufacturing plants is presented outlining their advantages and disadvantages with respect to a FMS environment. A simple and reliable solution to the design of these large systems must be developed to handle complexity of these systems while maintaining total system coordination vis-a-vis unforeseen occurrences. In this section, a generalized scheme with a notation is presented called oligarchical, used to specify system control and coordination in complex organizations. This scheme allows for the description of new organizational alternatives to implementing FMS control systems.

3.3.1 Conventional Organizational Approaches

There are three main conventional organizational approaches to control and coordination of large systems: fully centralized, distributed, and hierarchical.

3.3.1.1 Fully Centralized Organizations

In fully centralized organizations, there exists only one controller-coordinator in charge of all entities (see Figure 3-2). All computations, control and coordination decisions are made by a single central computer. A single complex control program or

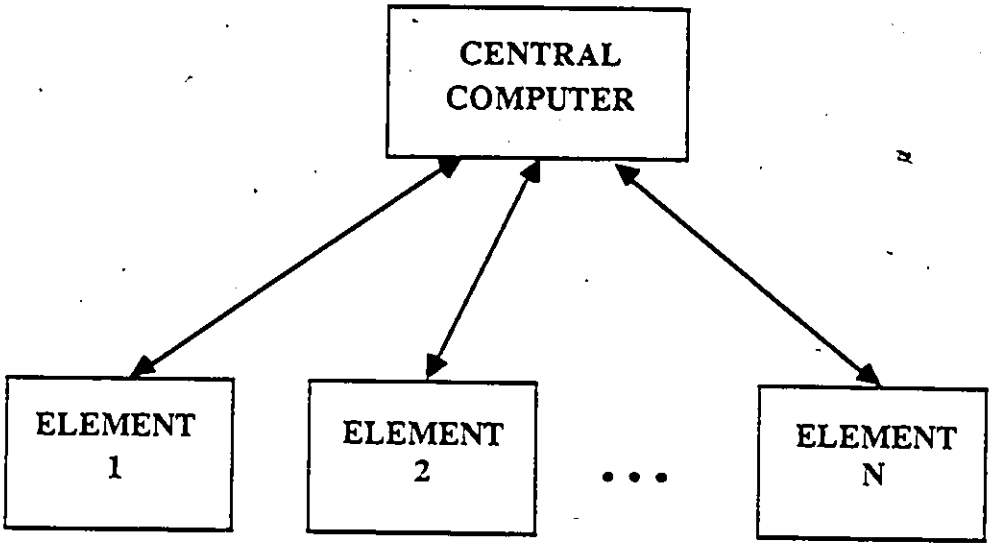


Figure 3-2. Fully Centralized Organization.

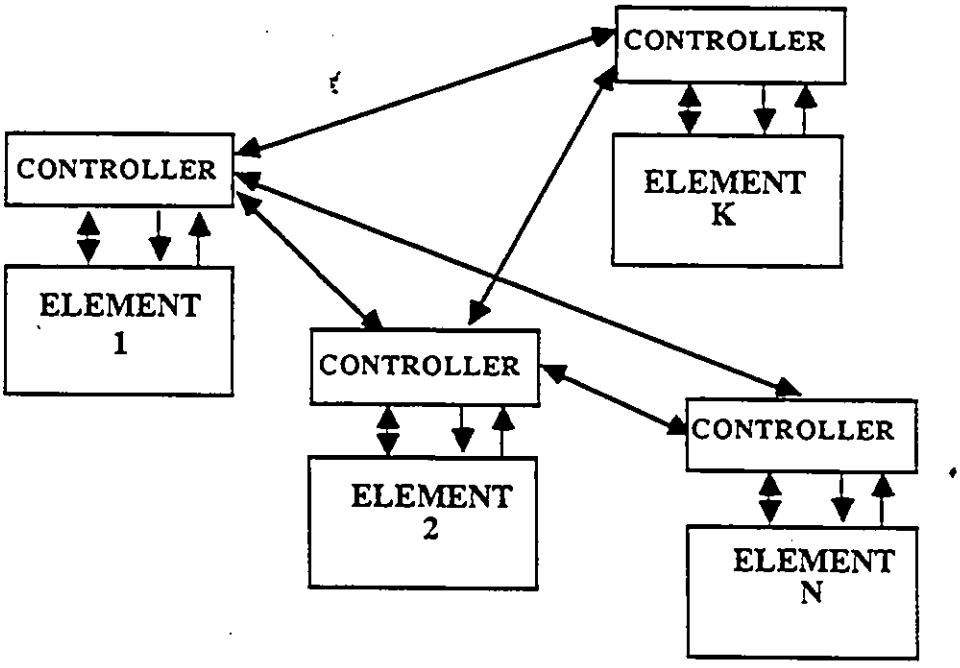


Figure 3-3. Distributed Organization.

executive performs all the control and computing tasks.

The main advantages of fully centralized systems are:

1: a minimum amount of hardware is needed for computing tasks and information flow;

2: fast response time for complex control operations is achieved; all subsystems are accessible directly and in minimum time;

3: quick and accurate decisions are reached and system status is updated quickly because all the information is centralized;

4: since all the data are contained in the central controller, there exists a form of data integration;

The main disadvantages of such a system are:

1: the controller complexity increases rapidly with the number of entities in the system above a certain complexity level; the centralized nature of these system leads to severe bottlenecks as the size of the system increases, thus effectively reducing the response time; only one controller exists with which each controlled entity must communicate; high speed communication links are required between central computer and controlled entities; more intelligent local controllers could be put in charge of the controlled entities to alleviate the bottleneck problem, but this changes the control system organization;

2: very complex software is required; the controlling program is difficult to create and validate and the maintenance cost is very high; when a single entity must be added to the system,

difficult growth is encountered because major software redesign have to be undertaken to meet the timing and interaction requirements; the system must control and coordinate either a larger number of entities or more complex ones, during the same interval of time; dynamic reconfiguration is impossible;

3: the single controller is highly vulnerable to hardware and software failures; the overall operability relies on the proper functioning of the single controller, the central computer, and its software; by providing redundancy in the system, system integrity can be increased, but at the cost of extra hardware and difficult system management;

In single computer fully centralized control and coordination systems, fast response time for complex control operations can be achieved at the cost of very complex software. In working systems, response time is found to be very fast because once the system has been completely organized and programmed to work, all the controlled entities have been taken care of in the alloted time and in the correct order. Using a complex and costly refinement process, the system can be tuned to completely control the entities correctly and at the proper time.

An extension to fully centralized control and coordination is known as a Federated System [PAQU85]. In these systems, there exist a master central computer which remains in charge and many slave computers that perform low-level specialized tasks. All computers are linked together through a multiplexed data bus and all communication is done by data transfers. Since only there is

only one central controller, these systems still have many of the qualities of the single fully centralized control, except that some modularity and functional independence is introduced via the slave computers.

3.3.1.2 Distributed Organizations

The FMS control and coordination system can also be decentralized or distributed. A distributed system, shown in Figure 3-3, is a set of nodes communicating via a suitable network. The network and links make information in a central data base accessible. Each node or entity in the system can be viewed as a separate processing element responsible for the control its manufacturing activities.

In a distributed FMS control and coordination system, each system entity is assigned a controller with dedicated activities. The controlled entities have dedicated local resources and local control programs. Application programs are specific to the local activities. The manufacturing activities are partitioned such that each controller performs one or more activities necessary to the manufacturing of the product. To allow for interaction between entities, each controller includes a coordinator which allows it to communicate with all other coordinators in the system. Each coordinator has the same level of capability. Decisions are reached by performing distributed computations, usually requiring the participation of many or all coordinators. The control program (executive) is therefore distributed among the entities.

Advantages of distributed systems are:

1: relatively simple controllers;

2: good tolerance to local failures;

3: the coordinators are mainly independent and have a defined communications interface and interconnection scheme; no central switching element exists where decisions are made or communication links are set up;

4: extremely flexible systems permitting straight forward partitioning and integration of functional program units of software, hardware, and new technologies;

5: reliability and survivability are greatly improved with autonomous subsystems; vulnerability is reduced through locally dispersed computing resources that can be reconfigured;

6: with immediate local feedback to the controllers of the FMC's in the factory, permitting the continuous adjustment of the planned flow of products through the factory;

7: by distributing the processing and providing local feedback, the design of the plant would emphasize flexibility, permitting a variety of products to be manufactured, even in unit quantities.

Some disadvantages of distributed systems are:

1: decreased system performance as the number of entities increases; the number of processors dictates their ability to respond quickly; since decisions are made distributively, they usually involve a large number of information transfers; the interaction between separate systems is less predictable or 'deterministic' than in centralized systems since the

controlled entities must operate without complete knowledge of the system state;

2: increased software overhead required for communication protocols and delays in processing due to waiting time in queues at the sending and receiving ends of the communications paths;

3: the units communicate using special protocols and interconnection systems to avoid network contention; these supplementary actions require even more time, and are often driven by complex software;

4: system-wide database management is required for functional system authority and validity.

Distributed control and coordination systems offer many advantages to the control of FMS's with low to moderate complexity. This type of control and coordination can suit FMS's with low to medium production runs with a limited number of FMC's. In large production run FMS's with a large number of FMC's, distributed systems may be used to achieve the needed flexibility but at the price of performance. For this reason, fully distributed systems may not be appropriate for large FMS's.

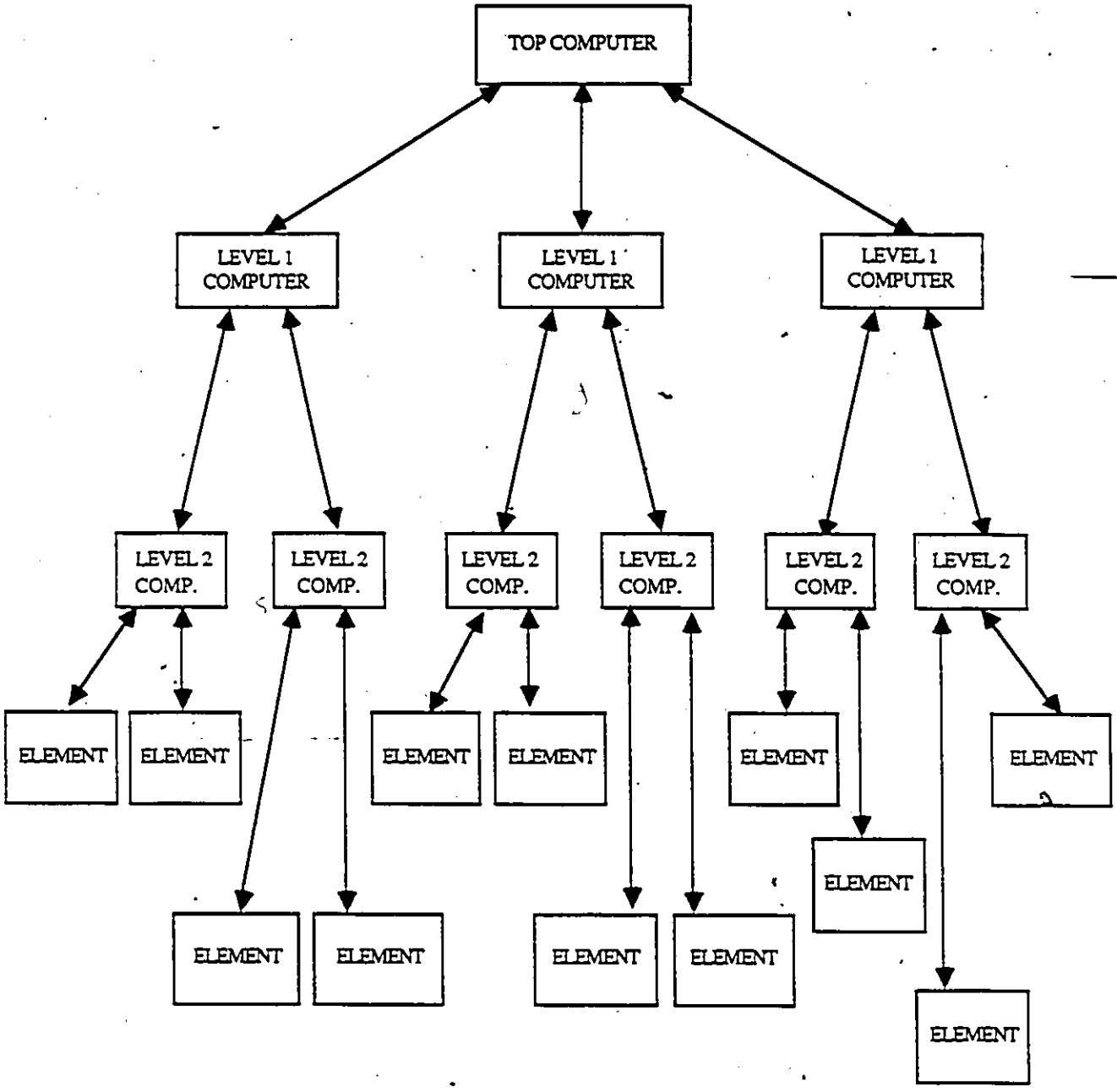


Figure 3-4. Hierarchical Organization.

3.3.1.3 Hierarchical Control Organizations

Hierarchical systems are tree structured organizations of processing elements where processor capability increases with the levels of the tree, as shown in Figure 3-4 [WEIT8Ø],[WILL78]. Subsystems are under the control of an upper echelon complex of processors that coordinate the operation of all the subsystem elements to perform the required activities. The lowest level elements are application dependent, special purpose and dedicated to a well defined set of activities. Custom or general purpose architectures can be used for processing elements as required. The top of the organization has a more general capability for controlling and coordinating the entire system. Hierarchical control systems store shared databases at the top rather than distributed throughout the system and information transfers are usually vertical (i.e. between the different levels of the hierarchy). The top of this organization is usually handled by maxi-computers, and the bottom is handled by mini-computers or micro-computers.

Advantages of hierarchical systems are:

1: system capabilities can be easily tailored to system requirements since subsystems can be isolated and be made independent;

2: with standardized communication links, these systems are easier to modify allowing for easier growth and permitting better integration of subsystems and straight forward reconfiguration;

3: hierarchical systems have reduced operational and support costs;

4: system integrity is preserved through independent redundancy and control allowing system fault-tolerance;

5: fault-tolerant reconfiguration is also possible.

6: functional isolation is achieved by using dedicated subsystem links and interfaces;

7: repetitive real-time functions are found at the lowest levels;

8: the organization provides for a natural and simplistic control structure matching the system's control hierarchy.

Disadvantages of hierarchical systems are:

1: systems may require a large number of processors to support the control and computing load demanded, although the low cost of microprocessor based processing elements counteracts this problem;

2: with such a large number of processors, potential system allocation, control, and coordination problems may arise frequently, and more overhead resources may be needed to resolve these problems;

3: software failure handling features are difficult to include without unduly increasing the system complexity;

4: beyond a certain complexity, hierarchical systems become very stiff with respect to any modification;

5: the response time degrades rapidly as control processing requirements go up in the hierarchy.

6: control hierarchy requires multiple levels of data communica-

tion links, although a high speed global interconnection is often found in such systems making the system appear distributed;

7: data rates increase as data migrate up the tree (unless data reduction techniques are used);

8: if the top processor fails, total system control may be completely lost;

9: if a failure occurs in any interconnecting link, all processors in the levels below it are disconnected from the system and complete system control is no longer possible;

Although hierarchical systems have many reliability traits similar to those of centralized and distributed systems, to ensure operation in a degraded mode or to do a safe shut down, they must have specific imbedded features. Redundancy is often used to improve reliability. Redundant paths may be provided to reduce failures in interconnection links, but is costly. To prevent total system failure, redundancy can be put at the top, but this means a doubling in maxi-computer costs. Also, system complexity increases when redundancy is used, as well as the software overhead. A doubling of hardware does not mean an increase in system reliability by the same ratio.

3.3.2 Oligarchical Control Organizations

In this section, a general methodology for control and coordination of interacting flexible manufacturing cells in a Flexible Manufacturing System is proposed based on previous work [KRIE82],[PICH83]. The methodology is called oligarchical control

and is mostly concerned with the aspects of total system coordination.

In an oligarchical system, the scope of the data manipulation is considered. In control systems, the scope of the data manipulation is usually very localized. Data pertain to the local situation only. Sending data to a central location to have them manipulated and retransmitting back should be avoided when at all possible. The oligarchical control approach provides a mechanism to minimize the amount of data transmissions during execution. This is done by incorporating into the local controllers the necessary computing power (mostly based on microprocessor technology) to perform the tasks at hand, leaving the aspects of system coordination to the coordinator levels above.

It should be noted that the system proposed is not a self-optimizing or self-learning system. In this system, the optimization has been done ahead of time, using a separate set of tools and design system.

General Organization

An oligarchically controlled manufacturing system is best described as a multi-layered organization whose lowest layer is composed of controllers directing manufacturing cells, and any other layer is comprised of coordinators. The various tasks of the manufacturing cells are specified as sequences of primitives in the local controllers, and are initiated by an action from an upper level coordinator (see Figure 3-5).

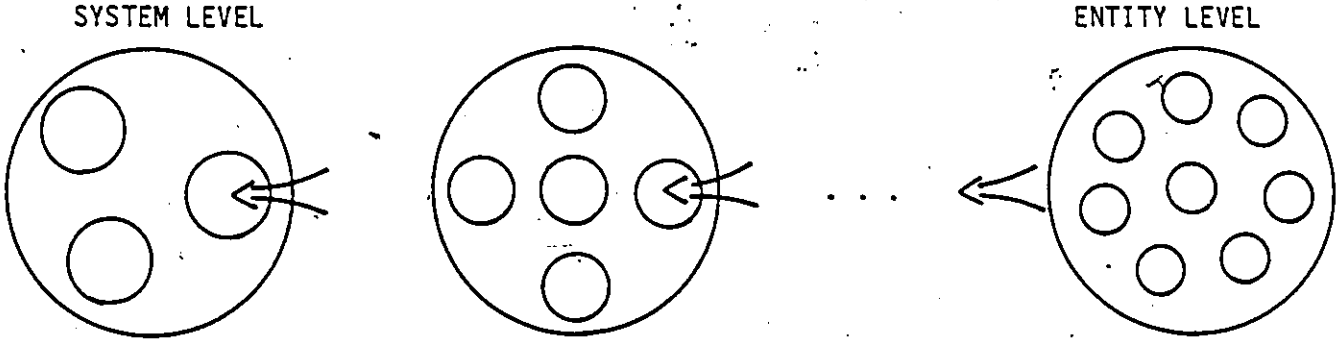


Figure 3-5. Coordination Levels.

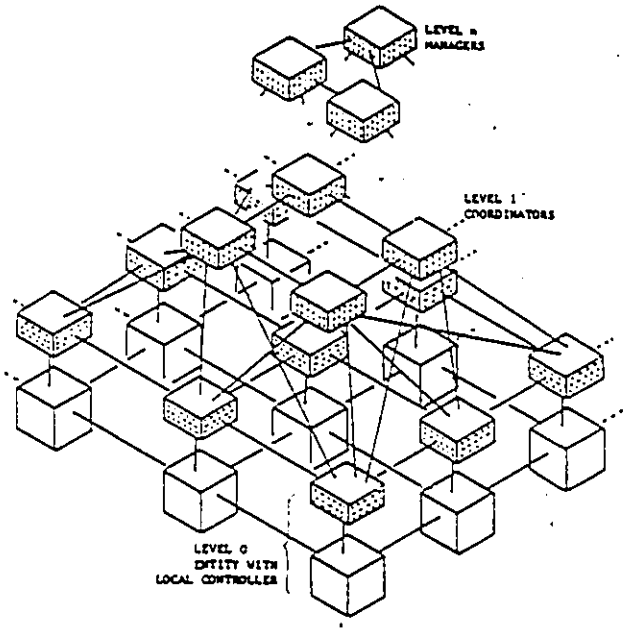


Figure 3-6. Oligarchical System Structure.

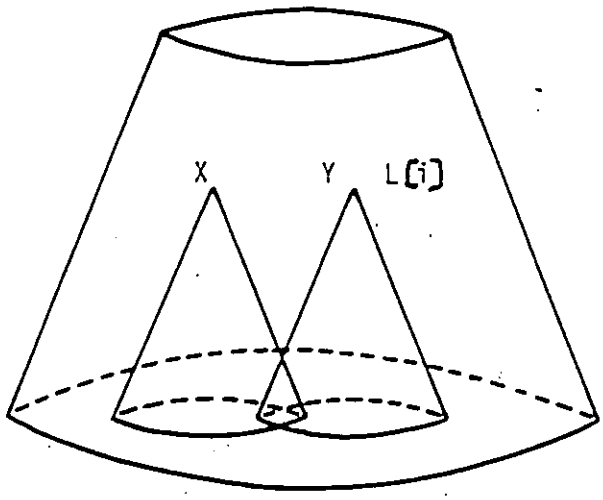


Figure 3-7. Views of Coordinators x and y at Level L[i].

The local controller makes all the necessary decisions and computations for the execution of a task. In the cases where interaction is required or a decision cannot be made by the local controller, a request for coordination is sent to the higher coordinators to which it is connected.

The coordinators evaluate local situation based on the status of the requesting controller and its subordinates, as well as status information of 'neighbouring' coordinators. If a decision is reached, commands are returned to the requesting controller, and all the other controllers within its scope, as required. In the case that no decision can be reached, a request is sent to the next level of coordination is issued. This process continues until a decision is reached about the original request, or until a timeout counter triggers a safe shut-down of the manufacturing cell and the related neighbor cells.

This coordination scheme allows local controllers to work together in the execution of a set of complex tasks; it also allows the isolation or the take-over of a failed device, thus providing some tolerance in device or process failure. In addition, the coordination scheme allows for the modular alteration of the manufacturing plant as production requirements change. This modularity is due to the fact that the coordinators are designed as simpler processing devices working in an event driven manner.

In the oligarchical organization, the set of manufacturing elements constitutes the base of the system. The control mecha-

nism is organized by levels: there are $n \geq 1$ levels of 'control' and three 'modes' of operation for each level: hierarchical, decentralized or distributed, and fail-safe. Each level **i**, where $1 \leq i \leq n$, is composed of a set **L[i]** of coordinators; each coordinator $x \in L[i]$ is directly connected to several coordinators at level **i-1** (in the case where **i=1**, the coordinators are connected to system elements), and to at least one coordinator at level **i+1** (except in the case where **i=n**). A general depiction of an oligarchical system is shown in Figure 3-6.

Additional specifications of the topology of an oligarchical system are:

1: $|L[i]| < |L[i-1]|$, i.e. the number of coordinators is decreasing in the levels;

2: $|L[n]| \geq 1$; i.e. the top level can have more than one coordinator, called managers;

3: each coordinator $x \in L[i]$ can communicate (either directly or indirectly) to all other coordinators in **L[i]**;

4: each level is 'globally connected' to the next levels, i.e. each coordinator in **L[i]** can communicate (either directly or via another coordinator in **L[i]**) to any coordinator in **L[i+1]**, and vice versa.

The level of coordination, as well as the size of the subsystem below it of which it is aware, determines its control capability. Formally, the view $V[x] \subseteq L[\emptyset]$ of a coordinator $x \in L[i]$ is defined as the set of elements directly connected to **x** if **i=1**;

otherwise, as the union of views of all coordinators **yeL[i-1]** directly connected to **x** (see Figure 3-7).

There are three kinds of (control) communication primitives: requests, control signals (or commands), and transfers. Requests are sent to the higher level, control signals are sent to the lower levels, and transfers are sent within the same level. Requests typically originate at the system element level. Associated to each request there is a scope which specifies the degree of decisional capability needed to resolve. When a request is resolved at level **i**, appropriate commands are sent to the lower level as specified by the control function **F[i]**. When a command is received at level **i**, the appropriate action is taken as specified by the processing function **G[i]**. When an element **EeL[Ø]** receives a command, it will perform the appropriate processing activities.

Each level can operate in three modes: hierarchical, decentralized, and fail-safe. In the hierarchical mode, when a request **p** arrives at **xeL[i]**, if **F[p]⊆V[x]**, (i.e. **x** has the control capability of resolving the request), it resolved the request; otherwise, it sends the appropriate request to level **i+1** for resolution. In the decentralized mode, any request arriving from level **i-1** is resolved collectively by the coordinators in **L[i]** in a distributed fashion. In the fail-safe more, a coordinator triggers a shut-down mechanism. In other words, a coordinator at level **i** is capable of resolving immediately certain requests; for other requests, it can either differ the resolution to a higher

level, or resolve it in a distributed fashion with the contribution of other coordinators in **L[i]**.

The decision on which mode of operation should be used is made according to the following rules:

1: coordinators at level i operate in hierarchical mode unless there is a disconnection between levels i and i+1. Level disconnection can be defined to occur when there is a total disconnection (i.e. an absence of any route) between two levels, or just functional disconnection between two levels (e.g. the average or maximum routine length between two levels exceeds a fixed 'performance degree').

2: coordinators at level i enter a fail-safe mode only when, in the decentralized mode, a physical or functional disconnection is detected within the level.

Basic features of an oligarchical manufacturing system are reliability, fast response and low system complexity. Reliability is achieved because, in case of failure of a coordinator **x** element of **L[i]**, any coordinator **y** element of **L[i-1]** directly connected to **x** can still send its request to another coordinator **z** element of **L[i]** either directly or via some **w** element of **L[i-1]**; recall the coordinators in a given level can communicate with each other. Similarly, in case of direct line failure between **y** element of **L[i-1]**, **xeL[i]**, **y** can still send its request to level **i**. If there is a disconnection between the two levels **i** and **i+1**, the coordinators in **L[i]** cannot send their request to level **i+1**. In this case, they switch to a distributed mode of operation

which allows them to resolve any request. Thus the system will be operational, albeit at a slower pace. Therefore depending on the level and the mode, fail-safe or fail-soft, failure tolerant operation of the system can be achieved; the former at the lowest level of coordination, the latter at higher ones.

Since the number of levels in an oligarchical system can be kept small [KREI82], the response time for a request resolution in absence of failure is comparable to the response time that would be experienced in a centralized system. In the case of serious failures, the response time is never greater (and usually much less) than in fully distributed systems; this is because the number of elements in a level (involved in the distributed resolution) never exceeds the number of elements (involved in the resolution in a fully distributed system).

Independently of their level, all coordinators are functionally equivalent in that they execute the same basic algorithm (only on different data); thus they can be physically identical. Furthermore, by controlling the ratio

$$\text{alpha} = \frac{|L[i]|}{|L[i+1]|}$$

between successive levels, the total number of coordinators can be kept very small while still maintaining all the other features. This keeps system complexity low [KRIE82].

In [PICH83], it was shown how to implement controllers and coordinators in an oligarchical system when applied to a prototype assembly line. There are several factors that must be consi-

dered when implementing an oligarchical control mechanism:

1: to increase system performance, the partitioning of the elements should be done according to their level of interaction; elements whose cooperation is often required should be assigned to the same coordinator (as for a vision system and a robot manipulator); accordingly, the partitioning of coordinators should reflect the degree of interaction between the elements within its view.

2: the overall system reliability can be greatly enhanced by connecting each system element to more than one coordinator at the next level at the cost of increasing the complexity of the coordinators and the number of connection links.

An oligarchical control system integrates features of centralized, distributed and hierarchical strategies allowing the design of control mechanisms that are highly reliable, have fast response time, and can be easily upgraded, modified and expanded. Furthermore, because of the inherent modularity of an oligarchical system, the design process is simplified, as well as in the testing, verification and fine tuning of the resulting system.

3.4 Summary

In this chapter, it was shown that the physical realization of CIM is the Flexible Manufacturing System (FMS). A FMS is a cost effective flexible system for manufacturing a variety of similar or different products. A FMS consists of an Automated Manufacturing Systems (AMS) and a Flexible Assembly System (FAS)

linked by an Automated Transfer System (ATS). The subsystem elements of a FMS are Flexible Manufacturing Cells (FMC).

A AMS is a FMS subsystem that transforms raw materials into finished parts. A AMS consists of Automated Machining Cells (AMC linked by a Materials Handling System (MHS). A AMC performs machining activities that shape materials and give them a geometric quality [OWEN84].

A FAS is a FMS subsystem in which assembly activities are performed to transforms parts and components into finished assembled products. A FAS consists of Flexible Assembly Cells (FAC) linked by a Product Transfer System (PTS). A FAC performs assembly activities that mate parts and components into products that acquire a functional quality [OWEN84].

In order to make the FMS a feasible reality, manufacturing processes must be partitionable into sequences of activities and mapped onto a suitable set of FMC's for execution. The partitioning of these steps must be such that efficient use of the resources is achieved, and to predict, relieve, and possibly avoid potential bottlenecks in the system, with acceptable costs.

In addition, reconfiguration (change-over) of the whole FMS should not incur large design costs. Timely and cost effective change-over to a new product requires the manufacturing engineer to execute the following steps:

1: Specify the production steps in terms of activities executable by Flexible Manufacturing Cells (FMC's);

2: Reprogram, and if needed, reorganize the FMC's to execute the required activities;

3: Reconfigure the transfer system to execute the various production steps in the required sequence;

4: Develop the computer control strategy needed to integrate and coordinate the overall system.

Proper coordination of activities performed by FMC's must be achieved for effective flexible manufacturing. This also implies the proper and timely handling of error situations starting from the lowest level controllers right up to the top level system coordinator(s). The proposed oligarchical organization scheme offers a usable methodology to identify and specify total system control and coordination of activities in the FMS and to handle failures in a tolerant system. It permits different organization alternatives to be investigated in terms of the needed interaction of activities performed by different FMC's in the FMS.

There are few tools available to aid the manufacturing engineer in performing the above steps. Therefore, to provide flexible manufacturing at acceptable costs particularly in the cases of small and medium production runs, it is important to develop easy to use design tools that allow quick and orderly production reconfiguration, retooling and reprogramming.

Each each Flexible Manufacturing Cell must be programmable. To permit programmability, the elements of and FMC must also be well integrated. A major aspect that must be resolved is the reprogramming of reorganized FMC's with different or new elements

from different sources, to perform new activities in the manufacture of new products. A general scheme is needed to help in automating the process of FMC programming. In the next chapter, the problem of integration of elements into a programmable FMC will be addressed.

CHAPTER 4

Flexible Manufacturing Cell Design

4.1 Introduction

In this chapter a design scheme for a Flexible Manufacturing Cell (FMC), a computer controlled manufacturing element able to perform manufacturing activities for a wide variety of products, is proposed. Since this approach applies both to the design of AMC's and FAC's, it is presented in only terms of FAC's.

A Flexible Assembly System generally consists of a number of robot manipulators based FAC's whose actions are coordinated such that they execute assembly activities to assemble a product from its components. The FAC's are fed by part-feeders and linked together by a Product Transfer System (PTS).

A FAC is a flexible computer controlled assembly subsystem which specifically performs assembly activities by transforming parts and components into finished assembled products. In a large FAS, each FAC may perform partial assembly of the final product, and complete assembly is achieved when all assembly activities have been successfully executed [OWEN84][SPEC83].

The elements of a FAC for flexible assembly are first presented to identify and classify the range of functional, local control and coordination, and performance requirements such a FAC must satisfy. It is assumed that a FAC contains robot manipula-

tors, sensory systems, parts feeders or presenters and product transfer systems and other elements are considered for flexible assembly. Second, by showing that a FAC is the mechanical equivalent to a computer, a FAC organization can be defined to integrate the elements of a FAC, their controllers, interfaces, and their programming aspects. Furthermore, based on this equivalence, FAC programming is in terms of a proposed Cell Language.

4.2 Elements of Flexible Assembly Cells

In this section, the types of elements found in FAC's are presented to identify their complexity and variety, and the need for a methodology and organization that allows their integration into a single working FAC.

A basic FAC is composed of the following elements: a robot manipulator(s), a sensory system, parts feeders, fixtures, a central controlling processor, a local database, and communication links to other FAS elements (see Figure 4-1).

The robot manipulator consists of many arm segments and joints, and a power unit. Robot manipulators can be fitted with specialized end effectors needed to perform different types of assembly tasks. The sensory systems may consist of a vision system, tactile sensors, positioning sensors, sound sensors, and proximity sensors. To execute the required tasks by performing positioning transformations, a central controller is needed. This processor executes a set of programs needed to gather the sensory data, recognize objects, calculate joint positioning parameters,

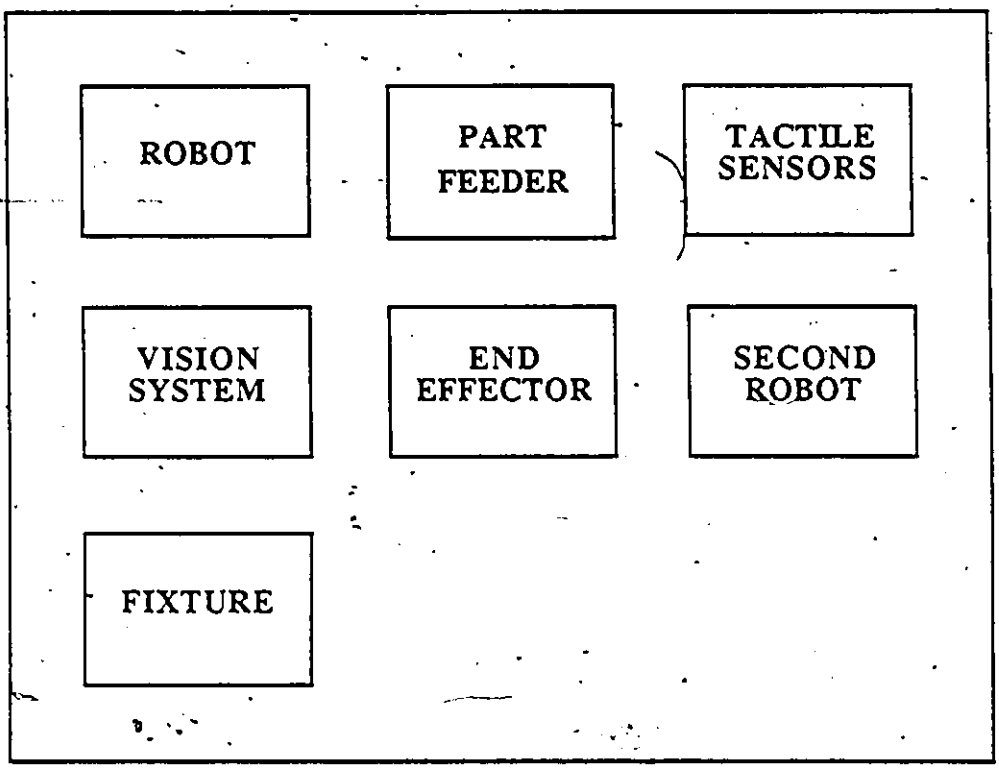


Figure 4-1. FAC Consisting of Various Elements.

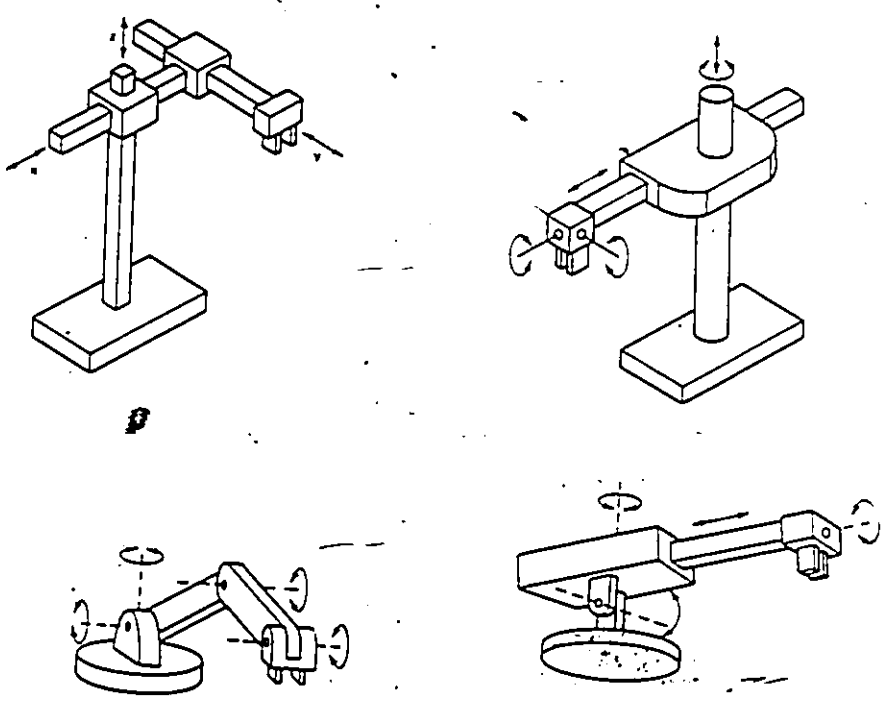


Figure 4-2. Various Robot Manipulator Configurations.

etc. The local database is needed to keep on-line all the required information about parts and products (models), possibly a set of programs that implement different algorithms required for controlling the local resources, and to acquire data from the assembly process that would be needed for archiving and inventory purposes.

4.2.1 Robot Manipulators

An industrial robot manipulator is mainly a movement oriented device, and is used to move the product to the tools, move tools to the product, or add components to an unfinished product. What differentiates this device from any previously conceived tool or device is that it is able to manipulate geometrically different objects and functionally different tools (pincers, drills, arch welding guns, etc.) under external control to perform manufacturing tasks.

The mechanical structure, the end effector or end tool found on the end of the robot arm, the movement control mechanism, and power source are the main components of these devices. There exists a wide variety of manipulators with different kinds and combinations of mechanical structure, end effector, movement control, and powering units. In this section, the different characteristics of robot manipulators are classified according to mechanical structure, end effector, movement control mechanism, and power unit source used.

4.2.1.1 Mechanical Configuration

The mechanical configuration of industrial robots generally consists of a central pedestal or frame with mechanical linkages. Movement is done by actuating the linkages and joints actuators, usually pneumatic, hydraulic, or electric. The manipulators may consists of many arm segments, joints, and special end tools (specific to a task), with many axes. The number of axes is usually referred to as the number of degrees of freedom, and gives an indication about the manipulator's flexibility, capability, and versatility. The actuators may be directly connected to the mechanical links or joints or may drive the mechanical elements indirectly through gears, chains, wires and pulleys, or ball screws. There is a wide variety of mechanical arrangements. Most available manipulators can be classified according to the type of coordinate system in which they work (see Figure 4-2).

Cartesian System Manipulators are controlled to move in the cartesian coordinate system, with at least one limb controlled in the x-, y-, and z- axis. The end unit is connected to the last limb and could have other limbs for flexibility. The work 'envelope', the space in which the robot can work, is a box.

Cylindrical System Manipulators are of the most common configuration. A horizontal limb or arm is mounted in a vertical column, usually fixed to a rotating base. The arm moves in and out, the carriage moves up and down on a vertical column with both upper components rotating as a single element on the base. The work 'envelope' is a portion of a cylinder.

Spherical System Manipulators have similar mechanics to the turret of a armored tank. The arm moves in and out, pivots vertically, and rotates horizontally about the base. The work 'envelope' is a portion of a sphere.

Articulated (Jointed-Spherical) System Manipulators have a forearm connected to the base through a second upper arm. At each arm elbow there is a joint usually permitting movement in one plane. Thus an upper arm and forearm move in a vertical plane through a trunk, connected to the base. Since there is an elbow between the forearm and upper arm, and there is a shoulder joint between the upper arm and the trunk, the work 'envelope' is a larger portion of a sphere, or an almost complete sphere.

4.2.1.2 End Effectors

An end effector is part of a wrist unit which can have a number of additional axes of rotation. The usual movements permitted by additional axes in such units are: roll, pitch and yaw. The roll movement is the motion of the end effector in a plane perpendicular to the end of the manipulator. The pitch movement permits rotation in a vertical plane through the arm. Lastly, yaw permits rotation in a horizontal plane through the arm. Other movements are allowed by using additional axes that could, for example, have 'fingers' and give increased versatility.

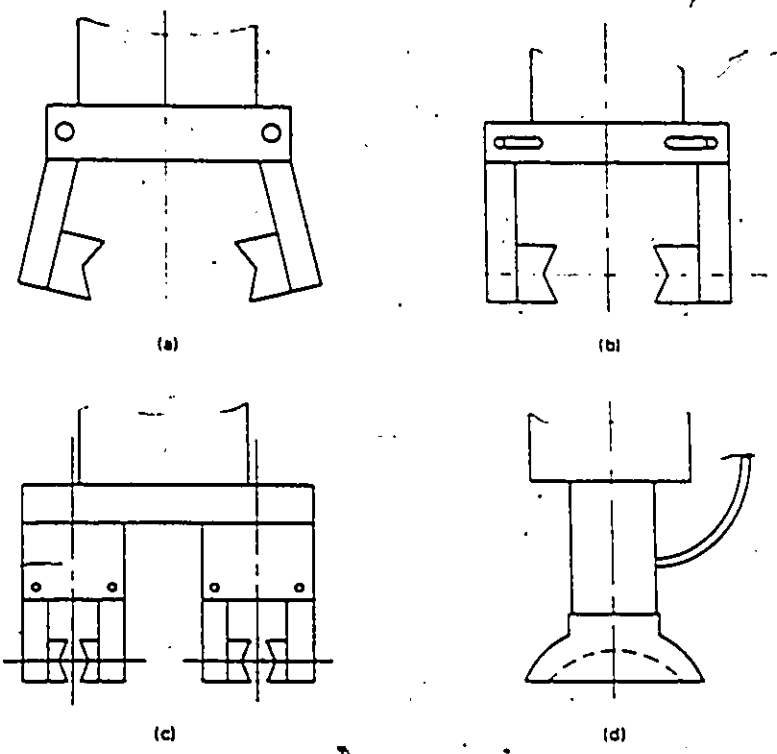
The end effector defines the set of coordinates for which the calculations are made to have the robot move it to the proper destination coordinates. Computations are made [PAUL81] for each

joint of an end effector as well as for each joint of a robot manipulator. Joint positions are translated into the needed actuator control values to bring the end effector to the destination.

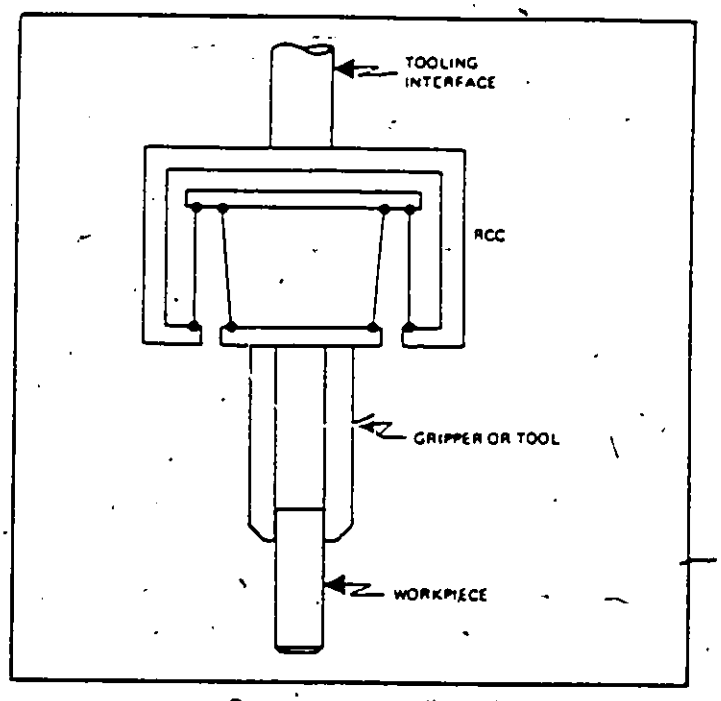
The end effector may itself span in complexity from a hook to pick up objects, to a highly advanced multi-fingered hand. Such hands allow extremely sophisticated operations to be performed where meticulous manipulation operations are needed such as in fastening. The are generally two types of end effector tools: grippers, used to hold workparts or other tools (mechanical grippers, suction cups, magnetized grippers, hooks, scoops); and function tools that are fastened directly to the robot wrist (spot welding, arc welding, and spay painting guns, drilling spindles, routers, grinders, brushes, torches).

4.2.1.3 Movement Control Types

Movement is governed by the robot manipulator's controller. The controller initiates and terminates manipulator motions, stores position and sequence data, and interfaces with the 'outside' world. There are many types of controllers found in robot manipulators. The range of controllers span from simple limit switch-based controllers and simple pneumatic logic systems, to advanced minicomputers. They are housed either in the manipulator or in separate cabinets, usually close by, or, in cases where there is a damaging environment, at a remote location.



Sample gripper designs: (a) pivot action gripper; (b) slide action gripper; (c) double gripper—pivot action mechanism; (d) vacuum-operated hand.



Remote-center compliant wrist.

Figure 4-3. End Effectors

The types of movement control exercised by the controllers can be classified based on the presence of sensors, and their level of sophistication.

A robot manipulator system that does not have any sensors is referred to as open loop controlled. All open-loop controlled systems correspond to table look-up systems, where the tables contain data that represents a sequence of steps required to execute the preplanned activities. The preparation of the tables is often done by precalculation, or by 'teaching', i.e. hand holding the robot through the different steps and recording the sequence of positions on tape or in memory for later playback. Some open loop controllers execute a set of pseudo-instructions determined by the programmer. This language allows the programmer to specify textually the sequence of operations to be done by the robot manipulator. Since no feedback is available, the sequence of operations just performed cannot be checked, nor can the robot manipulator react in the face of unforseen disturbances.

Advantages:

1: programming by 'teaching' is a fast and easy technique;
2: this type of manipulator is usually lower in cost;
3: this type of manipulator often has a higher load capacity (using hydraulic actuators) and has a wider working range;

Disadvantages:

1: 'teaching' has limited flexibility in terms of programming and positioning capabilities;

2: programmed positions cannot be easily modified during operation;

3: cannot be used in assembly applications;

4: this type of manipulator has low repeatability, i.e. the ability to make the same movement accurately with limited tolerance for several tries, and is a major drawback of this type of manipulator since slight changes can occur after a number of runs producing unacceptable misalignment;

To overcome wasteful operation in the absence of parts, simple presence sensors (switches) can be added. Typical applications for this type of manipulator are in parts handling and tool handling tasks (parts transfer) where consistent operation but not high repeatability is needed. Open loop controlled manipulators are used in similar applications such as spray painting, arch welding, polishing, and grinding, but not in assembly applications. Open loop controlled manipulators are often referred to as end-point, pick-and-place, bang-bang, or limited-sequence manipulators.

Closed loop controlled robots employ sophisticated feedback devices and monitor the appropriate variables until they reach a point at which the controller is instructed to stop actuating the robot arm or tool. Manipulators of this type have greater flexibility and are much more expensive, although less reliable (in some cases) than simpler machines because of their complexity. The general features of closed loop controlled manipulators are:

1: various manipulator members can be commanded to stop anywhere

(within their resolution) within their limits of travel;

2: velocity, acceleration and deceleration of arm members on various axes can be controlled;

3: the accuracy of operation is varied by controlling the magnitude of the relevant error signals;

4: programming is done as a design task during the development stage and the control is by program; the level of control is directly dependent on the complexity of the task to be done and the complexity of the program controlling the actions; sensors also contribute in the determination of the processing requirements to perform the tasks;

5: multi-program operation is facilitated and by using mini- or microcomputer based controllers, alternative actions can be taken within a program.

The different kinds of closed loop controllers can also be defined as being either point-to-point, or continuous-path.

A. Point-to-Point Closed Loop Controlled Manipulators

In point-to-point closed loop control, a destination position is fed to the robot actuators, movement is performed, and its position is then verified to determine if proper destination has been reached; if not, a correction is made. For complex paths, path is subdivided into intermediate segments and when the intermediate destination has been properly reached, the next position is calculated relative to the present position with refinements done at each step. The robot moves along segments to its final position, and corrections to the movements are done at each point

in the trajectory with no control of the path between the points reached.

Point-to-point closed loop controlled robot manipulators are used in any industrial applications which do not require exact positioning and are mostly used for small parts and tools manipulation. These systems have a high flexibility of operation, are usually found in the lower end of the scale in load capacity but have a broader working range. The actuators are usually mechanical devices with medium resolution positioning control.

B. Continuous-Path Closed Loop Controlled Manipulators

Continuous-path closed loop controlled manipulators are robots in which the path followed to reach the end point is achieved in a smooth continuous movement. The trajectory followed by the robot is computed and controlled in such a way that the robot moves in a continuous motion to the proper destination. The control system uses sensors to continuously track or check the progress of the robot as it moves toward the destination. Thus continuous adjustment can be performed without impeding the progress of the robot. More complex hardware, including advanced sensors, and advanced software development techniques are needed to achieve high operational flexibility.

In the case of continuous-path closed loop controlled robot manipulators, data is sampled periodically instead of at predetermined points in space; Thus large and fast memory storage is required such as mass storage computer disks to retain a considerable amount of information.

These robot manipulators are smaller in size and weight with high operation speeds but low loading capacity. These robots are usually found in applications such as parts manipulation for simple assembly where fine movement is needed or in more advanced parts transferring tasks as in moving conveyor systems.

4.2.1.4 Power Units

The power or energy source required by robots to activate its actuators is usually supplied from hydraulic, pneumatic, or electric sources. The supply is often found as an integral part of the manipulator, but can be a remote unit. For example a hydraulic power system generally comprises of an electrically driven pump, filter, reservoir and heat exchangers, with an petroleum-based fluid to transmit the power to the actuators. Hydraulic drives allow robots to have heavier payloads but usually at a cost of resolution. Pneumatic drives are the most widely form of power unit found, usually in medium payload situations. In pneumatic systems, compressed air is used instead of liquid. Electrical sources are now becoming the mainstay of robot power units for joint actuator motors because of their high controllability and reliability, although they often are found at the lower end of the payload scale. Power drive units are often found linked to the joints by levers, gears and steel rods, and by string and pulley.

4.2.2 Sensory Systems

In closed loop control, the use of sensors is required to provide the controller with the data and error information needed

to execute tasks. By providing the controller with positioning, vision, proximity, and tactile information, advanced tasks can be performed even with elementary manipulators.

Program control can use these data for different purposes from trajectory or path planning to object recognition. The complexity of the processing for some purposes puts a demand on the local computing power since the robot must react to the environment, often in a real-time fashion. [PETR86] identifies four major types of sensors: positional, proximity, machine vision, and tactile.

4.2.2.1 Positional Sensors

Positional sensors are used to determine the proper position of the robot manipulator. Positional data is required to determine the exact location of an object. In many applications, the information is required to find the position of the end effector of a robot manipulator. Positional sensors vary in complexity from simple switches actuated by the manipulator arm to advanced machine vision systems.

Robot internal positional sensors, called proprioceptors, are used on robots to measure the position, direction, and velocity of the robot joints. The devices used are potentiometers, optical shaft encoders, and resolvers or velocity sensors. Potentiometers measure the position of a robot joint using an analogue technique. Optical shaft encoders are more accurate digital devices used to measure position. There are two basic types of optical

shaft encoders: absolute, where the unique angular position is read optically as a unique (Gray) code on a circular disc; incremental, where a glass disc is imprinted with one or two circular rows of slots of equal size and spacing, and a reference slot. Direction of movement is also detected. The sensed data are either analogue or digital, although today almost all applications convert analogue data to digital data for ease of processing by computers.

4.2.2.2 Proximity Sensors

Proximity sensors are used to determine object absence or presence of surfaces, without actually contacting a surface [SNYD85]. Proximity sensors provide collision avoidance and information about approach conditions to permit controlled movement of a gripper such as deceleration or maneuvering.

Optical proximity sensors in the simplest form can be used to provided a simple binary signal indicating object presence. Each object requires separate calibration because the sensitivity of the sensor depends on object surface reflectivity. Reflectivity can be used to determine distances by projecting a calibrated light source onto the objects. This approach is used by optical ranging sensors. Fiber optics and signal processing techniques are increasingly used in the implementation of optical sensors in ranging situations because they provide improved performance.

Other techniques used to measure distance include triangulation. In this technique a plane of light is projected at a known angle and position onto an object and a light sensor is put at a

known distance from the light source. The observation angle is measured and from this the distance to the object is computed. A similar technique called light stripping uses a single light stripe and a CCD sensor put directly in the robot hand to measure distance; this is often referred to as a hand-eye configuration.

4.2.2.3 Machine Vision

Machine vision is a key sensor in modern robotic technology because of its versatility and the amount of information collected in one image sample. Their continuing decrease in price and increasing availability make this type of sensor very attractive in adapting to a FAC. Machine or robotic vision systems are used not only to obtain information about the objects and the scene where the robot is operating but can also be used as robot position sensors. They can be used to search and recognize objects, to determine the position and orientation of the objects, to perform robot path planning and tracking, and to determine the status of a task.

Robotic vision is divided into several functional layers: image acquisition, segmentation, parametric extraction, pattern recognition, and scene interpretation.

The lowest layer is image acquisition where light patterns are encoded usually in levels of gray. Some image restoration and enhancement can be done at this level, but focusing, aperture and viewpoint setting are the main functions performed. Image quality and accuracy depend on lighting conditions, depth of field, sensor sensitivity (pixel word size, number of quantization le-

vels, signal-to-noise ratio), scene and object reflectance, and the way the image is scanned. Image sensors are usually based on electron beam scanning as found in vidicon tubes, on 2-dimensional Charge-Coupled Devices (CCD's), on laser scanners and light source scanning called structured light. Structured light techniques allow both position and object image acquisition to be done using the same sensor. Other approaches imply the use of radar systems and illumination-generated coded events lighting.

In image segmentation, the scene image is broken up into components which correspond to semantically distinct objects in the scene. Segmentation partitions the image into relevant regions or objects. Meaningful partitioning results in the labeling of component regions of the image. Basic algorithms used to break up the image are edge finding and region clustering or growing.

In parameter extraction, objects which have been isolated by segmentation are specified according to position, shape, color and texture. Parameter extraction improves segmentation (by closing holes), separates and connects objects and help in the selection of objects based topological or shape features. In binary images, operations done at this level include pixel inversions, logical operations between two images, neighborhood operations such as dilation, erosion, contour finding, propagation, and skeleton extraction. Physical information can also be extracted such as the area of an object, its dimensions, its shape, its moment (center of gravity) and invariant moments, number of holes in the object, etc.

Pattern recognition is used to find features that are effective in discriminating between pattern classes. By recognizing patterns, the reduction of pattern data in stages is done and it is often viewed as an information reduction process [ENCY83]. Pattern recognition is done by the processes of feature extraction and evaluation, and selection. In feature extraction and evaluation, key features are found that permit the generation or reconstruction of the original patterns. In feature selection, features are selected that, with great care, characterize patterns. Pattern recognition may require greater computing power to perform recognition algorithms in a timely fashion. Many approaches are taken to reduce this requirements by reducing the number of features without adversely affecting error performance. There are several types of pattern classification approaches: parametric, non-parametric, and structural.

Parametric pattern classification is an approach based on a technique where pattern categories are known a priori to be characterized by a set of parameters for some classification tasks. This type of approach requires the definition of the discriminant function by a class of probability densities defined by a relatively small number of parameters. Pattern classes are usually assumed to arise from multivariate Gaussian distribution where the parameters are the mean and the covariants [BOW84].

In non-parametric pattern recognition, no assumptions are made about the characterizing parameters in many other classifications. Some parameterized discriminant functions (such as li-

near, non-linear, and piecewise linear functions) are used in non-parametric methods, but no conventional form of the distribution is assumed [BOW84]. Classification procedures also include stochastic approximation and potential function methods, clustering procedures, density estimation methods, nearest neighbor classification rules, statistically equivalent blocks, and discrete variable methods [PETR86].

In structural pattern recognition, patterns are defined in terms of primitives and relations among the primitives in order to describe pattern structure explicitly. In vision, primitives are simple objects and relations are spatial. The basic idea is to describe complex patterns in terms of simpler patterns. A formal language theoretic approach is employed to represent pattern structure in terms of syntax rules, and classes in terms of grammars and their associated languages. An observed pattern is assigned to the class whose grammar allows a successful parsing. This method is used to reduce the number of features required for complex patterns and so simplify object representation.

The description of a set of objects in terms of their identities, position, orientation, juxtaposition, structural relationship and semantic associations in a consistent way is known as scene interpretation. Advanced processing techniques are used to determine object features to permit the proper operation to be performed in controlling a FAC element such as a robot manipulator in robot trajectory planning.

4.2.2.4 Tactile Sensors

Tactile (or touch) sensing is one of the most important sensing methods for assembly applications. There are two aspects of tactile sensing: cutaneous sensing concerned with the textural patterns; and kinesthetic sensing concerned with the determination of forces and moments. Highly accurate positional information can be collected by tactile sensors (higher than in many current machine vision systems), and they also allow to perceive objects from more than one angle at once. This type of sensor is needed for systems which require highly accurate positioning of a robot end effector such as required in assembly. This leads to 'smart' end effectors complementing machine vision sensors. Tactile sensors detect: 1) presence of objects or parts; 2) part shape, location, and orientation; 3) contact area pressure and pressure distribution; 4) force magnitude, plane, and direction. The components of tactile sensors are a touch surface, a transducer medium to convert local forces and moments to electrical signals, structure, and control/interface [SNYD85].

4.2.3 Fixtures

Fixtures are devices used to hold objects in a known and fixed position during manufacturing steps. They are often used to overcome object manipulation shortcomings in robot manipulators. They permit the use of a single manipulator in tasks requiring that the object be held in a specific position while useful operations are performed on it. Fixtures can be simple static devices onto which an object is attached and then released, or

they can be more complex actively controlled devices by means of which the object held can be moved into more appropriate positions during work. In many cases (especially in assembly), major components of a product are modified in their design to act also as fixtures. This helps decrease manufacturing costs since it avoids the use of specialized fixtures.

4.2.3 Part Feeders

In any assembly line a method of presenting the parts is required to have continuous production. Parts presentation is a difficult problem because of the complexity and variety of component physical geometries. The parts feeders may orient the components either in a specific known position or in a general position where the details of the position will be determined at the time it is taken by the grasping device. In practice, a new solution to handling must often be developed for each new component in assembly, as is the case when bins are used. In simpler robot manipulator systems, components to be used in assembly of the product must be positioned at the FAC with high accuracy of location, with great consistency on every cycle, and usually at high rates of speed [GRO08Ø].

Also, problems such as defective components or parts must be properly addressed. In flexible assembly, components must be inspected in order to avoid jamming or complete defective products and possible assembly line stoppages. Sophisticated parts feeders could be involved in the inspection of the components prior to being used for assembly. There are generally two broad

types of parts feeders: dedicated or special, and general purpose.

4.2.4.1 Dedicated or Special Purpose Parts Feeders

Dedicated or special purpose parts feeders are used to present parts to the FAC in a completely prespecified and unchanging way. Dedicated parts feeders are designed in such a way as to reduce the amount of processing done by the work station and thereby aid in increasing the performance of the FAC, and permit the use of less sophisticated robot manipulators and existing devices in the FAC. The sensory system requirements for the station are also reduced in complexity to help performance and reduce the overall control complexity.

This type of parts feeder is meaningful under the conditions of large production runs where a high investment cost can be justified. These parts feeders tend to be specifically geared to a very narrow set of applications (and often only one), are custom designed and are usually very expensive to make or develop. For example, in Printed Circuit Board (PCB) manufacturing, there exist special purpose machines that prepare component shaped like cylinders, called axial components, by putting them in a feeding ribbon in the required order of insertion that will be followed during assembly. The investment made for the custom feeder is amortized over the life of the part feeder and the size of the production run. Dedicated parts feeders are not feasible in smaller production runs because of the overhead in designing them and their costs, and because they do not contribute to the

overall flexibility of automated FAC's.

4.2.4.2 General Purpose Parts Feeders

In small production run FAS's, general purpose parts feeders are required because special purpose part feeders are too costly to develop and would not make this type of FAS's feasible. General purpose parts feeders used with robot manipulator-based FAC's, are often coupled to advanced vision systems. The vision systems are needed to determine the orientation and position of objects. In order to recognize and grab the parts properly for the assembly process increased processing power is needed with this type of feeder. This increases the complexity of the FAC.

One method used to reduce the processing requirements and save processing time is to avoid losing the orientation information acquired by an object at one FAC by placing the object in a known and desired position. The object is then passed on to the next station by a Product Transfer System (PTS) without losing its orientation, and presents it to the next FAC in the expected orientation. Therefore, the PTS (see Chapter 2) can be viewed as general part feeders, but at a higher level.

General purpose parts feeders often take the form of bins where the parts are found with arbitrary orientation and size. The bins also act a product transfer buffers between FAC's helping in the decoupling of the assembly tasks at each FAC.

4.3 Current Programming Approaches

In robot manipulators, both control and programming are inti-

mately connected and cannot be considered separately. There are several levels of robot programming: physical level, computer level, guiding level, robot level, and task level. Control programming falls under three categories: guiding, motion or robot-level programming, and goal-oriented or task-level programming.

4.3.1 Physical Level

Physical level programming is the most primitive level of robot programming. There is no actual programming in the conventional sense using computers, but rather a robot control sequence is setup by an operator (programmer) by adjusting or fixing limit switches, stoppers and the like at the correct places in the correct sequence. Only simple and limited sequences are possible.

4.3.2 Computer Level

Of major interest are the programming aspects that involve computers, and three main levels of programming can be identified: guiding, robot level, and task level [LOZA83].

4.3.2.1 Guiding

Guiding refers to the control of the individual actuators such that proper positioning and motion is done. The method involves manually moving the robot using a teaching pendant to each desired position and recording the internal joint coordinates corresponding to that position. At some of these positions, operations can be specified such as closing a gripper or activating a welding gun.

Guiding is open loop control and is done where a set of simple actions sequences are performed repeatedly and no sensing is done. This type of programming is the simplest to use and implement and does not need a general purpose computer nor does it require skilled personnel. Guiding was the most widespread method used for many years. This technique resembles that used in NC machine programming since the sequence of operations are played back much in the same manner. There is no execution sequence control since there are no loops, conditional statements, or computations possible. It is best suited for spot welding, spay painting, and simple materials handling. This type of control is not appropriate for flexible assembly since there is no sensory feedback upon which to react. The robots having this type of control are widely available commercially.

An improvement on simple guiding is extended guiding where some sensory information is fed back to the controller. Sensors used to determine the location of objects provide information so that robot motions can be made relative to the object's coordinate frame. Taught motions must therefore be specified relative to some coordinate frame that can be modified during execution. By having the robot move until a touch sensor on the end effector encounters an object, the coordinate frames can be determined. Alternately, a machine vision system may provide the coordinate frames. Some extended guiding systems provide additional but limited control structures such as conditional branching and simple procedures with no explicit computational support. Programs or guiding input from a teach-pendant can be specified off-

line using CAD systems that simulate the motions of the robot.

4.3.2.2 Robot Level

Robot level programming relates to the positioning and response of mechanical joints in performance of tasks. In robot level programming, or motion control, explicit commands determine each robot action; it assumes the availability of computing resources to calculate the various parameter values needed for the execution of a specific motion. These systems can either be open loop where no sensory information is available, or closed loop where sensors are used to guide the actions.

Motion control requires the use of computers and suitable programming languages with commands to access sensors and to specify robot motions. This type of programming is also called explicit programming and requires expert programmers in the design of sensory based motion strategies. With sensors, robots can cope with a greater degree of uncertainty in the position of external objects, thereby increasing their range of application.

Robot motion control for assembly requires the path of the end effector to be described. A robot can have, for example, all motions of the end effector to be straight lines with controlled acceleration and deceleration, or robot joints could be moved along a trajectory as a coordinated function of time [PAUL72].

In straight line motions, the motions require continual mathematical transformations between cartesian coordinates of the end effector to joint actuator coordinate in order to provide

adequate robot control. Therefore a consequence of this requirement is that motions must be generated as cartesian functions of time. Velocities and accelerations may be specified and the robot is designed to operate at the maximum resulting joint rates. But the robot rarely carries out motions requiring maximum joint rate, and where this does occur it is only for brief periods. This means that the robot is always running at less than optimum performance.

The coordinated function of time joint trajectory approach allows cartesian motion of the hand over small distances and predictable motions over large distances. The motion is described in terms of the same coordinated system in which the motion occurs. Hence the robot joints can be driven as a function of the slowest joint rather than as a function of time. This eases the controlled end effector motion to the point where it operates at the speed of regular point-to-point motions. The scheme described by Anderson and Paul [PAUL79] can be extended to include devices such as conveyors or presses or second arm coordinated with the first arm. This allows time delays to be eliminated [PAUL79], [RENA79]. This means that coordination signifies that the equivalent relative position of all joints is the same throughout the motion. This is accomplished by means of microprocessor-based software-driven controllers (implementing position-derivative) for each joint whose control signals are generated as a function of relative position of the slowest joint. When the slowest joint is driven at its maximum speed, coordination is maintained and time is minimum.

There are many robot level programming languages in existence today and they are well surveyed in [BONN82], [BONN83], [LOZA83], [PETR86].

4.3.2.3 Task Level

Task level programming refers to the control of the whole FAC required to perform a task or set of tasks in achieving a goal, such as assembly. In this sense every goal task may correspond to a sequence of motions, which in turn may represent a set of microsteps such as actuator activations, e.g. in the control of a joint.

Also known as goal oriented control or world modelling, task level programming requires that goals be specified for the position of objects rather than motions of the robot needed to achieve those goals. This is meant to be completely robot independent implying that positions or paths dependent on the robot geometry or kinematics are not specified by the user. Complete geometric models of the environment and the robot are required as input.

Task level programming languages have been developed to permit task specification in terms of operations on the objects being manipulated in the task. The sequence of robot motions needed to accomplish the insertion of a peg in a hole are not specified, rather, the high level task command would simply order the robot to INSERT PEG IN HOLE.

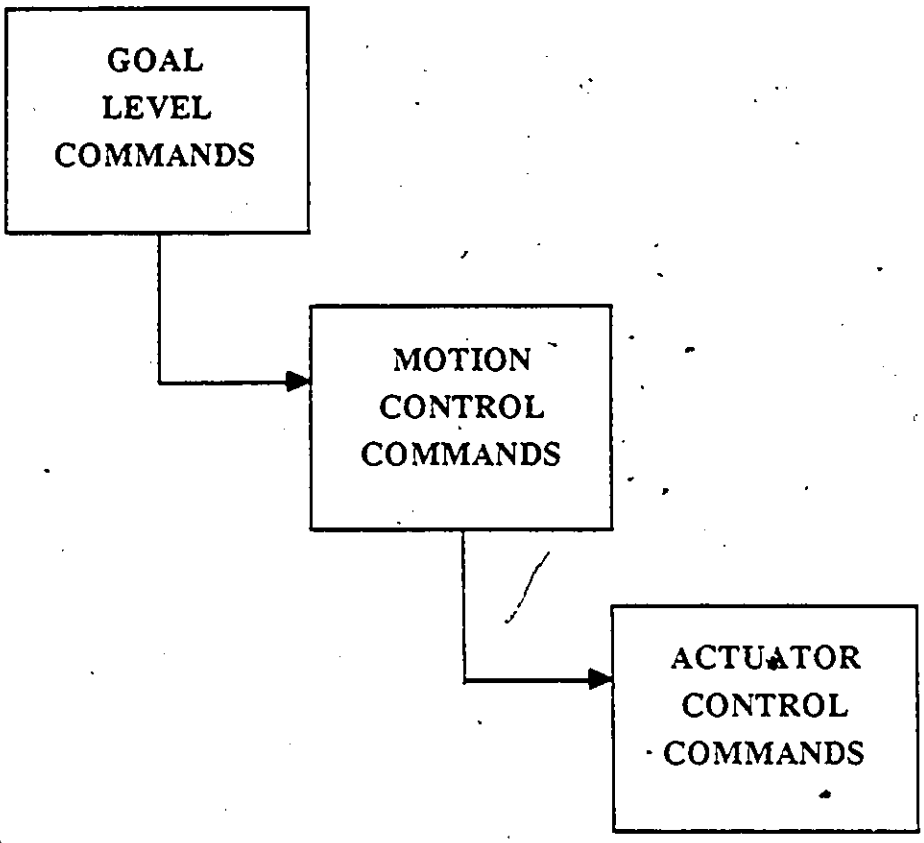


Figure 4-4. Levels of Computer Level Robot Programming

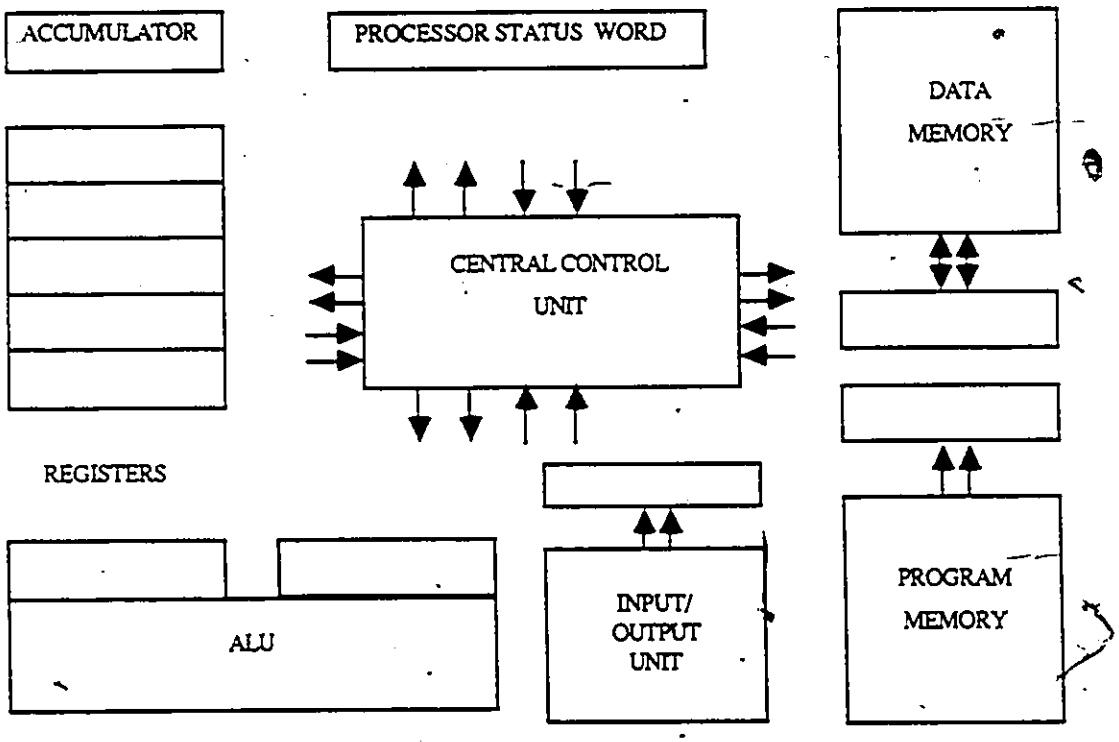


Figure 4-5. Central Processing Unit Elements.

To accomplish this kind of control a task programmer is needed to transform the task level specifications into robot level specifications. This requires that the task programmer have complete descriptions of the objects being manipulated, the task environment, and the final state to be reached at the completion of the task. The task programmer generates a program to achieve the desired final state when executed from the specified initial state. The task programmer's ability to synthesize programs that reliably achieve goals, depends on how well the program can use inherent capabilities such as compliant motion, and perform error checking. Thus a efficient use of sensors must be made in generating the robot level programs. The task programming aspect is another major research area where the complete synthesis of the task programs is seeked.

The three main phases of task level programming are: modelling, task specification, and robot level program synthesis. Even though there is much research to be done in goal oriented or task level programming, most of the robotic research being done in this area relate mostly to motion optimization and programming automation.

4.4 FAC: The Mechanical Equivalent of a Computer

In this section, the FAC is shown to be the mechanical equivalent of a computer in terms of its function, its elements, and its operation. Based on these equivalences, it is shown that an FAC instruction set, consisting of different instruction types with their different execution flow, can be defined and used to

program the FAC. Furthermore, based on this instruction set, a centralized control organization for an FAC is proposed to integrate elements into a coordinated, programmable, and flexible assembly cell.

4.4.1 Functional Equivalence

In assembly, objects acquire functional and structural qualities, and can be thought of as a process that reduces the surrounding entropy as parts are added. The mechanical assembly tasks performed by a FAC correspond to successive matings of components and parts resulting in a finished product or subassembly. Each mating operation is achieved by having the different elements of the FAC such as robot arms, fixtures, part feeders and various sensors (vision, tactile, positional, etc.) perform primitive operations in a well defined coordinated sequence. Assembly is therefore a coordinated sequence of well defined assembly steps to take components and parts and arrange them into a finished product.

In other words, the sequence of assembly steps are much like a sequence of well defined program steps in a computer whose purpose is to arrange data objects into a finished or desired format. By comparing the sequence of assembly operations with the sequence of machine language instructions in the execution of a program, the functional correspondences between a FAC and a computer can be observed. This means that a FAC functionally corresponds to a computer; components and parts in a FAC correspond to data in a computer.

4.4.2 Element Correspondence

Since the FAC has a single central control unit its elements will be associated with those of a simple Von Neumann architecture.

A conventional general purpose computer consists of the following main subsystems: Input/Output subsystem (I/O), Memory subsystems, and the Central Processing Unit (CPU). Such a computer executes programs as a sequence of machine language instructions (MLI's) that perform operations on data objects giving useful results. A MLI is further decomposed into a coordinated sequence of microinstructions that control the elements of a computer such that the overall system behavior results in the execution of the specific MLI.

The memory subsystem of a computer consists of main memory and secondary memory. In main memory, data and programs are stored. Mass memory is more of an archival storage area where data and programs are simply stored for later retrieval. In a FAC, components and parts used in the assembly correspond to stored data used in a computer. Preorganized components in an FAC are equivalent to a memory system; component selection is made by using its known position similar to addressing data in a computer. For unorganized components a sensory system must be used to find and identify objects, their position and orientation, just as associative memory is used in a computer to search and extract data from a data space. In a FAC the control sequence, the program, is stored separately as in a Harvard type computer

[KRIE8Ø]. There is no direct equivalence to secondary memory.

The CPU in a computer is the element that controls program execution and that manipulates data. A CPU usually consists of several subsystem elements as shown in Figure 4-5:

1: An Arithmetic-Logic Unit (ALU) where all low level data manipulations are done; the corresponding FAC element is the robot manipulator combined with the end effector; component mating operations done by the robot arm correspond to data manipulations performed by an ALU;

2: An accumulator where data can be temporarily stored during program execution; the corresponding FAC element is the assembly fixture;

3: the status of an instruction execution is held in a special register called the Processor Status Word (PSW). In a FAC, execution status is provided by a sensory system;

4: A central control unit from which all control and coordination of computer elements emanate. Control is achieved by decoding instructions and issuing control signals in a well defined sequence to the other subsystems and elements; the sequence of control signals are themselves microsteps or microinstructions that instruct the subsystem elements to perform low-level operations and the sum of low-level operations results in the desired effect as told by the instruction. The equivalent FAC element is the central coordinator which controls all other elements.

The I/O subsystem provides the computer with communication

channels to and from the external world. In a FAC, part feeders are input devices while output bins with various conveyor systems correspond to outputs.

4.4.3 Operational Equivalence

The FAC operational equivalence to a computer can be identified by considering the necessary operations needed in the execution of a FAC assembly instruction.

At the simplest level, a FAC receives assembly instructions that are executed 'blindly' without checks or decisions. All robot actions executed at this level can be expressed in terms of the following types of instructions:

MOVE instructions:	MOVE	<DEST><OBJECT>
ONE OPERAND instructions:	OPERATION	<OBJECT>
TWO OPERAND instructions:	OPERATION	<OBJECT-1><OBJECT-2>

where <OBJECT>, <OBJECT-1> and <OBJECT-2> identify the object to be moved by the MOVE instruction, or operated upon by the OPERATION instruction, and <DEST> defines where the object is to be moved.

In a computer, these types of instructions are used in the evaluation of expressions. It should be noted that in assembly, the data is always constant or fixed while in a computer, different results are obtained depending on the data.

The nature of component assembly is sequential with no inherent need for alternative execution paths. This is different with respect to a computer program where many decisions are involved

in the execution of an algorithm. General purpose programs are executed on different data and so computer programs make decisions on variable data.

In a FAC, if all goes well during an assembly process, there is no need for decision. But because of 'large' error potential during assembly, most FAC's must include a sensory system and the FAC must be provided with appropriate decision capability so that the execution process can be modified in case of error. In a FAC, decisions must be made with respect to improper or erroneous operation execution of the desired instruction. Moreover, an 'assembly' program deals only with identical and unchanging data (objects). Any change detected is considered a fault and must be handled as such. This means that every operation to be done on objects in the assembly of a product must be verified. This mode of instruction execution contrasts with that of a computer where checking operations are performed as separate instructions in the execution of a program.

This is what makes FAC program execution different to program execution in a general purpose computer. Errors such as missing components, faulty component, collisions, inaccurate positioning, etc., must be handled reliably to permit, as much as possible, continuous production in an FMS.

To properly control the execution flow in the case of error, the following enhancement to the CLI's must be made:

<INSTRUCTION><ERROR ACTIONS>

This modification is needed to allow the proper error information to be returned to the controller and to specify the specific alternate paths in case of error. Since each instruction consists of several microsteps executed by the elements in a FAC, and since a failure in the instruction can occur at any of its microsteps, each instruction must include a multi-branch checking function to establish exactly at which point the instruction failed, which microstep primitive failed, and to determine what recovery procedure must be executed to handle an error at this point. In this sense, each instruction of a FAC must be realized by a functional programming approach.

A concrete explanation of FAC functional programming approach can be shown by the following example. Since an instruction is a well defined sequence of microsteps, this instruction can be represented in the following way;

function INST(A1, A2, ... , Ai, ... , An): STATUS;

where each input parameter Aj to the function INST is an assembly primitive and the return value is status information indicating the function INST's execution status. This return value is used to determine the appropriate recovery action by decoding its value and branching to a specific recovery procedure. Recovery procedures in a FAC can take many forms: retry operation; postpone the present step and go on to the following step; perform a specific recovery operation sequence; perform an alternate operation; stop all work in the FAC and request for assistance; or sound an alarm; etc.

This functional representation corresponds to a computer programming language software function whose parameters are procedures and the return value is, let us say, an integer. The return value is then decoded to determine the next step in the program. This type of programming convention is often used in complex computer programs to indicate different problems in the execution of the function by encoding the function's return value appropriately. In this way the status of the operation can be verified by checking the return value of the function.

For example, a return value of zero often indicates that the function was executed properly and execution can go on normally to the next step; a non-zero value is used to indicate the type of problem met during its execution. A multi-branch conditional check (e.g. **case** statement in Pascal) on the returned value is often done to determine the type of error encountered. The program execution then follows the appropriate branch path where error handling steps are taken:

```
case INST( A1, A2, ... , Ai, ... , An ) of
{NON-ZERO function return value}
    1: Perform recovery process Pj;
    2: Perform recovery process Pk;
    .
    .
    .
    i: Perform recovery process Pm;
    .
    .
    .
    n: Perform recovery process Pq;
end; {case}
{ZERO function return value: perform the next step}
```

In a high level FAC, this type of processing, i.e. multiple branch conditional checking, must be performed for every assembly step in order to determine if the operation was successful and, if not, to perform the appropriate recovery procedure. Since the recovery processes must be specified for each microstep of an instruction, this approach is cumbersome and should be automated.

4.5 Modular FAC Architecture

In order to coordinate different FAC elements in the performance of an assembly process, meaningful control of the elements must be kept in such a way that the assembly activity can be executed in a well defined and ordered manner. Based on the shown equivalence between the FAC and a computer, the following centrally controlled or coordinated FAC architecture is proposed. It is based on Segmented Microprogramming, a technique that can be used to design special purpose VLSI processors [VAIL82],[PICH84]. It is shown to be modular in its design allowing independent FAC elements and their controllers to be integrated in a straight forward way into a single FAC.

4.5.1 Centrally Coordinated Modular Design of a FAC

FAC elements are usually not designed and implemented by users. Instead, FAC elements and their controllers are purchased as finished products from suppliers. Users are often restricted in the choice of elements that can be used because these elements must somehow be integrated into a single FAC to execute assembly activities.

This FAC design problem is often partially or temporarily avoided using 'compatible' hardware, usually from a single manufacturer. As a rule the main FAC element, often the robot manipulator, determines which other elements can be used with it in a FAC. This could potentially limit the freedom of the user in designing an FAC for a new application because element hardware from other sources may not be compatible. When adopting the 'family' approach, users are often forced to wait for the supplier to provide them with the needed elements.

The use of a 'family' of FAC elements from a single supplier or source may have the advantage of compatible elements. This is acceptable in single station (or work cell) systems or at most in systems consisting of a few FAC's. On the other hand, for small production run FAS's (FMS's) that manufacture a wide range of products in limited quantities, this approach can lead to strong dependence upon the element supplier and the supplier's schedules. Specific needs may not be met by the supplier and users may be forced to compromise and rethink their product manufacturing approach. The choice of a 'family' of elements often involves engineering and economic tradeoffs that may have little or nothing to do with the assembly activities to be performed. The use of out-dated or inappropriate 'family' elements may lead to restricted flexibility and restricted range of applications of the final FAC and many types of assembly activities may not be possible.

To avoid heavy dependence, FMS designers must be able to choose the best hardware available from other sources that can

meet the new requirements. True FMS realization can only be achieved if FMS designers are allowed this flexibility. Several 'incompatible' elements from potentially different sources must be made to integrate well to form a single functional and working FAC that executes the needed assembly activities. Also, integration of the needed elements must be made possible without incurring major costs in implementation and programming of the whole FAC for every product change-over. An FAC coordinator architecture must allow integration.

4.5.2 Coordinator Architecture

The coordinator architecture must satisfy the following criteria:

1: it must allow the straightforward integration of elements and their controllers from different sources;
2: it must accommodate a wide range of elements of varied sophistication necessary to the execution of the FAC activities;
3: it must keep the cost of integration low;
4: it must help designers use existing or available elements and avoid inventing new ones as much as possible, because of the cost of development of FAC elements (robots, sensor systems, etc.) is extremely high (at this time) for any manufacturer regardless of production runs sizes;
5: it must allow the FAC designer to use the autonomy of programming of the selected elements, since each element can be unique in the range and applications it can perform, and its

programming is likely to be based on specific characteristics;

6: it must permit the separation of coordination from direct element control to allow failures to be resolved or dealt with as a separate problem;

To make the elements work together as one, the chosen elements can best be integrated by a centralized controller-coordinator. A centralized coordinator architecture can conveniently interpret CLI's to issue element commands (microsteps) to the element controllers in the proper sequence and keep track of the progress of their execution and take appropriate actions upon the detection of failure situations.

To maintain modularity of the FAC, the central coordinator must be able to communicate with the FAC elements through a common or standardized interface and communication protocol. With an interface, elements execute received microinstructions or commands and return status and other information to the central coordinator. With this information the central coordinator can determine the next step in the execution of a FAC CLI program.

Each element is likely to have its own programmable controller. Every FAC microinstruction is implemented as an element subprogram. Upon receiving a coordinator command, the appropriate element subprogram is executed. Since an FAC CLI is a set of microinstructions executed in a specific sequence, and since each microinstruction is implemented as a separate element subprogram,

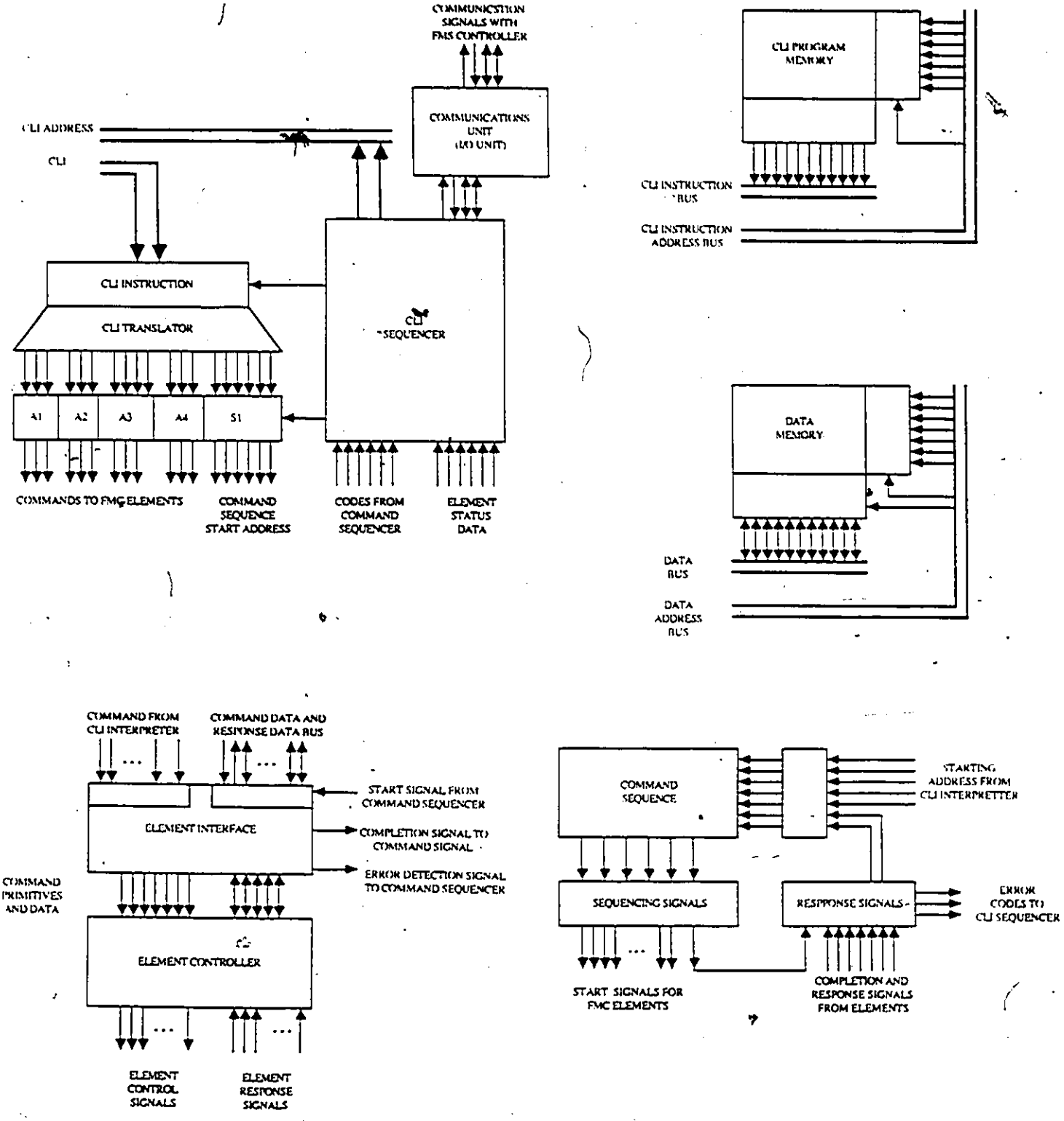


Figure 4-6. FAC Architecture

element programming can be done separately from the integration and coordination aspects. Integration is done through proper CLI translation and the standard interface, and coordination is achieved through the proper sequencing of microinstructions to the necessary FAC elements that implement the CLI. The proposed architecture is based on a segmented microprogrammed architecture for computer design [VAIL82], [PICH84]. A functional diagram of the centrally coordinated FAC architecture is shown in Figure 4-6.

This FAC architecture can execute a sequence of CLI's and consists of the following subelements: an CLI decoder or interpreter unit, a CLI sequence unit, a command sequence unit, an element interface control unit per element, and the FAC elements and their controllers.

The coordinator's function is to coordinate the execution of CL instructions of an assembly activity by interpreting CLI's, issuing appropriate element commands to the FAC element controllers in the needed sequence, analyzing element responses, to determine and take the next action. In this way, the FAC coordinator coordinates all the FAC elements in the completion of the assembly activity through the execution of Cell Level Instructions (CLI). To achieve this goal, the coordinator must perform several duties:

1: it must translate the CLI's into microinstructions (commands, subprograms calls) that each element understands;
2: it must transmit the microinstruction to the appropriate elements;

3: it must transmit the microinstructions in the proper sequence;

4: it must resolve failures, temporary or permanent, such that as much assembly as possible can take place and make error recovery possible;

5: it must communicate with the other FAC's in the FMS to provide for total system control and coordination.

Some of the attributes that favor the proposed centrally coordinated architecture are:

1: allows the separate programming of the FAC elements before integration since CLI's define a set of microinstructions which define the set of subprograms that must be implemented for every element;

2: allows the FAC designer to use the appropriate elements in the needed numbers to implement the needed assembly-activity;

3: redesign and setup costs are reduced since subprograms which implement microinstructions formerly used in the realization of other CLI's can be reused with little of no modification; this resembles the gains of group technology found in machining activities as well as those of segmented microprogramming [PICH84];

4: lower complexity of central coordinator;

5: reduces the need to develop in-house FAC elements;

6: lowers overall design cost;

Some of the drawbacks of the centrally controlled coordinator are:

1: the need to design an interface for each new element in the FAC;

2: the central coordinator potentially adds another layer of control hardware and software which may increase complexity;

A CL program consists of a sequence of CL instructions. The CL program implements the FAC activity. A FAC coordinator consisting of several subunits is able to execute CL programs by the following process:

1: the CLI sequencer fetches a CLI instruction of a CLI program;

2: the CLI is translated by a CLI interpreter into element commands sent to the element interface controllers and an execution sequence is selected in the command sequence controller;

3: the CLI sequencer initiates the command sequence controller;

4: the command sequence controller enables interface units in the required sequence to execute the element commands;

5: the interface units translate the element command into sequences of command primitives implementing it;

6: the element controllers instruct the element to perform the sequence of command primitives;

This process is analogous to the decoding of an instruction in a computer by a CPU into a sequence of microinstructions issued to subsystem microcontrollers in a computer. The microcontrollers in turn issue control signals (nanoinstructions) to

their subsystem elements in the execution of a microinstruction.

4.5.2.1 CLI Sequencer

The function of the CLI sequencer is to fetch CLI's from the stored sequence of CLI's in program storage, and send them to the CLI interpreter. It also extract needed execution data from the local data stores and makes it available during the CLI's execution. The CLI sequencer controls the execution of the CLI program and makes decisions about the executing activity. The CLI sequencer starts the execution of the CL instruction by initiating the sequence controller. It therefore acts as a CLI program monitor that deals with each CLI's execution progress and takes the appropriate action when 'abnormal' situations arise.

4.5.2.2 CLI Interpreter

The CLI interpreter translates the fetched instruction into element commands (subprogram calls or start addresses) for each element and sends them to the elements through their interface control unit. The CLI interpreter also sets up the command sequence controller so that the appropriate command sequence will be used.

4.5.2.3 Command Sequence Controller

The command sequence controller contains a set of different command sequences that coordinate the execution of the commands sent to the different elements. It sends start up signals to the interface units at the proper time and waits for responses from the interface concerning the element command being executed. If

the command sequence controller detects an erroneous response from an interface controller, it informs the CLI sequencer which decides on the next step in the assembly activity.

4.5.2.4 FAC Data Memory Unit

In the FAC data memory unit is a local data-base containing data required during the execution of CLI's. These data are accessed through an Interface unit, under the control of the command sequence controller and are sent to the appropriate element controllers via the element interface units.

4.5.2.5 FAC Program Memory Unit

In the FAC program memory unit is found the FAC CL program. The CL instructions are fetched by the CLI sequencer and brought to the CLI interpreter for translation.

4.5.2.6 Interface Units

The interface units accept element commands from the CLI interpreter. Upon the reception of the enable signal from the command sequence controller, the element command is translated into command primitives. Each command primitive is sent to the element controller. Information is returned to the command sequence controller describing the status of the command execution. The interface units are analogous to subsystem microcontrollers in a computer which generate a sequence of nanoinstructions.

4.5.2.7 Element Controllers

At the lowest control level of a FAC, each element is controlled by its own controller. Each command primitive is executed by the element controller by issuing sequences of low level control signals to the element's controlled devices. This process is similar to nanocontrollers in a computer that issue control signals to their subsystem elements in the execution of a nano-instruction.

The FAC element controllers accept the transmitted primitive commands and initiate their execution; feedback information on the progress or success of the primitive commands is returned to the interface unit. Execution information may also be sent directly to another FAC element controller under control of the command sequence unit in cases where this information, such as position data as found by a vision system and sent to the robot controller, is needed by elements to execute the primitive command.

4.6 FAC Programming

Given the FAC architecture proposed above, the problem of integrating the functional aspects of the FAC can be done through a matched suitable programming paradigm.

4.6.1 Cell Level Language

In line with the above, an assembly activity is a sequence of Cell Level Instructions (CLI's) and are executed by a FAC in a similar way a program consisting of Machine Language Instruc-

tions's (MLI's) is executed by a computer. The Cell Language corresponds to a task level language in the robotics literature [GROO83][HUTC84]. By defining a CLI set, FAC designers can 'program' the FAC's assembly activity at a high level early in the design of the FAC without being hampered by low level details. New CLI's could be created from new element commands. In time, a large and properly defined set of CLI's could be developed. These would then be used in implementing new FAC activities more quickly, further reducing FAC programming time, effort, and cost.

4.6.1.1 Cell Level Instruction Set and Format

To show that an assembly activity can be described in terms of CL instructions, the assembly of a toy doll's head is presented. The task at hand is the successful mating of the doll's eyes, nose, and mouth to the head. In this example, the mating is done by gluing the parts obtained from feeders to the head. The FAC elements also include a robot arm, a fixture, and an output bin. The robot arm can grab all the parts including a glue dispenser. The workspace is defined as the work envelope of the robot arm. The assembly of the head can be specified in the following set of task level commands:

```
PLACE   HEAD IN FIXTURE
PLACE   GLUE AT LEFT EYE POSITION
PLACE   GLUE AT RIGHT EYE POSITION
PLACE   GLUE AT NOSE POSITION
PLACE   GLUE AT MOUTH POSITION
PLACE   LEFT EYE AT LEFT EYE POSITION
PLACE   RIGHT EYE AT RIGHT EYE POSITION
PLACE   NOSE AT NOSE POSITION
PLACE   MOUTH AT MOUTH POSITION
PLACE   HEAD IN OUT BIN
```

To emphasize the machine language flavor of the CL instructions, the above assembly activity can be rewritten in a pseudo-machine language form, highlighting the operations and operands of the instructions, as:

```
MOVE        FIXTURE,              HEAD
PUT_GLUE    LEFT EYE POSITION
PUT_GLUE    RIGHT EYE POSITION
PUT_GLUE    NOSE POSITION
PUT_GLUE    MOUTH POSITION
MOVE        LEFT EYE POSITION,    LEFT EYE
MOVE        RIGHT EYE POSITION,   RIGHT EYE
MOVE        NOSE POSITION,        NOSE
MOVE        MOUTH POSITION,       MOUTH
MOVE        OUTBIN,               HEAD
```

The CL instruction can be divided into fields. The first field of the instruction defines the operation to be performed; the second and third fields define either operands (objects) or addresses (locations).

To allow for failure handling, the CL instruction format must include a failure actions field to define the selected set of corrective actions to be performed in the detection of failures.

OPCODE	OPERANDS	FAILURE ACTIONS FIELD

Cell Level Instruction Format

4.6.1.2 Cell Level Instruction Execution

These instructions must be translated into the appropriate sequence of primitive commands. To show this, the following instructions are be expanded as follows:

```
MOVE        FIXTURE,   HEAD

SENSORS:    CHECK      FIXTURE
SENSORS:    CHECK      HEAD PRESENCE
ROBOT:      MOVE       END EFFECTOR TO HEAD
ROBOT:      GRAB       HEAD WITH END EFFECTOR
SENSORS:    CHECK      FORCE OF HEAD GRAB.
ROBOT:      MOVE       END-EFFECTOR TO FIXTURE
ROBOT:      PUSH       STRAIGHT HEAD ONTO FIXTURE
SENSORS:    CHECK.     FORCE OF HEAD PUSH
ROBOT:      RELEASE    HEAD
SENSORS:    CHECK      HEAD IN FIXTURE

PUT_GLUE    NOSE POSITION,      NOSE

SENSORS:    CHECK          GLUE-DISPENSER
ROBOT:      MOVE           END-EFFECTOR TO G-D POS.
ROBOT:      GRAB           G-D WITH END-EFFECTOR
SENSORS:    CHECK          FORCE OF G-D GRAB
ROBOT:      MOVE           TO HEAD IN FIXTURE POS.
APPLICATOR: ACTIVATE       GLUE-DISPENSER
APPLICATOR: DEACTIVATE     GLUE-DISPENSER
ROBOT:      MOVE           END-EFFECTOR TO G-D POS.
ROBOT:      PUSH STRAIGHT  G-D ONTO G-D POSITION
SENSORS:    CHECK          FORCE OF G-D PUSH
ROBOT:      RELEASE        GLUE-DISPENSER
SENSORS:    CHECK          GLUE-DISPENSER
```

The set of element commands can be readily defined. Also, the command sequence can be determined easily.

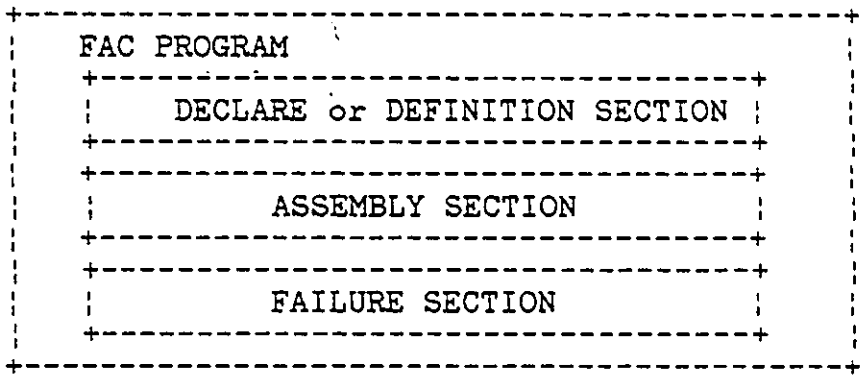
4.6.1.3 Error Processing

In assembly, the primitive command executions may be unsuccessful, and these situation must be properly handled. This can be done by performing specific failure strategies such as retrying failed assembly procedures to allow for as much assembly as possible, or sending alarm signals to the controlling system or requesting further action from another control level above. Assembly failures may consist of missing parts in the feeders, detection of broken or unfit parts, and the detection of unsuc-

cessful mating actions. Other types of failures involve more system oriented aspects such as timeouts, system abort conditions.

4.6.2 FAC CL Program Structure

To complete the programming aspect of a FAC a CL program must include three section: the DEFINITION (declare) section where part locations and/or characteristics are defined, the ASSEMBLY section where the assembly sequence is described with CL instructions. The last section can be defined as the FAILURE section where the alternative actions are defined.



The appropriateness that each FAC can be considered as a set of interacting elements was shown. From the coordination point of view they can be considered as driven entities. In this case there can be a number of conditions that must be appropriately addressed. These are:

1: the task is not applicable

2: the proper conditions for execution are not present

3: failures detected during execution of a task

4.7 Conclusion

In this chapter, it was shown that a FAC is the mechanical equivalent of a digital computer and implies the following three aspects. Firstly, a FAC can be constructed from independent elements that are coordinated by a central controller. This modularity of a FAC allows the reorganization of its elements to meet specific needs. The central controller is reprogrammed for every specific need much like a microprogrammed controller of a computer is reprogrammed to implements a new or enhanced machine language instruction set [VAIL82],[PICH84]. Secondly, the FAC can be reprogrammed in an assembly like language (CL) to execute the various production steps. This means that the manufacturing engineer can use available assembly language programming techniques to describe the product assembly to be executed by the FAC.

Since there is conceptually no major difference between a FAC and a AMC, AMC's can also be shown to be mechanically equivalent to a computer. Therefore the methodology proposed in this chapter which integrates the required elements into a coordinated FAC apply equally to AMC's and to FMC's in general.

Chapter 5

Integrating FMC's Into a FMS

5.1 Introduction

The purpose of this chapter is to show that Process Activity Diagrams (PAD) can be used as a specification tool of the manufacturing process in terms of manufacturing activities, and as a design tool for organizing and coordinating Flexible Manufacturing Cells that execute the required activities of a Flexible Manufacturing System.

In the previous chapter, a FMC was shown to be equivalent to a computer. Based upon this equivalence, a FMS can be considered as equivalent to a multiprocessor system and that it can be designed and implemented in the same manner as that of a multiple processor system.

The major problem in the design of a FMS is the full and coherent specification of the overall manufacturing process in terms of separable manufacturing activities and the required coordination. The FMS design process must also take into consideration, change-over (or retooling) costs incurred when a new product is to be manufactured. Therefore, when designing a FMS, the manufacturing engineer must execute the following steps:

1: describe in detail the whole manufacturing process in terms of a coordinated set of activities;

2: map the required activities onto FMC's;

3: identify the set of resources needed in each FMC to implement its activity (or activities);

4: reprogram, and if needed, reorganize the FMC's to execute the required activities;

5: Reprogram the transfer system to execute the various production steps in the required sequence.

There are few tools available to the manufacturing engineer to aid in performing the above steps and therefore there is a need for highly trained specialists. Not only do these specialists represent a major investment and dependence for the manufacturer, but in general they are in short supply [TRUX83].

Therefore, to provide flexible manufacturing at acceptable costs particularly in the cases of small and medium production runs, it is important to develop easy to use design tools that allow quick and orderly production reconfiguration, retooling and reprogramming. Process activity diagrams (PAD's) originally developed for the design of real-time multiple processor systems [JOAN86] have the potential of being such a tool.

5.2 PAD's: Process Activity Diagrams as a Tool for FAS Design

Process Activity Diagrams, PAD'S, are a graphical tool used for the specification and design of event-driven multiple microprocessors systems that must respond to discrete events that occur in the environment [JOAN86][KRIE87]. In these systems, PAD's are used to represent the sequencing and timing relationships between the various activities of an event driven system. As it was shown that FMC's are the mechanical equivalent to a

computer, it should be possible to apply PAD's to specify and design FMS's consisting of multiple FMC's.

PAD's consist of two components: Activity Sequence Networks (ASN's) which are modified flow diagrams used to indicate the sequencing relations, and Activity Timing Charts (ATC's) that indicate timing requirements and are similar to GANTT charts [CHAS81]. The ASN is defined as a structured construct allowing the manufacturing engineer to capture the required manufacturing activities at a high level, and to subdivide them by successive refinement into basic manufacturing activities directly executable by FAG's. A manufacturing activity is a meaningful sequence of Cell Level Instructions (CLI) which when started, can be executed from beginning to end without waiting for additional data or objects from other activities. The execution sequence of activities is determined by the data and object dependencies that may exist between activities within an manufacturing process implemented by the FAS. The ATC's specify the timing requirements of the various activities and can be used in assigning available resources to execute activities in order to avoid bottlenecks.

Any manufacturing process can be specified at various levels of abstraction or description. For example, in the assembly of a toy doll, the complete doll assembly can be specified at the highest level, with one activity, as simply ASSEMBLE DOLL. This level of description does not provide any detail about the assembly process for the doll. The next level of description could specify the assembly of the head and torso of the doll in separate activities as shown in Figure 5-1. This level still contains

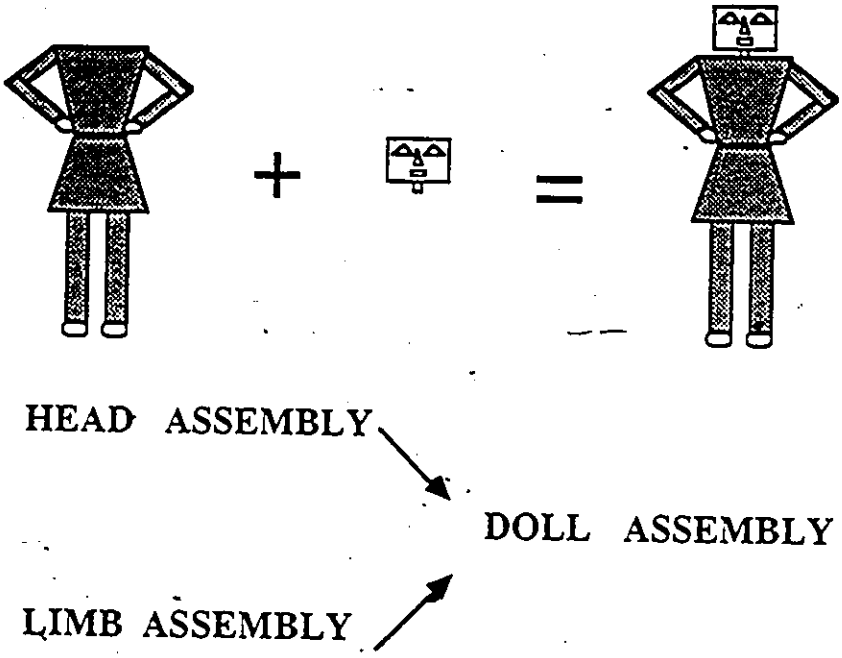


Figure 5-1. Doll Assembly.

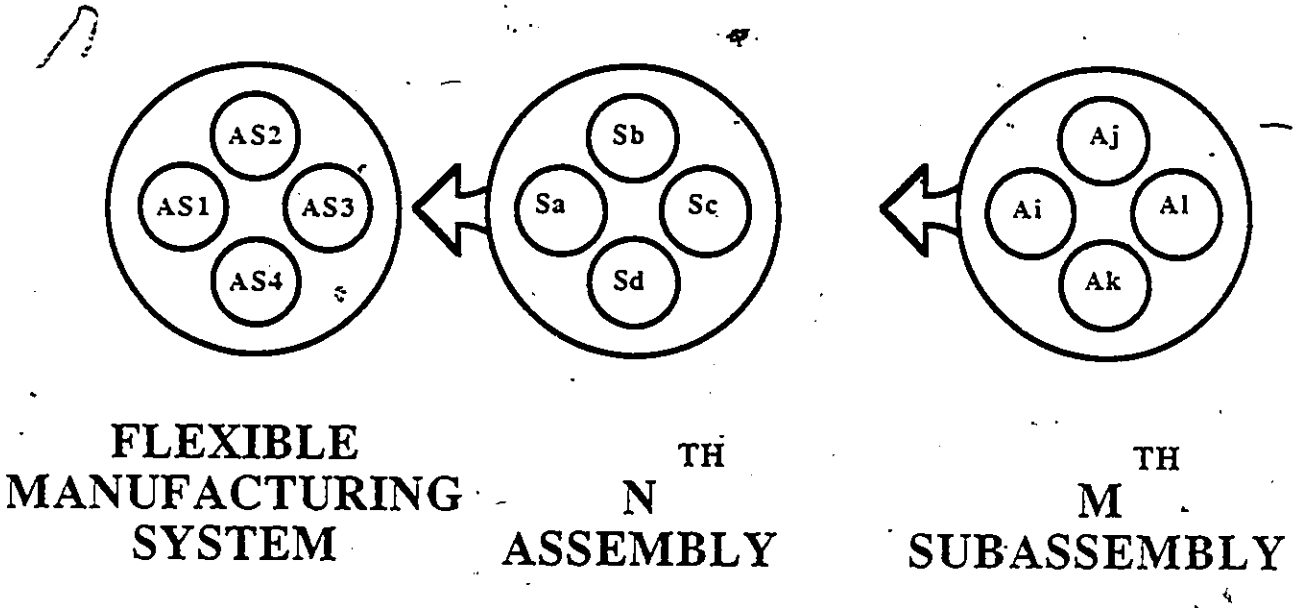


Figure 5-2. Levels of FMS Specification.

very little information on the assembly process, but it does indicate that the assembly process can be performed in separate and potentially concurrent activities: one to assemble the head, and another to assemble the torso. Final assembly is achieved by executing the last activity of assembling the doll using the torso and head.

These levels of abstractions may be refined to the lowest level, that of the primitive manufacturing commands outlined in the previous chapter. Such a description carries an enormous amount of information on the manufacturing process itself; however, it is of little use when the overall FMS is being planned. For this reason, in terms of FMS's, the lowest useful level is that of manufacturing activities executable by a single Flexible Manufacturing Cell (FMC). The breakdown of manufacturing processes is shown in Figure 5-2.

The sequence of activities of a manufacturing process can be described by Activity Sequence Networks (ASN's) consisting of a number of nodes (circles), connected by directed arcs (an arc is a line with one arrow) called triggers. ASN's are defined in terms of nine types of nodes and two types of triggers. They are used to represent all activity sequences and their associated control and data flow. The nine nodes are: ACTIVITY INITIATOR node, END node, BRANCH node, MERGE node, WAIT node, GATE node, TIMER/COUNTER node, MESSAGE node, and ABORT node. The two types of triggers are INTERNAL triggers, and EXTERNAL triggers.

The various nodes allow for the description of concurrent

execution of activities, the selection of different activity sequences depending on one or more conditions, timeouts, suspension of execution, etc. An ASN is said to be initially in the DORMANT mode until it is put in the AWAKENED mode by its starting event. When it is terminated, it is said to be back into the DORMANT mode.

5.2.1 Elements of PAD's

The basic node is the ACTIVITY INITIATOR node shown in Figure 5-3. It is the only node of an ASN that deals directly with the execution of activities. The activity Ai associated with the node is defined by an arc leaving its center and pointing to its name. Input triggers are said to be activated when the conditions they represent are satisfied. These conditions may relate to the availability of the data and/or objects required for the activity to execute, or may represent control information such as precedence relationships between activities not based on data and objects. The activity can only be started when all input triggers are activated. The initiation of the activity results in the resetting of the node to forget all past triggers. When the activity terminates, all the node's output triggers are activated.

A TIMING BAR can be associated with every activity whose length is proportional to the execution time of the activity. The TIMING BAR may be further broken down into three distinct intervals: input data and/or object transfer time, actual activity execution time, and output data and/or object transfer time.

During early design stages, time bounds or estimates may be used because these times may be undefined. Even so, the execution time of an activity may not be fixed because of the its (data) object dependencies. In this case, average or worst-case activity execution times must be used.

There are two types of triggers: internal triggers and external triggers. Internal triggers are generated by either the completion of some activity or by the occurrence of some condition and originate and end within the network. For example, the completion of an activity is an internal event represented by the activation of the output triggers from the activity initiator node. In general, each event has information associated with it. External triggers are generated as a result of conditions external to the process and are represented by arcs with no node at their origin but a label identifying the external event (see Figure 5-4).

If the sequencing relationships do not include any decisions, then only activity initiator nodes are needed in the ASN as shown in Figure 5-4. To represent more complex sequences of activities, additional nodes are introduced. These are the END, BRANCH, MERGE, WAIT, GATE, TIMER/COUNTER, MESSAGE, and ABORT shown in Figure 5-5.

The END node is identified by a circle with an 'E' inside and is used to indicate the termination point of an ASN. When all its input triggers are activated, all the nodes of the ASN are reset, and the ASN is said to be in the DORMANT mode. This is the only

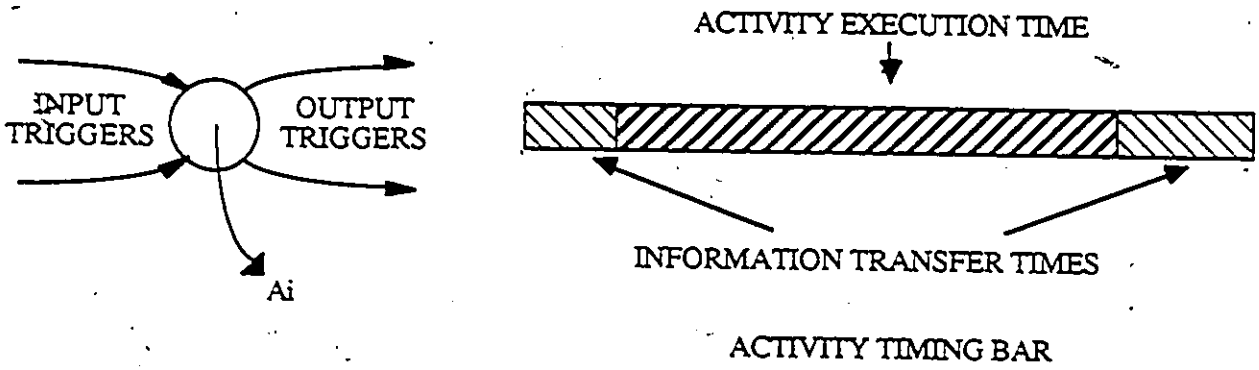


Figure 5-3. Activity Initiator Node With Timing Bar.

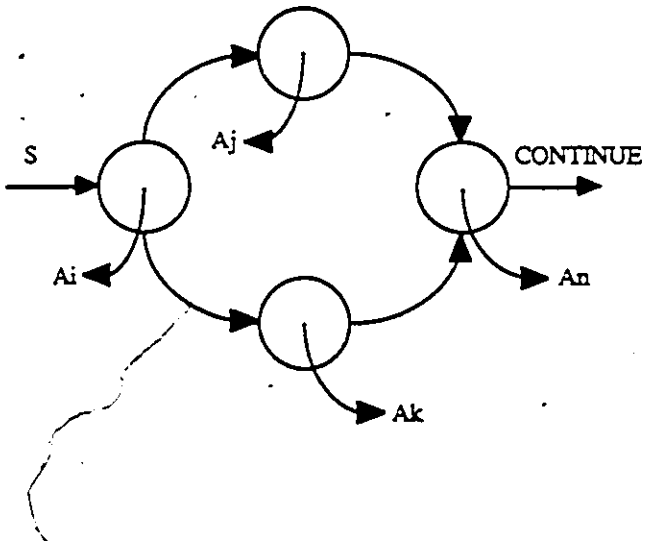
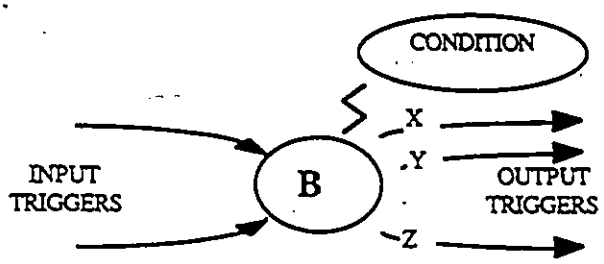
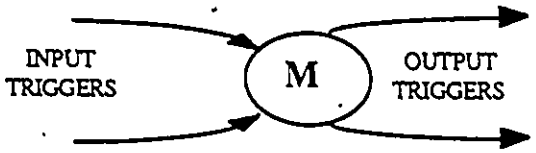


Figure 5-4. ASN Only Consisting Of Activity Initiator Nodes With Internal and External Triggers.

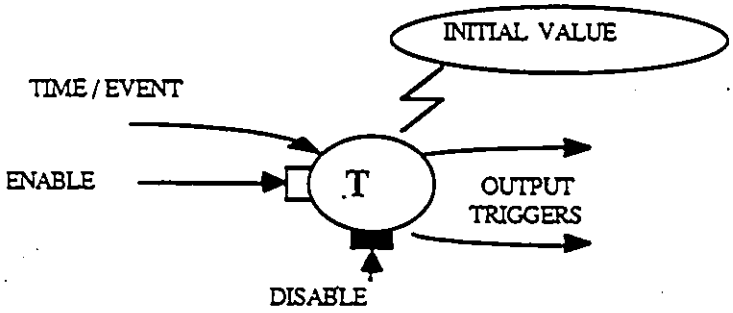
BRANCH NODE



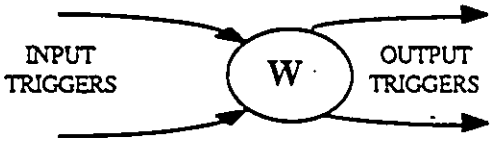
MERGE NODE



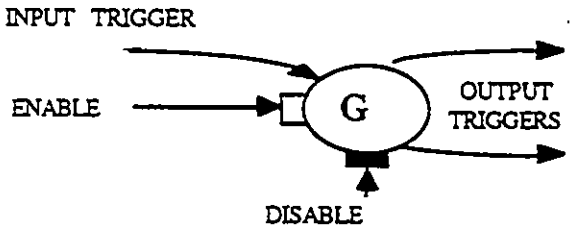
TIMER / COUNTER NODE



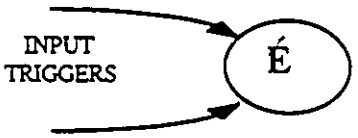
WAIT NODE



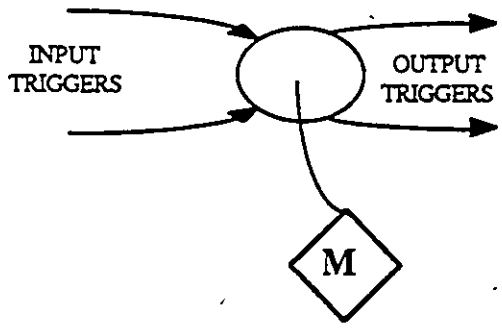
GATE NODE



END NODE



MESSAGE NODE



ABORT NODE

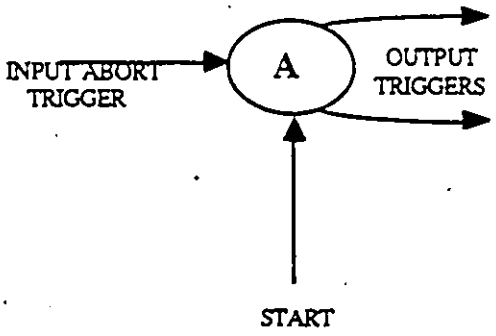


Figure 5-5. ASN Nodes.

node that has no output triggers.

The BRANCH node is identified by a circle with a 'B' inside and an ellipse with the condition for branching described inside. BRANCH nodes permit ASN's to represent selection of various activity sequences depending on the condition indicated in the associated ellipse. It operates as follows: when all its input triggers are active, a condition variable is examined and only those output triggers corresponding to the condition are activated and the node is reset to forget all past input triggers.

The MERGE node is identified by a circle with an 'M' inside. A MERGE node activates its output triggers when any one of its input triggers is activated. This allows, among other uses, the merging of various paths of the branch node and the implementation of feedback paths. The MERGE node does not need to remember the activation of its input triggers since it immediately takes action when any one of its input triggers is activated. This is the only OR type node in ASN's, while all others are AND type, i.e. all their input triggers must have been activated for an action to occur.

The WAIT node is defined as a circle with a 'W' inside. A WAIT node only allows the joining of concurrent paths without executing an activity. It can be considered as an activity initiator node without an activity. It simply waits for all input triggers before activating its output triggers. It is often used as an editing convenience to 'clean up' ASN's by reducing the number of arcs in the network.

The GATE node is defined as a circle with a 'G' inside, with a small light square and a small filled square on the outside periphery of the circle. The GATE node is an ASN node used to block the propagation of triggers under certain conditions. Beyond the usual input and output triggers, it also has one enable input, represented by the light square, and one disable input, represented by the filled square. The GATE node is initially in the disabled state. When the GATE node receives a trigger on its enable input, it is put in the enabled state and it acts like a WAIT node: it will remember its input triggers, and will issue output triggers only when all input triggers have been activated. When the output triggers are issued, it puts itself in the disabled state. If the disable input is activated when the gate is in the enabled state, it forgets all past input triggers and is put in the disabled state. In the disable state, the GATE node ignores all inputs except the enable input.

The TIMER/COUNTER node is the most complex node and consists of a circle with a 'T' or 'C' inside, a light square and a filled square on the circle periphery, and an ellipse connected to the circle by a directed arc (not a trigger). It also has one enable input and one disable input, but can only have one input trigger. The node is initially put in the disabled state and is therefore non-operational. When it is put in the enabled state, a start-up value, represented by the counter variable value defined inside in the ellipse, is entered in a 'counter register'. Whenever its input trigger is activated, the counter is decremented provided that it was not put in the disabled state through its disable

input. When the counter reaches zero, all output triggers are activated and the node puts itself in the disabled state. The node can also act as a timer if the input trigger is defined as a system clock event. If the node gets a trigger on its disable input before its counter reaches zero or before the time-out occurs, the node stops counting and puts itself in the disabled state. When it is put back into the enabled state, the initial count is loaded once more into the 'counter register'. The TIMER/COUNTER node is used to provide time-outs or to count a required number of events before activating its output triggers.

The MESSAGE node is defined as a circle with an arc beginning at the center and connected to a diamond with a 'M' inside. This node behaves in the same way as an ACTIVITY INITIATOR node except that it initiates the transmission of a message; once the message is transmitted, all output triggers become active. This type of ASN node is used to describe communicating PAD's, in which direct information transfer between two processes may be accomplished.

The ABORT node is defined as a circle with the a 'A' inside. An ABORT node allows the selective abortion of a process. An ABORT node is awakened by an input trigger from the START event of a process defined by a PAD. Upon the activation of the ABORT event on its other input trigger, it terminates the process initiated by the START event. The ABORT node is part of an ABORT PROCESS, which may include a SAVE SUBPROCESS.

Also, the process or PAD to be aborted may include CRITICAL REGIONS which once entered must be completed before the process

can be terminated. As shown in Figure 5-6, CRITICAL REGIONS are defined between two inpointing light triangles called CRITICAL PATH DESCRIPTORS which are put on the trigger arcs.

When specifying a process using ASN's, the following aspects must be considered:

1: Each ASN must start with an external event; the external event may be provided by another ASN, by an external sensor or condition, or by the main controller.
2: Each ASN should terminate with an END node unless it is to execute continuously, in which case there must be a feedback path.

It is to be noted that several ASN's may be required to completely specify a process.

Activity Timing Charts (ATC's), the other component of PAD's, show the execution times of the various activities of an ASN. The ATC of the ASN in Figure 5-4 is shown in Figure 5-7. Before implementation, execution times must be estimated. Even after implementation, execution times for activities in some activity executors (FMC's) can only be specified in terms of upper and lower bounds or average times.

Figure 5-7 shows the nesting property of PAD's. The entire ATC can be considered as a single composite activity at a higher level, or as a single activity that can be decomposed into a PAD. This feature of PAD's allows the system designer to start from a coarse specification of the process to a complete specification

of an assembly process by doing successive refinements.

ATC's are useful to obtain rough estimates of assembly process execution times. By using Critical Path Method (CPM), Program Evaluation and Review Technique (PERT) or other such methods, it is possible to find which activities would benefit most from speed improvements. Using ATC's, the required allocation of FMC's can be determined, i.e. the number of FMC's needed, the activities to be performed by the FMC's, and the hardware needed in every FMC. A simulator, developed in [JOAN86], can be used to test allocation of FMC's before implementation. Using actual execution times and available resource information from the shop floor, the overall manufacturing can be optimized.

5.2.2 PAD Usage

The use of PAD's is shown in the following small example. In this example, let us say that it is necessary to manufacture K parts of two types, parts P1 and P2, before the manufacturing process can proceed onto the next step. The ASN for the manufacturing of the product is shown in Figure 5-8. The ASN is activated when it receives the START input trigger and is applied to the first node, MERGE node M1. MERGE node M1 decouples the starting process from the rest of the ASN. MERGE node M1's output triggers activate MERGE node M2 which accommodates the feedback path from WAIT node W1 via GATE node G1 back to MERGE node M2. Another output trigger from MERGE node M1 is applied to COUNTER node C1 to initialize it for K counts. The output triggers of MERGE node M2 initiate activities A1 and A2, simultaneously decrease the counter of COUNTER node C1 by one to indicate that

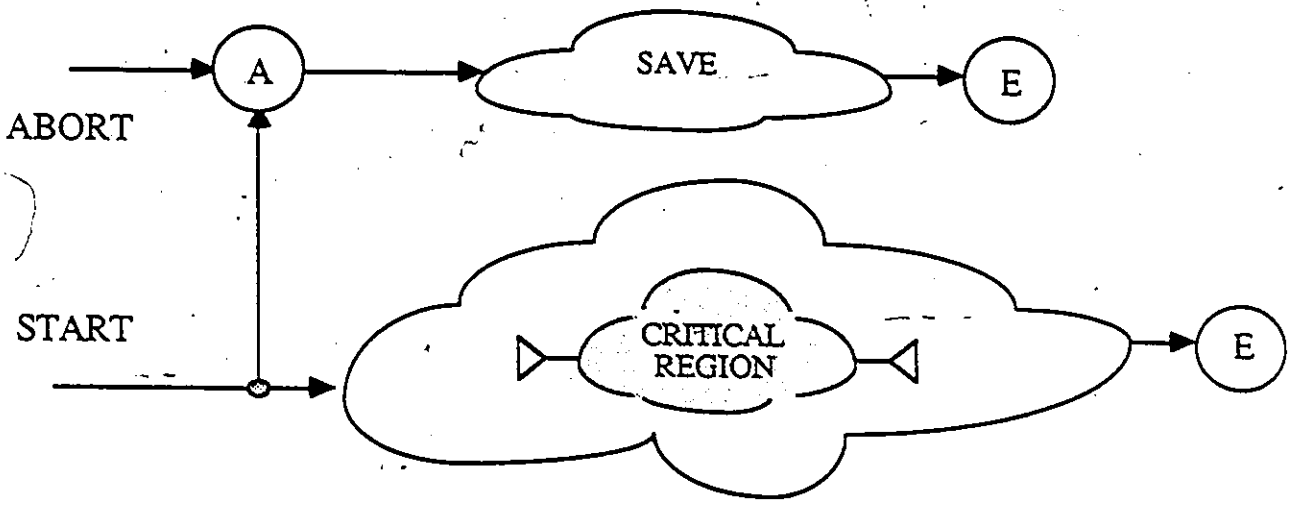


Figure 5-6. CRITICAL REGIONS and CRITICAL PATH DESCRIPTORS Usage.

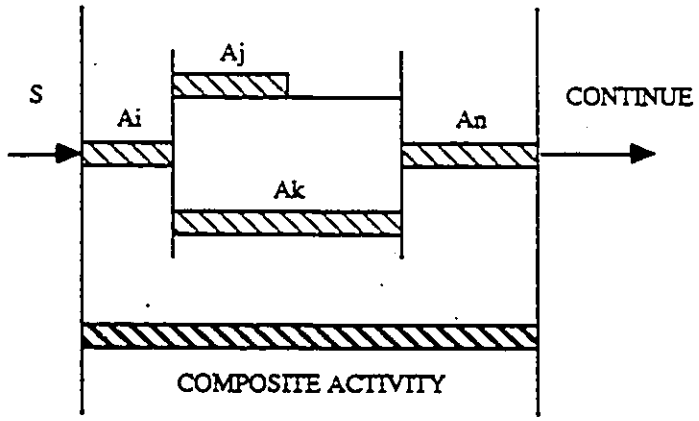


Figure 5-7. ATC of ASN of Figure 5-4.

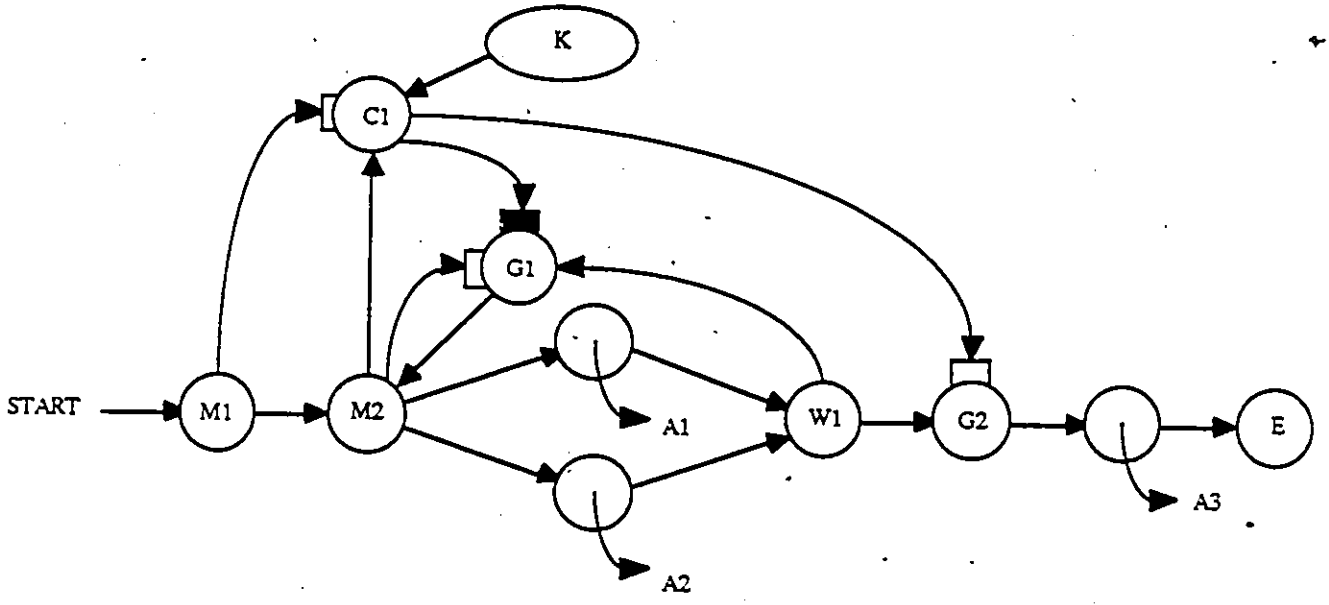


Figure 5-8. Example ASN.

		NODES									
		M1	M2	A1	A2	W1	C1	G1	G2	A3	E
	START	I	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	M1	Ø	I	Ø	Ø	Ø	I	Ø	Ø	Ø	Ø
E	M2	Ø	Ø	I	I	Ø	I	E	Ø	Ø	Ø
V	A1	Ø	Ø	Ø	Ø	I	Ø	Ø	Ø	Ø	Ø
E	A2	Ø	Ø	Ø	Ø	I	Ø	Ø	Ø	Ø	Ø
N	W1	Ø	Ø	Ø	Ø	Ø	Ø	I	I	Ø	Ø
T	C1	Ø	Ø	Ø	Ø	Ø	Ø	D	E	Ø	Ø
S	G1	Ø	I	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø
	G2	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	I	Ø
	A3	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	Ø	I

Figure 5-9. ASN Matrix Representation.

K-1 parts are left to process, and enable GATE node G1 to allow feedback since GATE nodes disable themselves when they activate their output triggers.

When the MERGE node M2 sends its K-th set of output triggers, the last two parts will be processed by activities A1 and A2. Simultaneously, the counter reaches zero and activates its output triggers. These triggers disable GATE node G1, removing the feedback path, and enable GATE node G2. When COUNTER node C1 issues its output triggers, it also automatically disables itself. When both activities A1 and A2 have been completely executed, activity A3 which required two sets of K parts P1 and P2, is activated via GATE node G2. When GATE node issues its output trigger, it disables itself. Since GATE node G1 was disabled, no new P1 and P2 parts will be processed until a new START trigger is applied to the ASN. The ASN returns to its initial state when activity A3 terminates.

For ease of processing, a matrix representation of the ASN is introduced. The columns of the matrix represent the nodes of the ASN, one column per node, while the rows of the matrix represent the external and internal events associated with the ASN. Each entry in the ASN matrix can take one of four possible values: I, which means that the input trigger is used to activate the node; E, which means that the input trigger is used to enable the node; D, which means that the input trigger is used to disable the node; and Ø which means that this trigger is not applied to that node. The matrix representation of the ASN example of Figure 5-8 given in Figure 5-9.

5.3 Application of PAD's to Printed Circuit Board Manufacturing

In this section, the applicability of PAD's to FMS design is presented by indicating their use for the specification of the manufacturing process, and of the organization and coordination of the FMS. First, the general PCB manufacturing process is described at a high level to show the main phases of PCB manufacturing and to indicate the scope of the application; second, each phase is described in more detail by identifying the product information required from the CAE/CAD/CAM system and by explaining where and how it is used; third, a detailed description of the PCB assembly process is presented using a simple PCB assembly example; fourth, PAD's are used to design and specify the FAS that implements the described PCB assembly process; fifth, the type of information useful in the design of the FMS is identified and it is shown that much of this information can be extracted from PAD's. Such information is mainly concern with aspects such as the identification and specification of needed assembly activities, their control, their sequencing, their execution time, the resources needed to implement the activities executed by the FAC's, and the programming requirements and specifications for executing the activities on the different FAC's of the FAS. And sixth, the organization, architecture and programming of one of the FAC's is also demonstrated using the methodology based on computer equivalence.

5.3.1 The General PCB Manufacturing Process

The general PCB manufacturing process of an arbitrary number

N PCB's for a fully automated FMS is shown in the ASN of Figure 5-10. In each loop of the ASN are manufactured K PCB's, and the loop is executed I times to give N=IxK PCB's. The value of K is often determined, for example, by the tray capacity used in the transfer of PCB's in the FMS, or is determined by the number of PCB's that can be cut from a single sheet. The main activities of PCB Manufacturing are: plant setup, PCB making, part and component testing and preparation, PCB assembly and soldering, and assembled PCB functional testing.

5.3.1.1 Plant Setup

The FMS is first set up with all the information necessary to carry out the manufacturing of the specific PCB. Programs and data are down-loaded from the CAE/CAD/CAM systems in the formats needed by the devices that use them. When the setup process is complete, PCB making, part and component testing and preparation activities can be initiated.

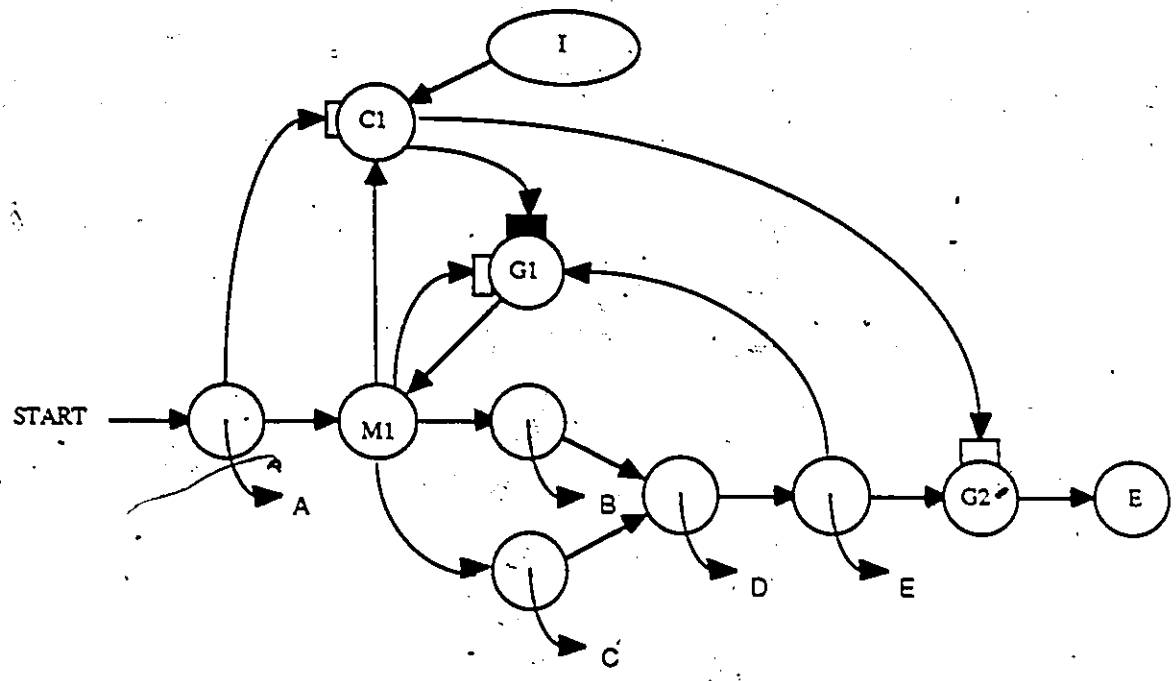
5.3.1.2 PCB Making Subprocess

The PCB making subprocess begins with prepared metal electro-plated sheets. Sheets are cut to appropriate sizes for the circuit at hand and are submitted to the separate activities necessary to imbed them with the wiring of the circuit.

The next step in the PCB making subprocess is the etching process. In the etching process, circuit masks are created by means of photographic methods using the physical layout data generated after CAD/CAM tools have completely layed out the

circuit. Patterns are extracted from the layout in the form of rectangles and are used to control a pattern photo machine. The pattern photo machine uses this information to draw rectangles on clear Mylar sheets prepared with light sensitive material that changes properties when exposed to light. The sheets are then developed leaving on them a copy of the circuit tracks and the connection pads for a layer. Similarly, a complement copy of the circuit can be put on the Mylar sheets. These Mylar sheets are use repeatedly with copper-plated boards. The boards are coated with light sensitive material; by a contact exposure method, the circuit features are copied onto the boards. Unexposed material is washed away and the board is put into an etching tank where they remain until the exposed copper is removed. The remaining copper represents the desired circuit.

The boards are then brought to NC drilling machines. Holes of the appropriate sizes are drilled through the board using the position and hole size information generated by the CAD/CAM system and down-loaded to the NC machines. All hole position information is provided relative to the prime manufacturing hole. The board is held into place by a fixture that can be controlled in the X- and Y directions. Thus, all holes are drilled anywhere on the board. If holes are of different sizes, either a separate machine is used to drill holes of a specific size, or if the NC machine has the capacity, all holes of a specific size are drilled first, and the machine exchanges the drill bit with one of a different size until all the holes have been drilled. Holes consist of pin holes, vias (used to connect different layers),



A = **Plant Setup:** setup of all FMC's, down loading of FMC programs, data, etc. needed for full production of **N** PCB's.

B = **PCB Board Making: and Testing** the etching of circuit onto PCB layers and drilling of holes, vias, and layer-to-layer connecting, done for **K** PCB's.

C = **Part and Component: Testing and Preparation** incoming parts to be used in the assembly process including integrated circuits (IC's), discrete components, and parts, are tested for physical and functional defects, and good parts and components are prepared for assembly of **K** PCB's.

D = **PCB Assembly and: Soldering** good boards are fitted with components on the component side of the PCB; the PCB is passed through a wave soldering machine that solders all exposed pins on the pin side of the PCB, for **K** PCB's.

E = **Assembled PCB: Testing** Functional, structural, and electrical testing is done for **K** PCB's and bad boards are put in a repair loop, good boards are sent to final product manufacturing.

Figure 5-10. ASN of the PCB Manufacturing Process for **N** PCB's.

and manufacturing holes used in securing the PCB in fixtures during the rest of the manufacturing process, or for fastening purposes during product assembly.

At this point, two boards for the same circuit are ready to be pressed together in multiple layer board manufacturing. This is done by heating the stacked boards under pressure. The amounts of heat and pressure depend on the board material, the number of boards, and the required tolerances that guarantee hole alignment and keep the circuit tracks intact.

After impression, a back-plating process is done to connect vias between layers. The prime manufacturing hole and other manufacturing holes not used as feed-through's are plugged to prevent back-plating. The impression and back-plating processes are repeated until all layers have been put together in one multi-layer PCB.

The next activity to take place in the PCB making process is the depositing of the solder-resist screen. This screen is needed to prevent any solder deposited by the wave-soldering machines to short any of the tracks. The exposed places on the board are usually pin holes and the like, where soldering must take place.

Another masking process called silk-screening, takes place to draw on the lower density boards the component markings, identification numbers, etc.

This PCB making subprocess is completed after the bare PCB's are tested by inspecting for defects, and by performing electri-

cal and tolerance tests on the board. Bad boards are trashed, and good boards are stored in special bins, magazines, or racks of trays for later retrieval during component assembly.

The PCB making subprocess can be represented by the ASN of Figure 5-11.

5.3.1.3 PCB Part and Component Testing and Preparation

In this subprocess, the required set of parts and components needed for assembly are tested with respect to their physical or geometrical, structural, electrical, and functional attributes. After testing, good components are prepared for assembly by sorting and arranging them in suitable ways needed for part feeding to the assembly FAC's.

The testing aspect begins with a visual inspection process that detects geometrical defects early in the manufacturing process, and rejects bad items. Parts such as screws, nuts, washers, etc., used in the assembly process usually only require visual inspection.

Other components usually require more and different testing since they contribute to the function or behavior of the final product. Specialized testers are used to test different types of components such as transistors, inductors, capacitors, and resistors. Tests conducted for these parts include behavioral pattern matching (e.g. transistor gain curves, logic behavior), electrical measurements of key variables such as input and output currents and voltages, and timing and frequency responses. Test

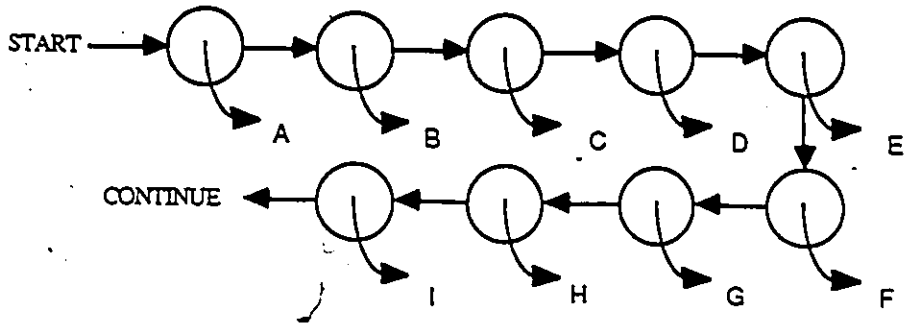
results are compared to expected results within an allowed tolerance. Devices that do not conform to expected behavior within the allowed tolerances are rejected.

More complex devices such as integrated circuits, require even more advanced and complex testing processes and hardware. Tests conducted include electrical and functional behavior.

Preparation involves the proper positioning and orientation of parts and components into containers that conserve this information. Preparation for box-shaped components is done by putting them in compartmentalized feeding trays in the desired presentation manner to conserve their orientation before use and during transfer. In this way, part feeding at the FAC is greatly simplified since parts can be assumed to arrive at FAC's in a known position and orientation, reducing the need for very advanced and complex elements, sensors, and their programming at FAC's. Other types of parts such as axial components are put into feeding ribbons in the proper sequence for axial component assembly using special axial component insertion machines.

After testing and preparation is completed, parts and components are sent via a suitable part transfer system to the specific FAC's for assembly.

An ASN describing the PCB part and component testing subprocess is shown in Figure 5-12.



A = **Mask and Screen Making**: circuit masks are made using mask making hardware which use the connection information generated by the CAD/CAM system in the form of patterns extracted from the layout in the form of rectangles and transferred onto masks; these masks are made using photographic processes that put the circuit on Mylar plates that will be used in the etching of the bare copper-plated PCB; one mask is made for every side, for each PCB layer.

B = **Sheet Cutting**: copper plated sheets are cut to the required size for the PCB using layout dimension information from the CAD/CAM system.

C = **Circuit Etching**: the cut copper boards are coated with a light sensitive material; boards are then contact exposed using the masks, photo-developed to 'fix' the exposed circuit; unexposed areas of the board are etched away, leaving the circuit tracks.

D = **Hole Drilling**: vias, feed-throughs, pins, manufacturing holes and the reference or prime manufacturing hole are drilled using positional and size information generated by the CAD/CAM system.

E = **Layer Impression**: sandwiching of several single-sided PCB's into one multi-layer PCB using heat and pressure.

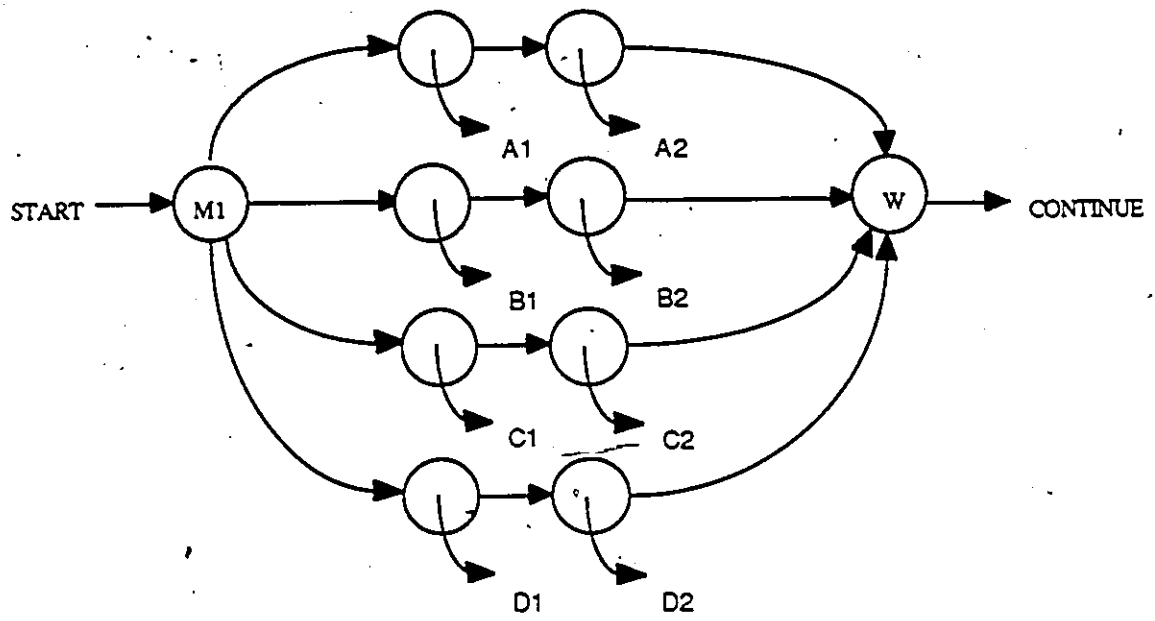
F = **Back Plating**: connecting vias between layers.

G = **Solder Resist**: circuitry that is not exposed for soldering or testing purposes is covered with solder resistant material to prevent shorts between tracks and to permit wave soldering to take place after components have been installed.

H = **Silkscreen Marking**: component location markings on the PCB (not for high density component boards).

I = **Bare PCB Testing**: PCB's are finally visually inspected and electrically tested for possible short circuits, defects; statistical data are collected with respect to the quality of production.

Figure 5-11. ASN of the PCB Making Subprocess.



A1= **Mechanical Part Testing**: Parts used in the assembly process to fasten other parts and components are visually inspected for any geometric defect.

A2= **Mechanical Parts Preparation**: Parts are prepared for the FAC part feeders.

B1= **Passive Component Testing**: Passive components such as resistors, capacitors, transformers, and the like are visually inspected and then functionally tested to verify that they meet tolerance specifications.

B2= **Passive Component Preparation**: Passive components are prepared for FAC feeders.

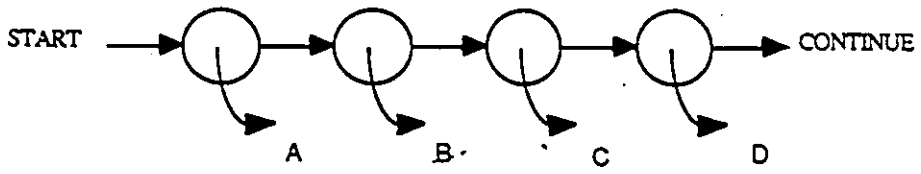
C1= **Active Component Testing**: Active components, such as diodes and transistors, are visually inspected and tested for the proper behavior within the allowed tolerances.

C2= **Active Component Preparation**: All active components are prepared for FAC feeders.

D1= **Complex Component Testing**: Complex component such as integrated circuits are visually inspected and then tested for the proper behavior within the allowed tolerances.

D2= **Complex Component Preparation**: Complex components are sorted and arranged for feeders used during assembly.

Figure 5-12. ASN of PCB Part and Component Testing and Preparation



A = **Axial Component Assembly**: Components shaped like cylinders such as resistors, capacitors, diodes , etc. are assembled to K PCB's.

B = **Box-shaped Component Assembly**: Components shaped like boxes like Integrated Circuits (IC) are added to K PCB's; often several pins must be inserted at the same time.

C = **Odd-Shaped Component Assembly**: Components with varying shapes and number of pins, and parts used to fasten other parts to the PCB are added to K PCB's;

D = **PCB Soldering**: K PCB's with the added components are soldered;

Figure 5-13. ASN for PCB Assembly Subprocess.

5.3.1.4 PCB Assembly

When the PCB making and PCB parts and components testing activities have been completed, the PCB assembly activity can be initiated. The PCB assembly activity consists of the following main subactivities divided along the lines of assembly object characteristics: axial component assembly, box-shaped component assembly, odd-shaped component assembly, and finally soldering. For PCB's that use newer technology such as surface mounted objects, further assembly may be required after the first PCB soldering activity.

Axial components are circuit objects that have a cylindrical shape with usually two ends or leads. These leads must be inserted into holes. Box-shaped components several pins often on two sides. All pins must be inserted into holes of appropriate dimensions and spacing simultaneously. Odd-Shaped parts and components consist of odd shaped circuit and mechanical devices that have different purposes. Some serve in a circuit function whose pins are inserted in holes, and others are used for purposes such as fastening (mounting screws, nuts), heat dissipation (heat sinks), etc. All components requiring soldering are done by a wave-soldering device.

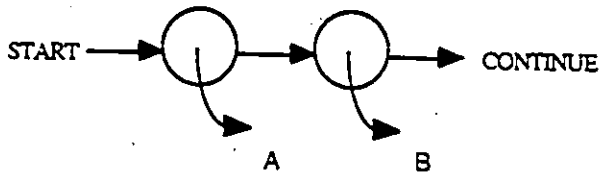
The PCB assembly subprocess is presented in the ASN of Figure 5-13 showing a first attempt to define details of the assembly process in a step-wise refinement manner. The details of the assembly subprocess will be considered in section 5.3.2.

5.3.1.5 Assembled PCB Testing

After the components have been soldered and parts fastened to the PCB's, these are checked for physical and functional defects.

Physical defect detection usually consists of visual inspection. Using sophisticated pattern recognition capabilities, PCB's are scanned for solder bridges, gaps, cold solder joints, bent pins or leads, missed insertions, accidental track cutting, and parts damaged during assembly. In this way most defects can be detected before expensive functional testing. The inspection phase consists of attempting to find and identify these anomalies, tag and record them for repair and quality control, and to redirect 'bad' boards toward a repair loop. In the repair loop, decisions are taken whether to keep and repair the boards, or to dispose of them. The decisions taken depend upon the severity of the defects, their number, and the production rate of the system. 'Good' boards are allowed through to the next testing phase: functional testing.

In functional testing, the behavior of each PCB circuit under test is compared with the expected behavior. Tests of this type are more elaborate and require large amounts of data generated during the design phase of the circuit in CAE functional circuit simulation. For PCB circuits, functional testing requires very advanced and complex computer driven testers called Automated Testing Equipment (ATE). These testers must be able to generate appropriate stimuli for circuit testing and accept behavioral patterns of the circuit and compare them with expected patterns



A = **Physical Defect Detection:** K PCB's are visually inspected for any physical abnormality.

B = **Functional Testing:** K PCB's are tested with respect to function and performance specifications.

Figure 5-14. ASN of Assembled PCB Testing Subprocess for K PCB's.

Parts	Number
PCB board	1
Axial Components	
Resistors	3
Capacitors	3
Box-Shaped Components	
Integrated Circuits	
U1 (14 pins)	1
U2 (24 pins)	1
Miniature Transformer (8 pins)	1
Odd-Shaped Components and Parts	
Circuit Components	
Power Transistors	2
25 Pin Connector	1
Fastening and Other Parts	
Screws Type 1	4
Screws Type 2	2
Lock Washers	4
Nuts Type 1	4
Nuts Type 2	2
Transistor Heat Sinks	2

Figure 5-15. List of Components and Parts for PCB Example.

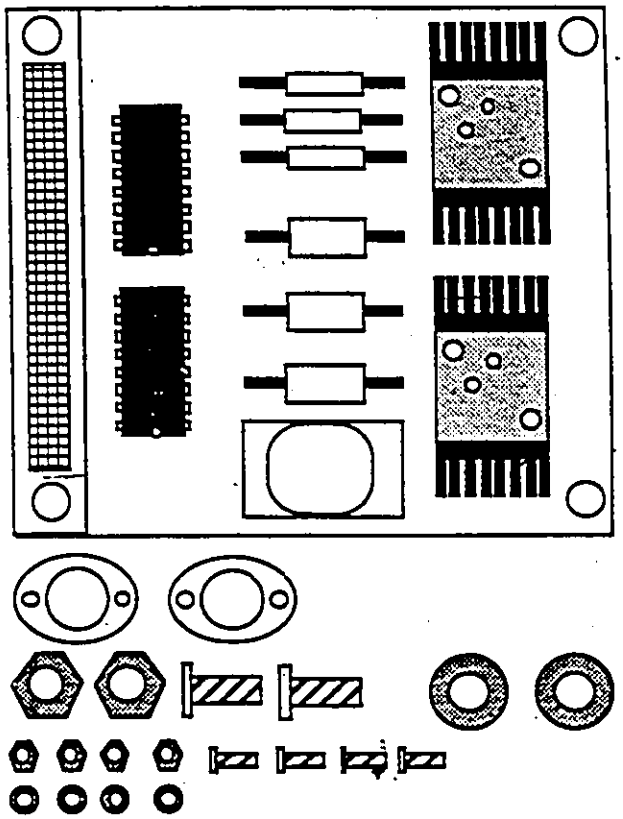


Figure 5-16. PCB Example.

found during functional simulation for the same stimuli. Although the subject of testability and testing is beyond the scope of this thesis, it should be noted that every PCB must be tested by the ATE machines and pass the tests before being shipped to product integration in order to achieve high product reliability in the field.

An ASN describing the assembled PCB testing subprocess is shown in Figure 5-14.

5.3.2 PCB Assembly Subprocess

In this section, discussions will focus on the PCB assembly subprocess, described by the composite ASN in Figure 5-13, to show how the subprocess can be refined down to the lowest level and define the FAC's needed to execute the activities. A simple PCB example, shown in Figure 5-16, will be used to illustrate the assembly steps. The list of parts and components required for the assembly of the example PCB is given in Figure 5-15.

As outlined in section 5.3.1, the PCB's to be assembled transferred in a slotted tray holding K PCB's. The main phases of the assembly subprocess will be considered separately, namely axial component assembly, box-shaped component assembly, odd-shaped component assembly, and PCB soldering.

5.3.2.1 Axial Component Assembly Subprocess

Axial components are the simplest to assemble since special purpose automatic insertion machines exist to do the job. It is assumed that the data used in this FAC were generated by the CAD

system: part layout was generated following specific optimal placement algorithms that take into account the part dimensions, the track width and lengths, track spacing, hole sizes, and even electrical criteria. The CAM system accesses part information from a common design and manufacturing data-base. All part position information is collected and is down-loaded to insertion machines. Part position data are used by the insertion machine to insert the correct axial device in the correct position on the board. The algorithm used by the insertion machine is extracted from the group technology area of the data-base and is also down loaded.

The axial component assembly subprocess is shown in the ASN of Figure 5-17. In this ASN, all axial components are added to **K** PCB's. The **K** PCB's are presented to the FAC in a tray of **K** slots. Each PCB is taken one at a time, the axial components are added to it, the PCB is returned to its original slot, and the same operation is executed repeatedly until **K** PCB's are done. The completed set of PCB's in the tray are then transferred to the next FAC for further assembly.

The detailed behavior of the AXIAL COMPONENT ASSEMBLY ASN is as follows:

1: the ASN is activated by the START trigger;
2: the START trigger activates the MERGE node M11;
3: MERGE node M11 simultaneously activates MERGE nodes M12, and M14; node M11 decouples the start process from the rest of the ASN behavior;

4: MERGE node M12's output trigger initiates activity A11, GET PCB TRAY;

5: MERGE node M14's output trigger ensures that at initial start up, WAIT node W11 is ready to respond to its other input trigger from GATE node G12;

6: ACTIVITY node A11 GET PCB TRAY gets a tray containing K PCB's that are ready for assembly, from the PCB making process or from inventory storage; when it is completed, one of its output triggers initializes the COUNTER node C11 to K so that it will count K PCB's, another output trigger activates MERGE node M13;

7: MERGE node M13 initiates ACTIVITY node A12, ASSEMBLE AXIAL COMPONENTS, enables GATE node G11, and triggers the COUNTER node C11 to count one event;

8: on every input trigger from MERGE node M13, COUNTER node C11 decreases its counter by one and tests if its limit has been reached; if the limit is reached, output triggers are issued; if not, no triggers are issued from it;

9: when ACTIVITY node A12 completes, it sends output triggers to GATE node G11 and GATE node G12; if GATE node G11 is enabled, MERGE node M13 is activated which results in ACTIVITY node A12 being initiated again; this means that repeated initiation of activity A2 is carried out; with COUNTER node C11 controlling GATE node G11, there will be K activations of activity A12 for K PCB's in a PCB tray; also, since MERGE node M13 is activated, active triggers are put on COUNTER node C11, and on the enable input of GATE node G11 being re-

enabled since GATE nodes disable themselves after issuing output triggers; since GATE node G12 is disabled, the input trigger from ACTIVITY node A12 is ignored;

10: when the limit of COUNTER node C11 has been reached, it activates its output triggers; one of its output triggers is sent to GATE node G11 to disable it, with the result that this will stop any further initiation of ACTIVITY node A12; disabling GATE node G11 means that K PCB's have been assembled with the axial components; the other output trigger enables GATE node G12;

11: when ACTIVITY node A12 completes for the K-th time, its output trigger to GATE node G12 is recognized since it is enabled at this point, and the output trigger is sent to WAIT node W11 whose other input trigger had already been activated;

12: WAIT node W11 activates its output triggers which initiate ACTIVITY node A13, TRANSFER.PCB TRAY, and activates MERGE node M12, restarting the whole process;

13: When ACTIVITY node A13 completes, output trigger NEXT1 is activated; this trigger is input the the next phase of the PCB assembly process, BOX-SHAPED ASSEMBLY, represented by the ASN in Figure 5-18;

14: input trigger READY1 originates from the BOX-SHAPED ASSEMBLY subprocess ASN, and means that the transferred PCB tray has been accepted; READY1 activates MERGE node M14 which activates one of the input triggers to WAIT node W11; combination of nodes M14, W11, A13, and triggers NEXT1 and READY1 establish the coordination required between the AXIAL COMPONENT

ASSEMBLY and BOX-SHAPED ASSEMBLY subprocess ASN's (or assembly phases); this ASN of the AXIAL COMPONENTS ASSEMBLY phase allows the next tray of K PCB's to be assembled with its axial components while the just finished tray is being transferred to the BOX-SHAPED ASSEMBLY phase; also, the ASN does not permit the transfer of the next tray until the previous tray has been accepted by the following phase (see BOX-SHAPED ASSEMBLY ASN).

From the above ASN, the following requirements can be identified for the AXIAL COMPONENT ASSEMBLY FAC implementing the activity of ACTIVITY node A12:

1: PCB feeding: Empty PCB's are presented to the FAC in a known position from a tray. Using a robot manipulator and sensory system, a PCB is selected from the tray and is latched into place on a fixture of suitable design using pins and bracing springs that conserve its positional information. The fixture can be controlled to turn the PCB in 90 degree increments and can be moved laterally in the X and Y directions. This allows the axial components to be inserted at the right place, in the desired orientation. Each PCB has a reference hole from which all positioning information is related. The PCB is held in place using the reference hole and other suitable manufacturing holes.

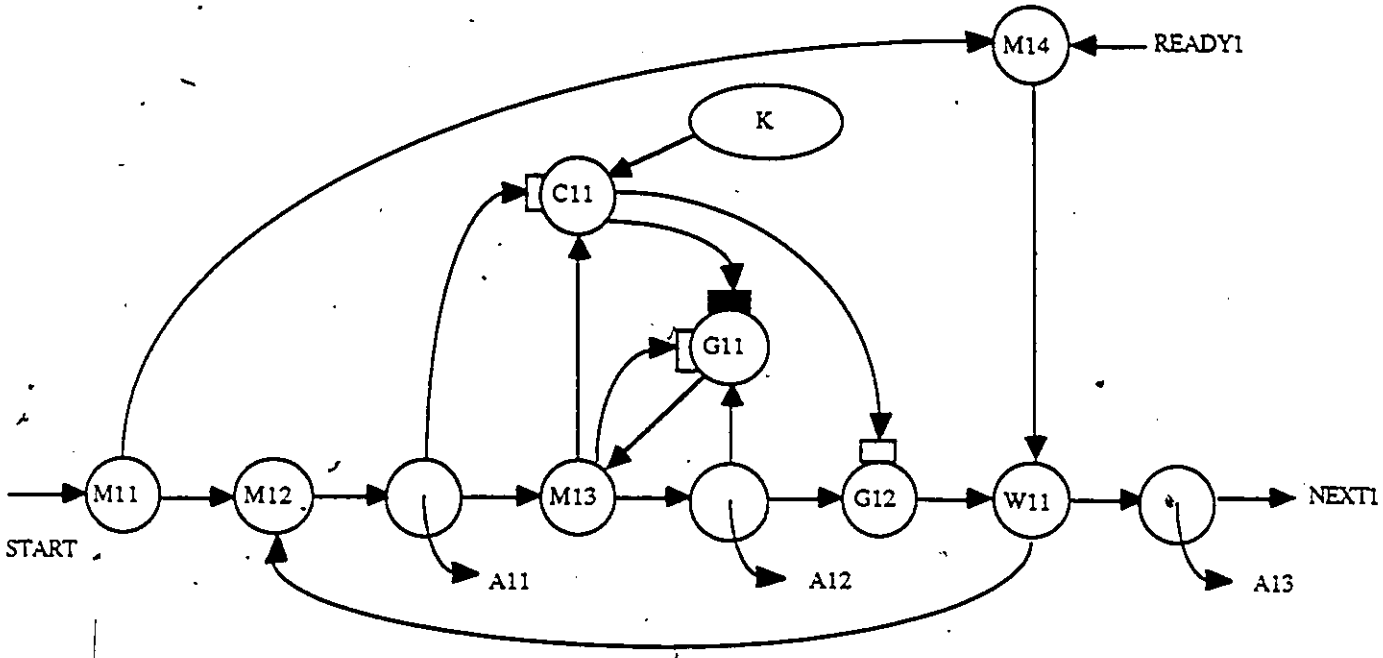
2: Axial component feeding: The axial components are fed to the FAC's insertion machine using a ribbon into which all the axial components are sequenced in

the order of installation. The insertion machine works very much like a stapler grabbing the lead on the component in the component ribbon and pushing them through their holes. Excess lead lengths are cut automatically. The ribbon is prepared in the part and components testing and preparation activity described above using data from the CAD/CAM system.

3: Insertion: Axial component insertion is done by the axial component insertion machine and the moveable fixture. Using part position data down loaded from the CAD/CAM system into local data storage, the insertion machine pushes the component leads through the board and crimps them from underneath to hold the components in place. Excess lead lengths are cut by the insertion machine. The fixture moves in the X- and Y- directions and rotates in 90 degree steps to allow the insertion of the axial components in any position on the PCB and in all orientations.

4: Sensor System: Microswitch sensors are used to detect if there exist incoming PCB's in the tray. A vision system is used in post-insertion inspection to collect data on the insertions. Quality control information with respect to axial component insertion is collected here.

5: Transfer: The central local coordinator decides if the PCB is allowed to continue in the manufacturing line or if it should be redirected to a reparation line. Bad boards are put into a bin that is discharged at regular intervals. Good boards are moved by the robot manipulator back into their



A11 = GET PCB TRAY

A12 = ASSEMBLE AXIAL COMPONENTS

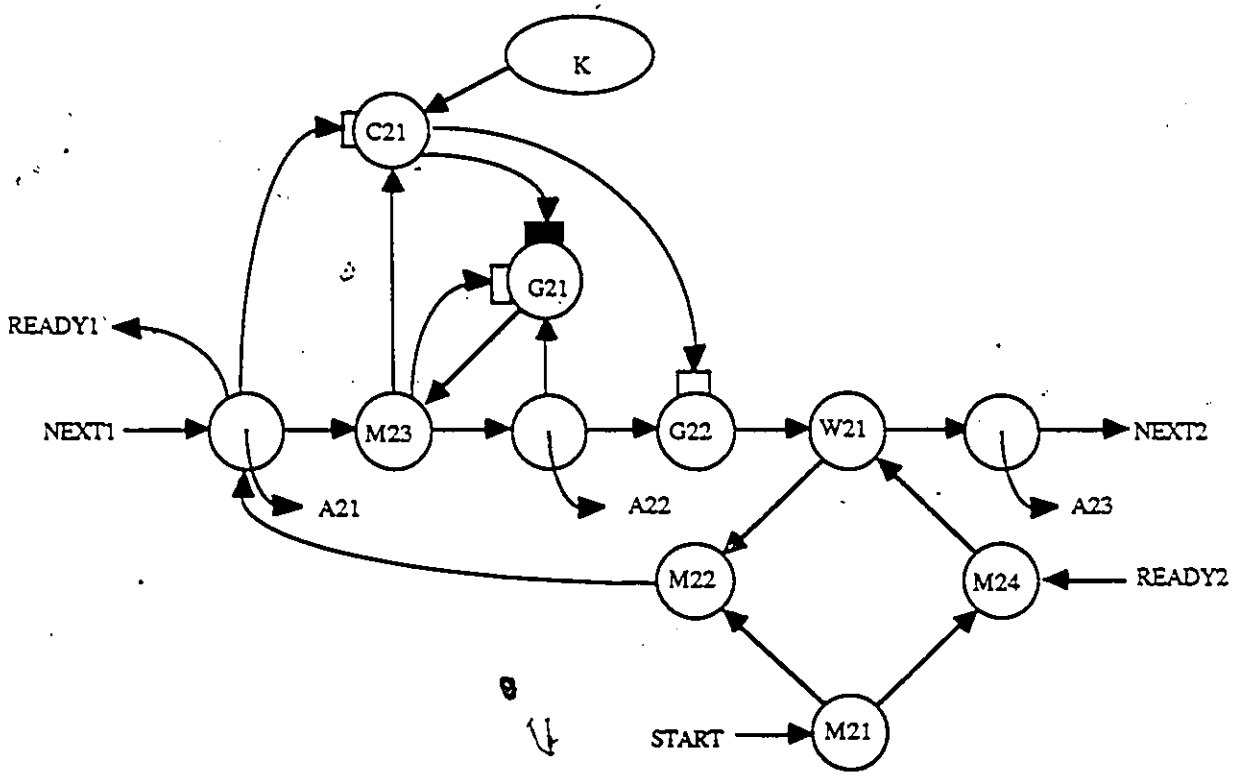
A13 = TRANSFER PCB TRAY

Figure 5-17. ASN of Axial Component Assembly.

slots in the tray in a known position; when all PCB's have been processed, the coordinator effectively issues a ready signal to the FAS controller (discussed later), indicating that the tray is ready to be sent off to the next FAC by the active conveyor transfer system element.

5.3.2.2 Box-shaped Component Assembly

Box-shaped components are more difficult to add to a PCB than axial components because the insertion task must be done with high precision and because the components usually involve the insertion of several leads or pins into several holes simultaneously. Again, it is assumed that the data used in this FAC have already been generated by the CAD system. The CAD system accesses part information from a common design and manufacturing data-base. In this type of assembly activity, a robot manipulator does the insertion using a complex algorithm and sensory feedback data used to guide the insertion. The insertion algorithm used is developed using a CAM system and simulator or test FAC. This algorithm can also be stored in the group technology area of the data-base for later use. The program implementing the algorithm is down loaded to the FAC after development testing completes successfully. Advanced feedback sensors such as vision, tactile and proximity sensors are used to measure forces and distance information during insertion. Also, the FAC's robot manipulator is equipped with a remote compliant gripper that detects and compensates for misalignment during insertion.



A21 = ACCEPT PCB TRAY FROM AXIAL COMPONENT ASSEMBLY

A22 = ASSEMBLE BOX-SHAPED COMPONENTS

A23 = TRANSFER PCB TRAY TO ODD-SHAPED COMPONENT ASSEMBLY PHASE

Figure 5-18. ASN of Box-shaped Component Assembly Subprocess.

In Figure 5-18, the ASN of the assembly of all box-shaped components to K PCB's is specified. It resembles very much the ASN of axial components. The K PCB's are presented to the FAC in a tray of K slots. Each PCB is taken one at a time and the box-shaped components are added to it. If the PCB assembly activity was done acceptably as detected by a vision-based inspection step in the FAC, the PCB is returned to its original slot; otherwise, the PCB is put in a rejection tray. When the rejection tray is full, it is sent to the repair and quality control centers for investigation. The whole operation is executed repeatedly until the tray's K PCB's are done. When all PCB's in the tray have been processed, the tray with the remaining acceptable PCB's is transferred to the next FAC for further assembly.

The behavior of the ASN for box-shaped components assembly subprocess is as follows:

1: the ASN is activated by the START trigger;

2: the START trigger activates MERGE node M21;

3: MERGE node M21 simultaneously activates MERGE nodes M22, and M24; MERGE node M21 decouples the start process from the rest of the ASN behavior;

4: MERGE node M22's output trigger makes ready ACTIVITY node A21, ACCEPT PCB TRAY FROM AXIAL COMPONENT ASSEMBLY;

5: MERGE node M24's output trigger ensures that at initial start up, WAIT node W21 is ready to respond to its other input trigger from GATE node G22;

6: when the input event NEXT1 coming from the AXIAL COMPONENT ASSEMBLY ASN occurs, ACTIVITY node A21 is initiated;

7: when ACTIVITY node A21 completes, its output triggers activate MERGE node M23, and initialize COUNTER node C21; ACTIVITY node A21 also activates the external trigger READY1, used to inform the AXIAL COMPONENT ASSEMBLY ASN that it has accepted the transferred tray and it is ready to accept another;

8: MERGE node M23's output triggers ensure that GATE node G21 is enabled while K PCB's are being assembled with box-shaped components and triggers COUNTER node C21 to count one event; another output trigger initiates ACTIVITY node A22, ASSEMBLE BOX-SHAPED COMPONENTS;

9: on every input trigger from MERGE node M23, COUNTER node C21 decreases its counter by one and tests if its limit has been reached; if it has, its output triggers are issued; if not, no triggers are issued from it;

1Ø: when ACTIVITY node A22 is completed, it sends output triggers to GATE node G21 and GATE node G22; if GATE node G21 is enabled, MERGE node M23 is activated which results in ACTIVITY node A22 being initiated again; this means that repeated initiation of activity A22 is carried out; with COUNTER node C21 controlling GATE node G21, there will be K activations of activity A22 for K PCB's in a PCB tray; since MERGE node M23 is activated, triggers are made active on COUNTER node C21, and on the enable input of GATE node G21 to re-enable it since GATE nodes disable themselves after issuing output triggers; since GATE node G22 is disabled for the first K-1 initiations of ACTIVITY node A22, the input

trigger from ACTIVITY node A22 is ignored;

11: when the limit of COUNTER node C21 has been reached, it activates its output triggers; one of its output triggers disables GATE node G21, with the result that this will stop any further initiation of ACTIVITY node A22; disabling GATE node G21 means that K PCB's have been assembled with the axial components; the other output trigger enables GATE node G22;

12: when ACTIVITY node A22 completes for the K-th time, its output trigger to GATE node G22 is recognized since it is enabled; at this point, GATE node G22's output trigger is sent to WAIT node W21 whose other input trigger had already been activated by MERGE node M24;

13: WAIT node W21 activates its output triggers which initiate ACTIVITY node A23, TRANSFER PCB TRAY TO ODD-SHAPED COMPONENT ASSEMBLY PHASE and activate MERGE node M22, making the whole BOX-SHAPED COMPONENT ASSEMBLY subprocess ready again;

14: When ACTIVITY node A23 completes, output trigger NEXT2 is activated; this is an input trigger for the next phase of the PCB assembly process, ODD_SHAPED COMPONENT ASSEMBLY, represented by the ASN in Figure 5-19;

15: input trigger READY2 originates from the ODD-SHAPED COMPONENT ASSEMBLY subprocess ASN, and indicates that the transferred PCB tray has been accepted; READY2 activates MERGE node M24 which activates one of the input triggers to the WAIT node W21; combination of nodes M24, W21, A23, and triggers NEXT2 and READY2 establish the coordination required between the BOX SHAPED COMPONENT ASSEMBLY and ODD-SHAPED COMPONENT

ASSEMBLY subprocess ASN's (or assembly phases); this specific realization of the ASN for the BOX-SHAPED COMPONENT ASSEMBLY phase allows the next tray of K PCB's to be assembled with its box-shaped components while the just finished tray is being transferred to the ODD-SHAPED COMPONENT ASSEMBLY phase; also, the ASN does not permit the transfer of the next tray until the previous tray has been accepted by the following phase (see ODD-SHAPED COMPONENT ASSEMBLY Subprocess ASN);

From the above ASN, the following requirements and needed elements can be identified for the BOX-SHAPED COMPONENT ASSEMBLY FAC implementing the activity of ACTIVITY node A22:

1: PCB feeding: PCB's are taken from the tray in a known position using a robot manipulator and sensory system. The PCB is latched into place on an immobile fixture using fixture pins and associated manufacturing holes on the PCB, and bracing springs. Each PCB has a reference hole from which all positioning information is related.

2: Box-shaped component feeding: The box-shaped components are fed to the FAC using compartmentalized trays at known feeding places. Each compartment contains one component. A robot manipulator is used to get the components. Dimension information on the components is provided obtained by the CAD/CAM system and is down-loaded to the FAC's controller. When a tray of components is exhausted, another is provided by the feeder. There are three box-shaped component feeders for the three different box-shaped compo-

nents: IC1, IC2, and TRANSFORMER.

3: Insertion: Box-shaped component insertion is done by the robot manipulator equipped with a remote compliant gripper. The robot manipulator will be used to insert the box-shaped components by positioning them and orienting them properly using component position data down-loaded from the CAD/CAM system into local data storage of the FAC and a vision system. The robot manipulator gets each component one at a time, moves it to the appropriate insertion position at the PCB in the correct orientation, and pushes the component's leads through the board with a specific limit force.

4: Sensor System: Microswitch sensors are used to detect if there exist incoming PCB's in the tray. A vision system is used to monitor and provide feedback to control the robot's movements. Tactile feedback sensors on the gripper provide the needed feedback information to control the insertion. A vision system is used in post-insertion inspection to collect data on the insertions. Quality control information with respect to box-shaped component insertion is collected.

5: Transfer: The FAC's coordinator decides if the PCB should be allowed to continue in the manufacturing line or if it should be redirected to a reparation line. Bad boards are put into a bin that is discharged at regular intervals. Good boards are moved by the robot manipulator back into its slot in the tray in a known position; when all PCB's have been

processed, the PCB tray is sent off to the next FAC by the active conveyor transfer system.

From the above 5 requirement points, the following FAC elements needed to execute the BOX-SHAPED ASSEMBLY activity can be identified: robot manipulator, remote compliance gripper end effector, fixture, microswitch and tactile sensors, vision system, large data storage for part information and data collection, three part feeders for IC1, IC2, and TRANSFORMER.

5.3.2.3 Odd-Shaped Component Assembly

Theses objects are the most difficult to assemble because each object to be used in the assembly may be different and irregular in shape. To permit assembly, custom end effectors in the form of grippers or end tools may have to be built for the different task and object to be manipulated during assembly. This requires more advanced robot technology. The assembly activities often require a fastening type of operation where one object is fastened to the board with screws and nuts, or one object must be held in place while another is attached in some way. Object information is provided by the CAD/CAM system through the common data-base. As a rule, a new FAC program must be created for every new end effector used and for every differently shaped object to be assembled.

In Figure 5-19, the ASN of the assembly of all odd-shaped components of K PCB's is specified for completeness. Since it is

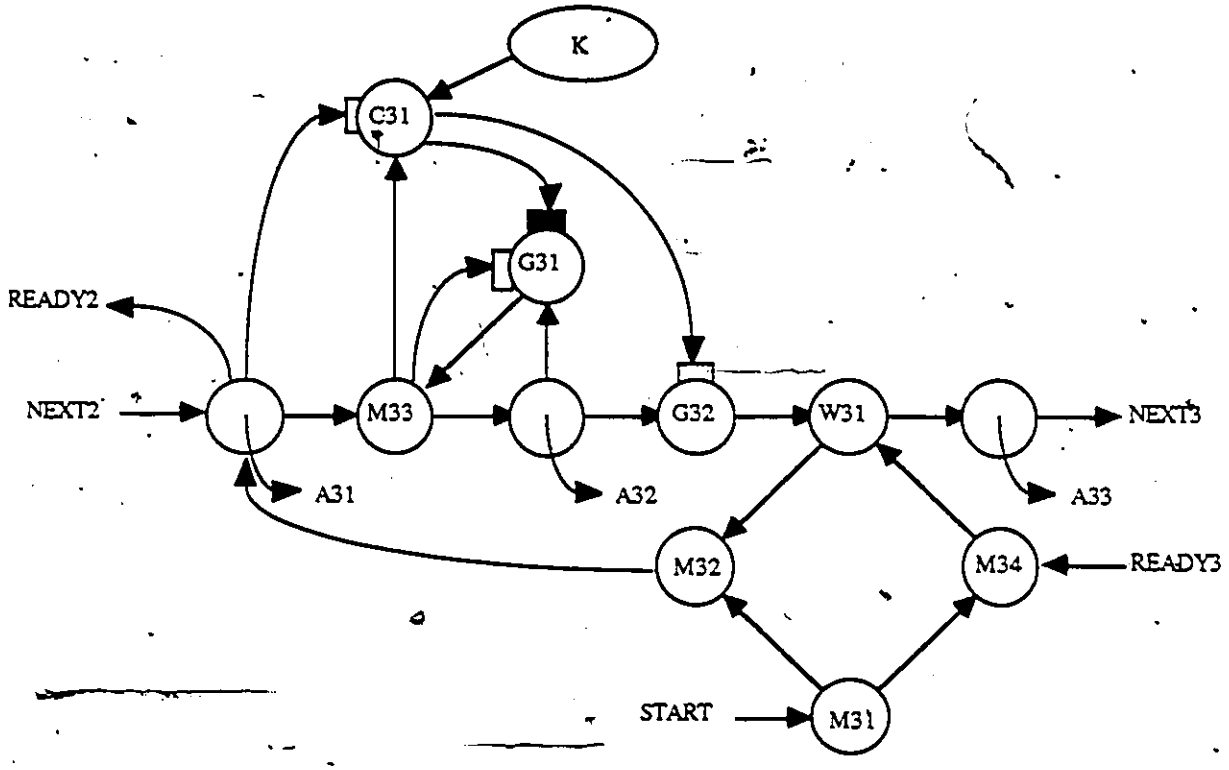


Figure 5-19. ASN of Odd-Shaped Component Assembly Subprocess.

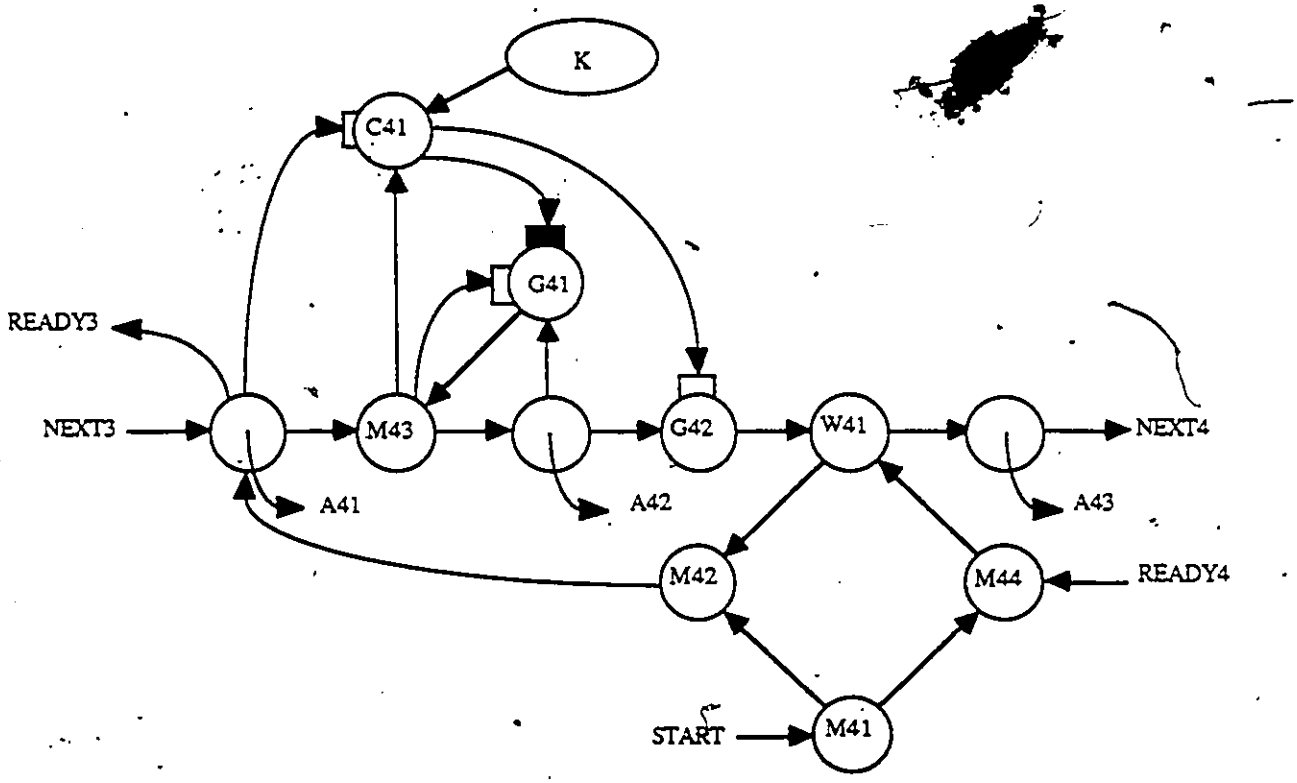


Figure 5-20. ASN of PCB Soldering Subprocess.

identical to the ASN of box-shaped components assembly except in the activities specified at ACTIVITY nodes A31, A32, and A33, its behavior will not be described here. The K PCB's are presented to the FAC in a tray of K slots. Each PCB is taken one at a time and the odd-shaped components are added to it. Similarly to the other component assembly activities, if the PCB odd-shaped component assembly activity was done acceptably, as detected by a vision-based inspection step in the FAC, the PCB is returned to its original slot in the tray; otherwise, the PCB is put in a rejection tray. When the rejection tray is full, it is sent to the repair and quality control centers for investigation. The whole operation is executed repeatedly until the tray's K PCB's are done. When all PCB's in the tray have been processed, component assembly is then complete and the tray with the acceptable PCB's is transferred to the next cell for soldering.

5.3.2.4 PCB Component Soldering

The PCB's with the added components are then soldered using a PCB wave soldering machine. This process is very simple: PCB's are pulled with their pin side momentarily skimming the surface of liquid solder in a hot tub and the exposed pins are soldered to their soldering points.

In Figure 5-2Ø is shown the ASN for the soldering of K PCB's selected from a tray. All soldering is done by a wave-soldering machine. It accepts PCB's and pulls them over a hot tub of liquid solder to solder all exposed pins on the soldering side of the PCB.

5.3.3 FAC Architecture and Programming

In this section, the programming and coordination of the FAC elements required for the assembly of the box-shaped components is presented as a example. From the requirements specification given in the previous section, the following elements are required in the FAC:

Robot manipulator:	used to manipulate the PCB, and the components;
Remote compliance gripper:	used to grab the PCB and box-shaped components, and to push the components's pins into their holes;
Fixture:	to hold the PCB during assembly;
IC1 Part Feeder:	feeds the FAC with IC1 components;
IC2 Part Feeder:	feeds the FAC with IC2 components;
T1 Part Feeder:	feeds the FAC with TR1 components;
Tray Transfer Conveyor Unit:	receives trays from the transfer system and transfers trays to the next FAC;
Tactile Sensors:	give force feedback information by the gripper;
Vision Sensor:	give image feedback used to identify objects, components and destinations visually, to analyze image data during inspection, etc.
Microswitch Sensors:	to detect the presence of PCB's in the tray;
Assembly Data Storage:	information storage in which component information can be obtained, and assembly result information can be collected;

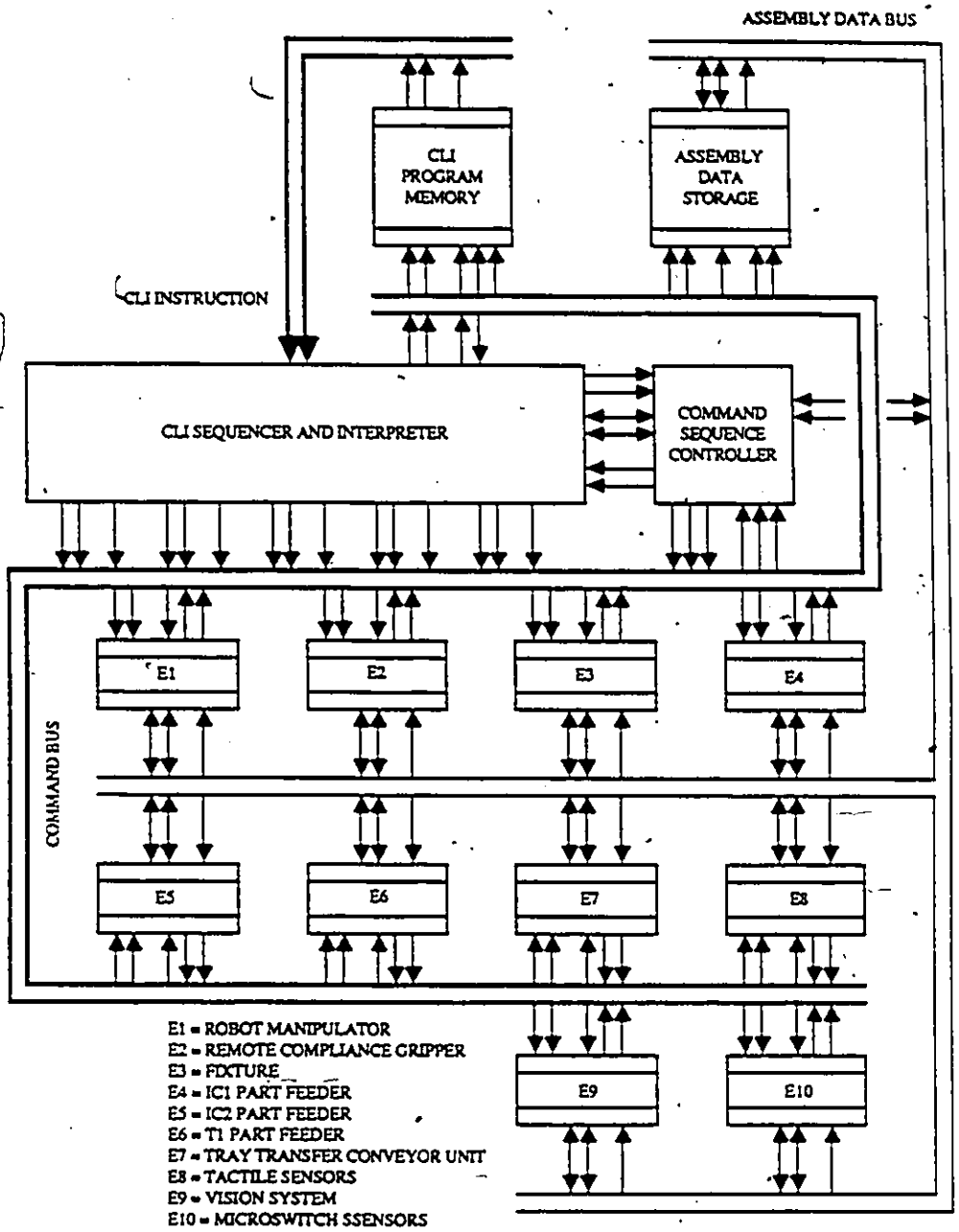


Figure 5-21. FAC Organization for Box-shaped Component Assembly

The FAC organization required for the assembly of the box-shaped components described by activity of ACTIVITY node A22 is shown in Figure 5-21. The CLI program needed to perform the assembly activity is given below.

The main portion of the CLI program that implements the activity of node A22 expressed in a machine language 'flavor' for the described FAC is:

```
INITIALIZE      ELEMENTS
MOVE            FIXTURE,          PCB
INSERT          IC1_PINHOLES,     IC1
INSERT          IC2_PINHOLES,     IC2
INSERT          T1_PINHOLES,      T1
MOVE            TRAY,             PCB
```

The level of expression of these instructions is identical to the task level specification found in the literature [LOZA83]. To implement the instructions as expressed, they must be divided into a set element commands executed in the proper sequence. These commands are implemented as separate element programs or subprograms. The sequence of element command execution must be defined to the command sequencer. The following breakdown of the MOVE and INSERT instructions shows the set of commands to be implemented as subprograms for the different elements and shows the required sequence of these element commands to implement the given CLI.

For MOVE FIXTURE, PCB:

1:VISION,MICROSWITCH SENSORS:	IDENTIFY PCB
2:ROBOT,VISION SENSOR:	MOVE END EFFECTOR TO PCB
3:ROBOT,GRIPPER,TACTILE SENSORS:	GRAB PCB WITH GRAB FORCE
4:VISION SENSOR:	IDENTIFY FIXTURE
5:ROBOT,VISION SENSOR:	MOVE END EFFECTOR TO FIXTURE
6:ROBOT,GRIPPER,TACTILE SENSORS, FIXTURE:	PUSH PCB ONTO FIXTURE WITH PCB PUSH FORCE
7:GRIPPER:	RELEASE PCB
8:ROBOT:	MOVE TO NEUTRAL POSITION
9:VISION SENSOR:	VERIFY PCB IN FIXTURE

For INSERT IC1_PINHOLES, IC1:

1:VISION SENSOR:	IDENTIFY IC1
2:ROBOT,VISION SENSOR:	MOVE END EFFECTOR TO IC1
3:ROBOT,GRIPPER,TACTILE SENSORS:	GRAB IC1 WITH GRAB FORCE
4:VISION SENSOR:	IDENTIFY IC1_PINHOLES
5:ROBOT,VISION SENSOR:	MOVE END EFFECTOR TO IC1_PINHOLES
6:ROBOT,GRIPPER,TACTILE SENSORS:	PUSH IC1 INTO IC1_PINHOLES WITH IC1 PUSH FORCE
7:GRIPPER:	RELEASE IC1
8:ROBOT:	MOVE TO NEUTRAL POSITION
9:VISION SENSOR:	VERIFY IC1 IN PCB

From the above instruction implementations, several instruction 'phases' can be identified. One or more low level command primitives may need to be performed by a corresponding number of different FAC elements in each phase of the CLI's execution. This means that more than one FAC element may have to work together, as in phase 6, to execute the command. In this case, the ROBOT is commanded to push IC1 found in the GRIPPER (used to prevent jamming and keep the alignment for insertion), into its pinholes using feedback force information from the TACTILE SENSORS until the force limit data found with the TACTILE SENSORS. The TACTILE SENSORS element is activated to detect the stopping conditions while the ROBOT and GRIPPER perform the push operation.

To permit the needed interaction between the FAC elements, a means by which the the TACTILE SENSORS element can inform the ROBOT and GRIPPER element controllers that stopping conditions have been reached must be provided. The message can be sent in the form of a simple dedicated signal to the ROBOT element interface unit from the TACTILE SENSORS element interface unit, or by using a shared communication media under control of another element in the system that acts as a communication arbitrator, also under the control of the command sequencer. The function of the arbitrator is to ensure the required control and sequencing for communication between the FAC element interface units, for the execution of each phase of the instruction.

After the command sequences have been identified, the necessary set of interface commands to be implemented in the separate FAC element interfaces are compiled. These interface commands are further implemented as sequences of primitive commands that are sent by the interface to the element controllers. Each primitive command of an element is implemented as an element controller program specifically designed to execute the needed primitive.

Therefore, a CLI is a sequence of FAC element commands; a command is a sequence of FAC element command primitives; and an element command primitive is single element program. This means that to execute one CLI, several element programs may needed to be executed, and several FAC elements may be activated concurrently.

When all commands have been executed under the control of the command sequencer successfully, the command sequencer informs the CLI sequencer which fetches the next CLI to be executed. If the command sequencer detects that a command failed to execute properly using internal timers for each phase, or from error signals from the FAC elements themselves, the command sequencer stops the execution of all other elements, aborts the remaining commands, and informs the CLI sequencer of the status of the current CLI being executed. Status information consists of: the element that failed, the current command phase being executed, and possibly status data from the elements. Since a failure handling procedure is defined for each instruction command phase, the CLI sequencer is able to decide which failure procedure to invoke and use the status information provided by the command sequencer.

5.3.4 The Functional Organization of the FMS Controller

In this section, the organizational alternatives of the FMS controller are considered. The organization of the controller depends on the manufacturing scheme to be implemented: single or multiple lot schemes. In single lot manufacturing, only one lot of a certain size of a single PCB design is manufactured by the FMS at one time. In multiple lot manufacturing, several lots different PCB designs are manufactured concurrently by the same FMS.

Within single lot and multiple lots manufacturing schemes of size **N** in batches of **K** of PCB's, different scenarios can be envisaged.

Single Lot Manufacturing Scheme

In single lot PCB manufacturing, the FMS is setup only once and all resources are used only to manufacture one PCB design. The resources cannot be used to manufacture a different PCB until all N of the current PCB design have been manufactured. In the single lot scheme, there are two main approaches:

1: A first approach to FMS controller design is based on the a single monolithic ASN, composed of composite activities, that realizes the whole manufacturing process. A PAD describing this situation was shown in Figure 5-10 using composite activities describing the manufacturing subprocesses which are executed for K PCB's. All the resources needed to manufacture the batch of K PCB's are allocated once. In this case, the FMS controller is simply a single ASN MANAGER, shown in Figure 5-22, and is governed by a single matrix representation of the ASN. The tasks of the ASN MANAGER will be discussed later.

This approach is obviously inefficient in the use of resources and in terms of throughput and resource utilization because potentially only one FMC of the set of FMC's in the whole system can be executing, and most FMC's in the whole FMS could be sitting idle most of the time. A composite activity may also have to wait until all necessary composite activities previous to it are completed for the next batch of K PCB's, before it can be executed. Also, as soon the currently executing composite activity finishes for the current

batch of K PCB's, it must wait until the all subsequent composite activities have been completed for this batch. The throughput is determined by the slowest component.

2: In a second approach, the monolithic ASN is partitioned into separate ASN's, one for each major subprocess. These are defined such that they can proceed immediately onto the next batch of K PCB's after the processing for the current batch of K PCB's is completed. This is done I times to give N PCB's. An example showing the processing of the next batch of K PCB's as defined in one ASN while the just completed batch is being processed by the next ASN is shown in ATC of Figure 5-24. In this case, each ASN is controlled by its own ASN MANAGER, and all the ASN MANAGERS in the system are sequenced by an ASN SEQUENCER, as shown in Figure 5-23. The ASN SEQUENCER is responsible for initiating ASN's of the FMS in the proper sequence in response to the start up event and subprocess completion signals from the ASN's.

In this approach, there is more efficient use of all resources in the FMS because there is a lot of overlap between the various execution phases. Throughput increases because the manufacturing phases of K PCB's are interleaved. Furthermore, the throughput of a previous phase can be matched to that of the next phase by allocating more resources (FMC's) to both phases in the correct ratio. This will help remove the bottlenecks that exits between phases.

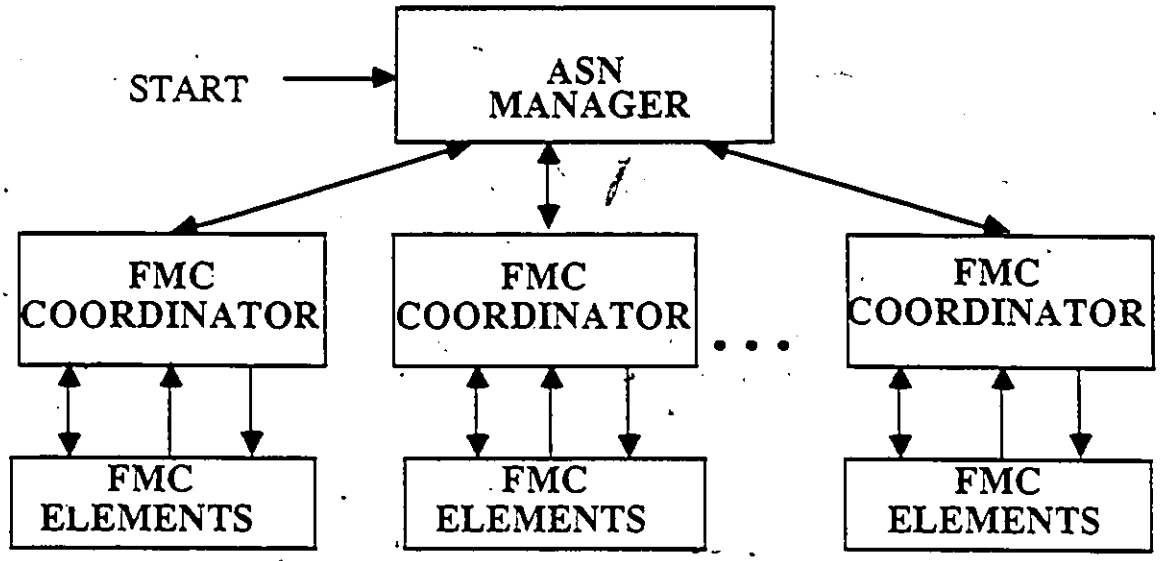


Figure 5-22. Single ASN MANAGER as FMS Controller.

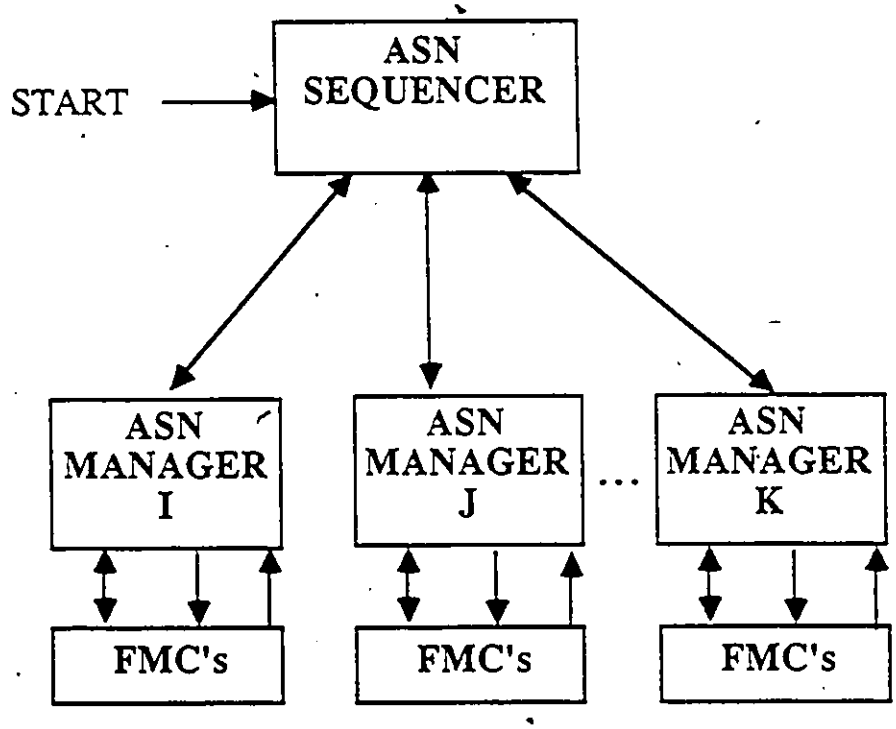


Figure 5-23. Partitioned ASN FMS Controller.

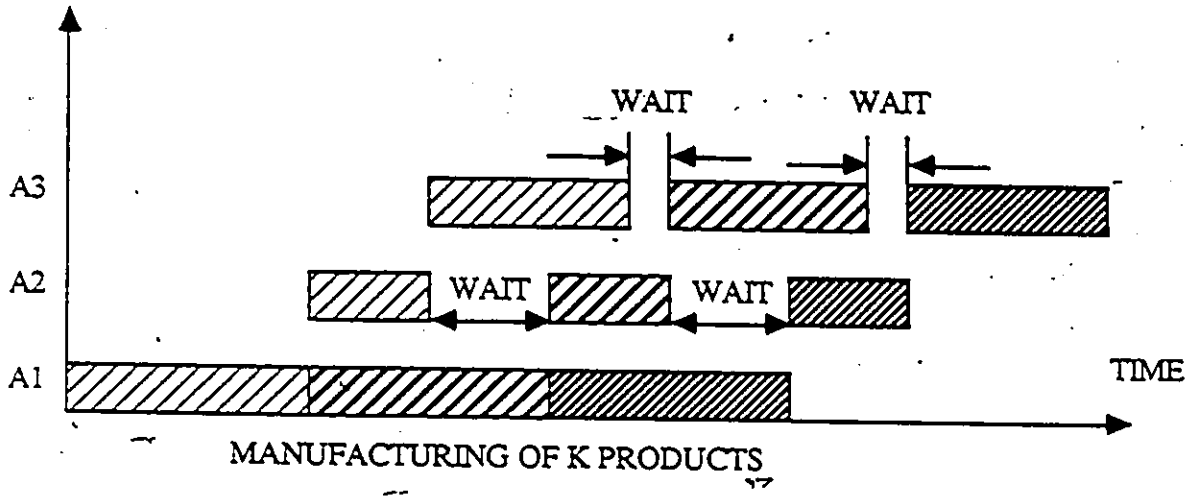


Figure 5-24. ATC of Partitioned ASN Showing Overlap.

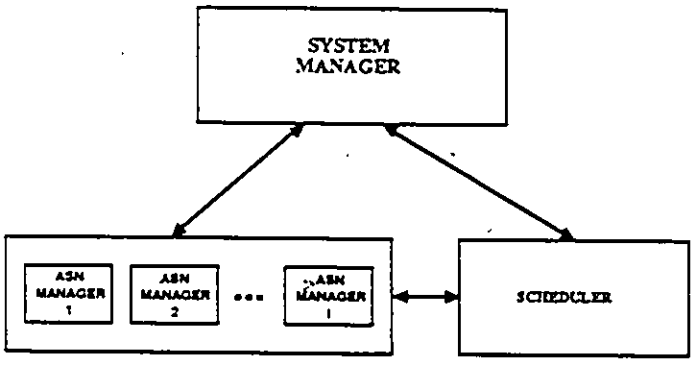


Figure 5-25. FMS Controller For Multiple Lot Manufacturing Scheme.

Multiple Lot Manufacturing Scheme

In multiple lot manufacturing, more than one PCB design is being manufactured at a time by the same FMS. This means that some resources can be allocated to execute ASN's for one specific PCB while the same ASN is being executed on similar resources for a different PCB. In this way manufacturing phases for different PCB's can be overlapped in the same FMS. If the execution time of an ASN for a first PCB design is longer than for a second design, then the same ASN could be executed several times for the second PCB design before the ASN is executed a second time of the first PCB design. Also, intermixing of different PCB designs lots can be done by using multiple resources for an ASN or, by providing highly reconfigurable FMC's in the FMS which can be used to execute different activities for different PCB designs.

To permit this kind of resource allocation, both a SYSTEM MANAGER and an ACTIVITY SCHEDULER are required (see Figure 5-25) to assign FMC's to execute the activities to be run for the different PCB designs, as determined by the ASN MANAGER. The SYSTEM MANAGER is required to sequence the ASN's in the proper order, and to initiate new instances of an ASN MANAGER to execute an ASN for another PCB design, or for another batch of the same PCB design. The ACTIVITY SCHEDULER is responsible for scheduling activities on the FMC's of the FMS in response to requests from the ASN MANAGER's.

In such a FMS, activities may be bound to particular FMC's. This would limit their ability to be allocated to execute other

activities for different PCB designs and therefore limit the resource utilization. Alternately, activities might be executable on several FMC's, allowing for better resource allocation and providing for better resource utilization. It is important that the ACTIVITY SCHEDULER to be aware of these limitations to have orderly assignment of FMC's in executing activities. Resource allocation for concurrently manufactured products becomes very complex and requires the use of techniques developed for job shop scheduling [BAKE74].

5.4 FMS Architecture and Functional Organization of Controller

There are two basic approaches to FMS control: distributed control in which the interconnected FMC's share information and control the whole FMS, and central control in which a dedicated controller manages the FMS based on the information received from the independent FMC's and other systems. Central control may be organized hierarchically, with many levels of control where each controller communicates with its peers, immediate superior and subordinates.

PAD's are best suited to the design of centrally controlled FMS's. The general architecture for a centrally controlled FMS is shown in Figure 5-26. At the lowest level of the FMS are found the FMC's which execute the activities of the assembly process. The CL programs for the activities are stored in the FMC's responsible for those activities. To execute a particular activity, the central FMS controller informs the proper FMC to start executing the required activity. When an activity is completed, the

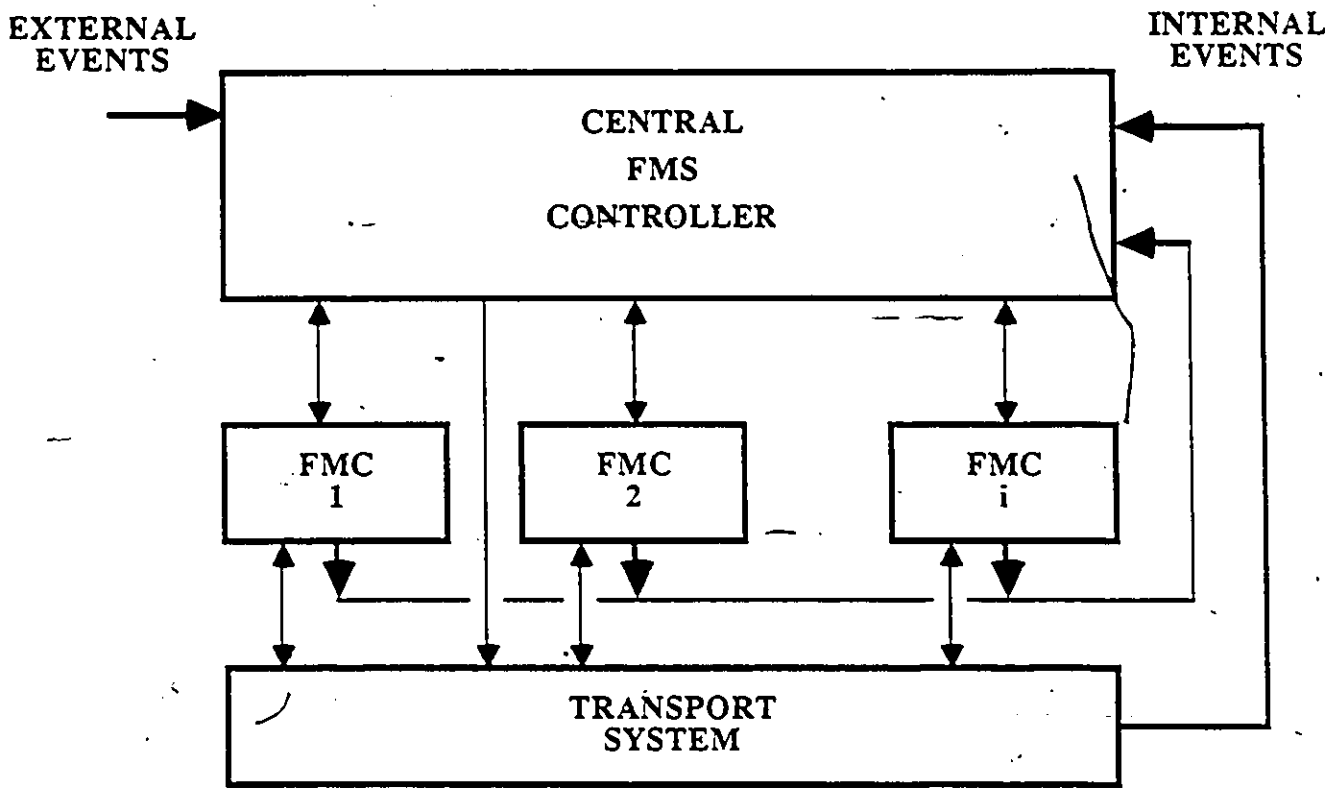


Figure 5-26. Centrally Controlled FMS Architecture.

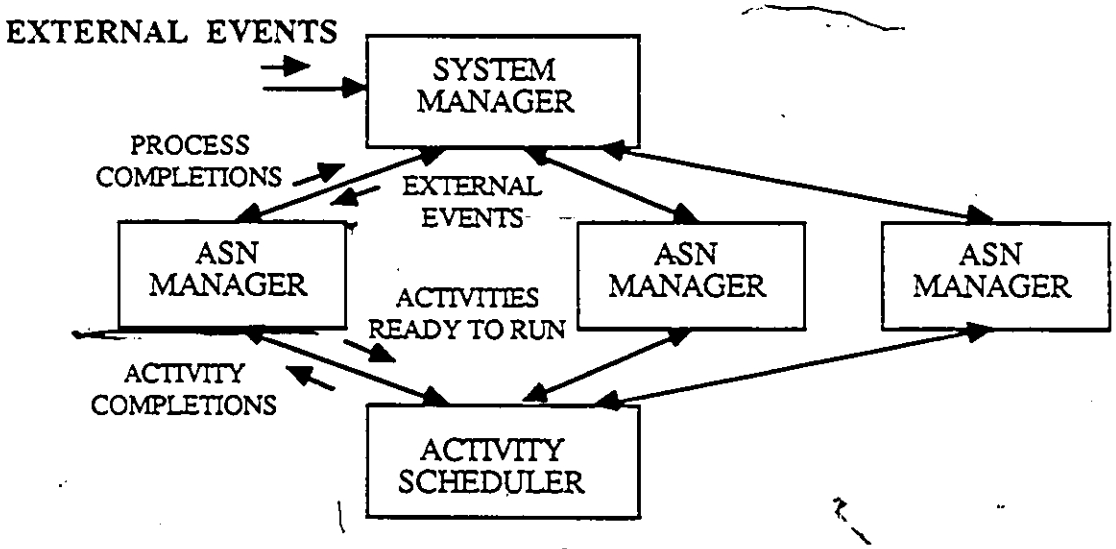


Figure 5-27. Functional Organization of the FMS Central Controller

FMC informs the central FMS controller. A control bus is used by the central controller to initiate FMC activities execution at the required time and sequence, and to receive acknowledgement and completion responses. A data bus is used to transfer data to and from the FMC's.

As shown in the above discussion on FMS scenarios, the central FMS controller must react to the external environment, and achieve proper control and coordination of the activities in the FMS. To achieve this, the general functional organization of the FMS central controller's must contain three main layers shown in Figure 5-27.

1: The SYSTEM MANAGER: It has general knowledge of the state of the system by keeping track of external events from the FMS environment or from system activities, of awakened (executing) ASN's in the system, and of process completion signals from the ASN MANAGERS. It is responsible for initiating ASN's of the FMS in response to external events by 'creating' ASN MANAGERS for the appropriate ASN, and transmitting the appropriate start up trigger to the ASN MANAGERS. It may also transmit any external event for which an awakened ASN may be waiting. External events may be ignored by the SYSTEM MANAGER or it may initiate a new instance of an ASN MANAGER, depending upon the type of FMS control system being designed. Upon receiving a process completion signal from an ASN MANAGER, the SYSTEM MANAGER 'kills' the ASN MANAGER.

2: The ASN MANAGERS: ASN MANAGERS are dynamically 'created' for each awakened ASN, and are dynamically 'killed' when the ASN's execution is completed. In a working system, there exists only one ASN MANAGER per ASN, and different instances of the awakened ASN's are represented internally to the controller. The operation of an ASN MANAGER is governed by the matrix representation of the ASN. ASN MANAGERS carry out the proper activity sequencing specified by the ASN's for which they are responsible by receiving external events from the SYSTEM MANAGER, and activity completion signals from the ACTIVITY SCHEDULER. Using this information, the ASN MANAGER checks for activities that have to be executed, and transmits this information to the ACTIVITY SCHEDULER.

3: The ACTIVITY SCHEDULER: It assigns FMC's to execute the activities to be run as determined by the ASN MANAGER. In this way it is responsible for scheduling activities on the FMC's of the FMS. In a FMS, the ACTIVITY SCHEDULER must be aware that some activities may be bound to particular FMC's, or they may be executable on several FMC's.

As shown in [JOAN86], an ASN MANAGER uses a simple control algorithm (see Figure 5-28). In a working system, this algorithm can be implemented by a dedicated processor (or microprocessor). Also, because of the differences between the time it takes for activities to execute and the time required by an ASN manager to make a decision concerning the sequencing of activities, several

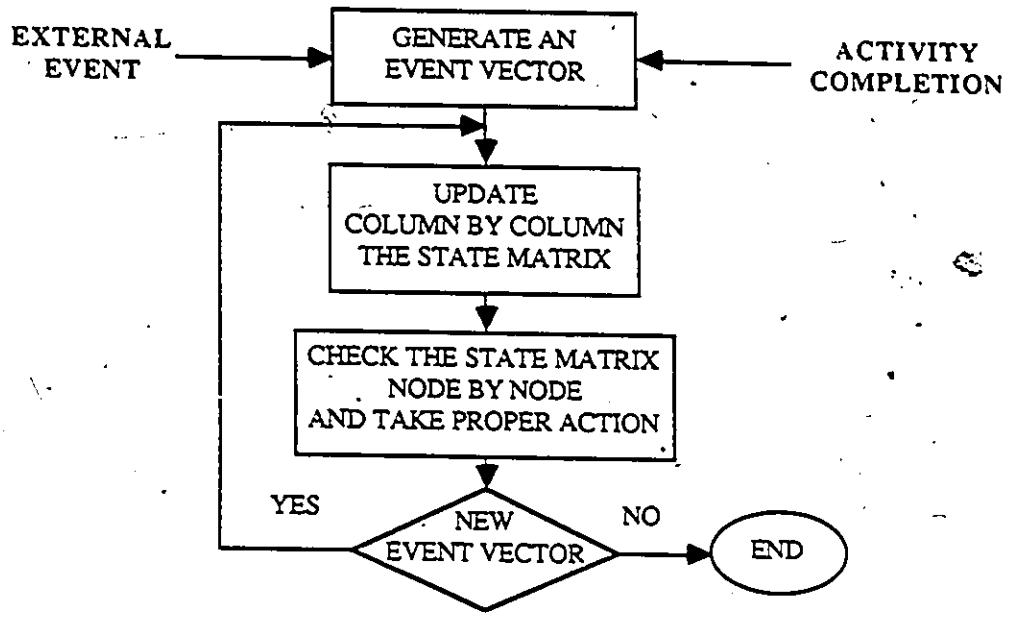


Figure 5-28. ASN MANAGER Operation.

ASN MANAGERS can be implemented on a single microprocessor, and executed in a time shared fashion.

In a working FMS, the production rate of the system may need to be increased. This can be achieved by allowing the ACTIVITY SCHEDULER to allocate more FMC's to the execution of the activities of the higher priority manufacturing process. Also, production may be slowed down, but not shut down, to permit plant servicing of FMC's and allow the introduction of new FMC's into the FMS without impeding production.

Since some PCB manufacturing elements used in FMC's are very specialized, and since FMC's should be kept as simple as possible, FMC's are usually only able to execute one activity. With this in mind, resource allocation in FMS's is not as free as in multiprocessor systems. This means that a FMC can only be used in specific phases of the manufacturing, to execute a single activity, limiting its ability to be used to execute other activities in completely different areas of the FMS without redesigning the FMC.

5.5 Conclusions

In this chapter, PAD's were introduced as a graphical tool used to define complex processes. An example was presented and showed the applicability of PAD's to the design of Flexible Manufacturing Systems (FMS). It was shown that the manufacturing process can be defined in terms of Activity Sequence Nets (ASN) that describe manufacturing subprocesses in a step-wise refinement manner. From these ASN's, the required set of manufacturing

activities can be identified and can be mapped onto the required sets of Flexible Manufacturing Cells (FMC) that are used to implement the various subprocesses. It was also shown that for each FMC activity to be executed, the elements required can also be identified. Using the methodology presented in chapter 4 for FMC design, these elements can be integrated into a programmable and coordinated FMC. Furthermore, it was shown that by defining the manufacturing subprocesses using ASN's, the sets of FMC's can be integrated into a FMS.

Integration can be done in different ways depending on production requirements. The FMS for single lots schemes can be controlled by a single ASN MANAGERS, or a number of ASN MANAGERS and a ASN SEQUENCER. This type of FMS controller is applicable to large production run FMS's with few change-overs. The FMS multiple lots schemes require more complex controllers and require FMC's that can be dynamically reconfigurable to to allow them to execute activities for different lots. Multiple lots schemes are more applicable to FMS's with smaller production runs in which change-overs occur often and/or to FMS's in which a number of different products are manufactured simultaneously.

Because timing data about actual manufacturing activities for PCB manufacturing could not be obtained, the full potential of PAD's, specifically including ATC's, was not completely demonstrated to show how the best throughput could be found, or what could be the best resource allocation. By using the activity execution times included in the ATC's, the manufacturing engineer

can verify the timing requirements of all the production steps at the design stage. In this way, performance bottlenecks can be determined and avoided by selectively improving activity executions using higher performance FMC's where needed, by allocating more resources to the execute the activities in tandem, or by partitioning if possible, a complex FMC activity among several FMC's.

In large production runs, the flow-through can be maximized by using or allocating a number of resources at the bottleneck points. In smaller production runs, a method similar to job-shop scheduling can be used to allocate resources in an optimal way for different products. Both the above aspects require more detailed investigation. For simple systems with few resources, it is believed that PAD's can be used to do reasonable job-shop scheduling with some effort. For more complex systems, more advanced techniques and deeper analysis on dynamic resource allocation must be done, and is beyond the scope of this thesis. A special case of such an analysis is the effect of changing product priorities, in which a product in the line gets a higher priority and must be manufactured in the needed lot size ahead of the original delivery schedule.

Chapter 6

Conclusions and Further Research

In this thesis a methodology for integrating manufacturing elements into a Flexible Manufacturing System (FMS) was presented. The approach allows low production run FMS's to be designed without expert personnel using PAD's. PAD's allow the manufacturing engineer to specify the manufacturing process in a top down fashion by successive refinement using Activity Sequence Networks (ASN), a component of PAD's, consisting of a small number of easy to learn elements. ASN resemble flow diagrams which are familiar to manufacturing engineer and therefore can be easily learned and used to define manufacturing processes.

Once the manufacturing processes are defined in terms of ASN's, the set of Flexible Manufacturing Cells (FMC) required to implement the necessary machining and assembly activities can be identified. A methodology was presented to integrate the manufacturing elements of the FMC's into a programmable unit, based on the shown mechanical equivalence of an FMC to a computer. The method allows elements from different sources to be integrated into a coordinated FMC that can be programmed at the task level. To this end, a cell level language with failure handling capabilities was proposed.

With the aid of ASN's, FMC's can be integrated into a coordinated FMS that implements the manufacturing process for different manufacturing schemes and requirements.

Further Research

This thesis only considered the feasibility of the methodology presented, and several notions and ideas must be transformed into a proper toolset.

PAD Editor

An editor must be developed to capture PAD's in graphical form. This editor must have access to design and manufacturing data from a common data-base. There exists a simple matrix editor [DURA87] for entering and modifying ASN's which also provides limited syntax checking. This editor is expected to be interfaced with a symbolic graphical editor, SYMED [BURG86], developed as part of the Advanced Real Time Tools (ARTT) project.

Simulator

Furthermore, a full fledged simulator must be developed which can help to verify the various defined processes, and which check for possible bottlenecks. The simulator must also allow the user to explore better allocation of resources. A basic simulator was developed in [JOAN86]. To make the simulator more applicable to FMS design, a more complex scheduler must be developed that allows different scheduling algorithms to be evaluated. Such a# simulator should also allow the investigation of the effect of buffering in the product transfer system.

Programming

Another area of interest that must be explored is that of FMC programming. A more product oriented language could be developed and various automated programming techniques considered. Also, there is a need to design a more general purpose FMC coordinator that can be easily integrated with existing element controllers. A possible architecture could be based on new interface standards such as MAP. Also the notion of using object oriented programming approaches to FMC programming must be investigated. The objects themselves could be defined with their 'methods' which describe, for example, how they can be mated to other objects. The whole concept of 'inheritance' may also have a far reaching impact in terms of group technology.

In conclusion, this thesis presented a top-down approach to FMS specification and coordination that also permits straightforward product change-over. This approach is applicable to various production run sizes and to multiple product manufacturing.

Bibliography

ABRA77 Abraham, R.G. 'A.P.A.S.: Adaptable Programmable Assembly-System.' In Robots and Computer Vision, pp117-140.

AGIN80 Agin, G.J. 'Computer Vision Systems for Industrial Inspection and Assembly.' In IEEE Computer. Vol. 13, No. 5, May 1980, pp11-20.

ALBU76 Albus, J.S., Evans Jr., J.M. 'Robot Systems.' In Scientific American. Vol. 234, No. 2, Feb. 1976, pp767, 20, 140.

ALBU79 Albus J.S., Barbera, A.J., Fitzgerald, M.L. 'Hierarchical Control of Robots Using Microcomputers.' In Proc. of the 9th Int. Symp. on Indus. Robots. Soc. of Manuf. Eng., 1979, pp405-422.

ALBU80 Albus, J.S., et al. 'A Measurement and Control Model For Adaptive Robots.' In Proc. of the 10th I.S.I.R., Soc. for Manuf. Eng. 1980, pp303-312

ALBU81 Albus, James, S. Brains, Behavior, And Robotics. North Quincy, Mass.: BYTE Publications Inc., 1981.

ALBU81a Albus, J.S., Barbera, A.J., Fitzgerald, M.L. 'Hierarchical Control For Sensory Interactive Robots.' In Proc. of the 11th I.S.I.R. Soc. of Manuf. Eng. 1981, pp497-505.

ALLA82 Allan, R. 'Industrial Electronics: Robotics Comes Into Its Own With Improved Vision, Sense Perception, Teaching Languages.' In Electronic Design. April 1982, pp87-100.

AKEL85 Akella, R., Bevans, J.P., Choong, Y. Simulation of a Flexible Manufacturing System. Laboratory for Information and Decision Systems, MIT, Cambridge, MA, Report 6-50-85, March, 1985, LIDS-P-1435.

AUSL78 Auslander, D.M., et al. 'Direct Digital Process Control: Practice and Algorithms for Microprocessor Application.' In Proc. of the IEEE. Vol 6, No. 2, Feb. 1978, pp199-208.

AVIZ78 Avizienis, A. 'Fault Tolerance: The survival Attribute of Digital Systems.' In Proc. of the IEEE. Vol. 6, No. 10, October 1978, pp1109-1125.

AYRE83 Ayres, R.U., Miller, S.M., Robotics Applications and Social Implications. 1983, Ballinger Publishing Company, Cambridge, Massachusetts.

BAKE74 Baker, K.R. Introduction to Sequencing and Scheduling. New York, N.Y.: John Wiley & Son, Inc., 1974.

BARR81 Barr, A., Feigenbaum, E.A. The Handbook of Artificial Intelligence, Vol. 1, Los Altos, CA.: William Kaufmann Inc., 1981.

BURG86 Burgess, J., Krieger, M. 'SYMED: A General Purpose Graphical Editor for Systems Design.' Presentation given at the O.C.R.I. sponsored workshop on Computer-Aided Design of Real-Time Systems held in Ottawa, November 21, 1986.

BONN82 Bonner, S.A., Shin, K.G. 'A Comparative Study of Robot Languages.' In IEEE Computer. Vol. 15, No. 12, December 1982, pp 82-96.

BONN83 Bonner, S.A. 'Comparative Study of Robot Programming Languages.' M.S. thesis, Resselaer Polytechnic Institute, Troy, N.Y., 1983.

BOW84 Bow, Sing-Tze, Pattern Recognition: Applications to Large Data-Set Problems. Marcel Dekker, Inc., New York, N.Y., 1984.

BOWE80 Bowen, B.A., Buhr, R.J.A. The Logical Design of Multiple-Microprocessor Systems, Englewood Cliffs, N.J.: Prentice-Hall, 1980.

BRIS82 Bristol, E. 'Process Control: An Application Theorist's View of Control.' In IEEE Control Systems Magazine. March 1982, pp3-8.

CHAS81 Chase, R.B. Aquilano, N.A. Production and Operations Management. Richard D. Irvin, Inc., Homewood, Illinois, 1981.

CLAR61 Clark, J.E.T. Musical Boxes: A History and an Appreciation. London, England: George Allan and Unwin Ltd, 1961.

CULB57 Culbertson, J.T. 'Robots and Automata: A Brief History.' In Computers and Automation. Vol. 6, No. 3, March 1957, pp32-37, and Vol. 6, No. 4, April 1957, pp28-35.

CULB63 Culbertson, J.T. The Minds of Robots, Urbana, Illinois: University of Illinois Press, 1963.

DAUM62 Daumas, M. A History of Technology and Invention: Progress Through the Ages. Vol I, II, III. Paris, France: Presses Universitaire de France, 1962.

DURA87 Duran, J. 'Applicability of Process Activity Diagrams as a System Design Tool.' M.A.Sc. E.E. thesis, University of Ottawa, May, 1987.

DWOR85 Dworkin, M.L., Powers, G., Perry, R. 'CIM Technology - A Virtual System Approach.' In Test & Measurement World. December, 1985, pp. 62-68.

ENCY83 Ralston, A., Reilly, Jr., ed. Encyclopedia of Computer Science and Engineering. Second Edition, Van Nostrand Reinhold, New York, N.Y., 1983, pp 1111-1119.

FAGE81 Fagenbaum, J. 'Manufacturing 1: Industrial Control.'. In IEEE Spectrum. January 1981, pp70-73.

FAGE82 Fagenbaum, J. 'Industrial Controls.' In IEEE Spectrum. January 1982, pp61-64.

GROO8Ø Groover, M.P. Automation Production Systems and Computer-Aided Manufacturing. Englewood Cliffs, Prentice-Hall, 198Ø.

GROO83 Groover, M.P. 'Fundamental Operations.' In IEEE Spectrum. May 1983, Vol.2Ø, No.5, pp65-69.

GROO84 Groover. M.P., Zimmers, E.W. Jr. CAD/CAM: Computer-Aided Design and Manufacturing. Englewood Cliffs, N.J.: Prentice-Hall, Inc. 1984

GUNN82 Gunn, Thomas G. 'The Mechanization of Design and Manufacturing.' In Scientific American. Vol. 247, No.3, September 1982, pp. 114-132.

HAMB8Ø Hamblen, J.O., Alford, C.O. 'An Inexpensive Modular Process Control Computer.' In IEEE Conf. on Computer Control Proc. 198Ø, pp71-73.

HARG8Ø Hargreaves, J.C., Krakowski, R.A. 'Distributed Process Control: A Micro-Mini Marriage.' In IEEE Computer. September 1977, pp44-56.

HOUS8Ø House, C.H. 'Perspectives on Dedicated and Control Processing.' In IEEE Computer. December 198Ø, pp35-49.

HUTC84 Hutchinson, G.K., Clementson, A.T. 'Manufacturing Control Systems - An Approach to reducing Software Costs.' In Robotics and Computer-Integrated Manufacturing. Vol. 1, No. 3/4, pp. 271-281, 1984.

JARV8Ø Jarvis, J.F. 'Visual Inspection Automation.' In IEEE Computer. Vol. 13, No. 5, May 198Ø, pp32-38.

JARV82 Jarvis, R.A. 'A Computer Vision and Robotics Laboratory.' In IEEE Computer. June 1982, pp8-23.

JOAN86 Joannis, R., 'Specification and Design of Multiple Microprocessor Systems Using Process Activity Diagrams.' M.A.SC E.E. thesis, University of Ottawa, August, 1986.

JOAN86b Joannis, R., Krieger, M. 'Process Activity Diagrams: A Tool for the Specification and Design of Multiple Microprocessor Systems.' In Proc of the I.S.M.M. International Symposium. Beverly Hills, CA, Feb. 1986.

JURG83 Jergen, R.K. 'Data-Driven Automation.' In IEEE Spectrum Vol. 20, No. 5, May 1983, pp. 34-35.

KASV80 Kasvand, T. 'Vision as Applied to Robots.' In Proc. of Seminar on Robotics in Manufacturing. Canadian Institute of Metalworking, 1980.

KASV82 Kasvand, T. 'The Robotics Scene in Japan.' National Research Council of Canada, Ottawa, Ont, 1982.

KELL81 Kelley, R.B. Birk, J.R. 'An Overview of the Basic Research Needed to Advance the State of Knowledge in Robotics.' In IEEE Trans. on Systems, Man, and Cybernetics. Vol. SMC-11, No. 8, august 1981, pp574-579.

KERZ84 Kerzner, H. Project Management: A Systems Approach To Planning, Scheduling, and Controlling. 2nd Edition, Van Nostrand Reinhold Company, New York, NY, 1984.

KRIE80 Krieger, M. Course notes on Computer Design.

KRIE82 Krieger, M., Santoro, N. 'Oligarchical Control of Distributed Processing Systems.' In Proc. of the 20th Allerton Conf. on Communications, Control, and Computing. 1982, pp793-802.

KRUG81 Kruger, R.P. Thompson, W.B. 'A Technical and Economic Assessment of Computer Vision for Industrial Inspection and Robotic Assembly.' In Proc. of the IEEE .Vol. 69, No. 12, December 1981, pp1524-1538.

LEFK79 Lefkowitz, I., Rose, C., et. al. 'Automatic Control by Distributed Intelligence.' In Scientific American. Vol. 240, No.6, June 1979, pp78-90,202.

LEFK82 Lefkowitz, I., Buchner, M.R. 'Distributed Computer Control for Industrial Process Systems: Characteristics, Attributes, and an Experimental Facility.' In IEEE Control Systems Magazine. Vol. 8, No. 2, March 1982, pp8-15.

LOZA83 Lozano-Perez, T. 'Robot Programming.' In Proc. of the IEEE. Vol. 71, No. 7, July 1983, pp821-841.

MAHM77 Mahmoud, M.S. 'Multilevel Systems Control and Applications: A Survey.' In IEEE Trans. on Sys., Man, and Cyber. Vol. SMC-7, No. 3, March 1977, pp125-143.

MCCL52 McClay, S.T. French Inventions of the Eighteenth Century. University of Kentucky Press, 1952.

MERC85 Merchant, M.E. 'Computer-Integrated Manufacturing As The Basis For The Factory Of The Future.' In Robotics and Computer-Integrated Manufacturing. Vol. 2, No. 2. 1985, pp.89-99.

MESA7Ø Mesarovic, M.D. 'Multi-Level Systems and Concepts in Process Control.' In Proc. of the IEEE. Vol. 58, No. 1, January 197Ø, pp111-125.

MICH79 Michel, G., Laurgeau, C., Espiau, B. Les Automate Programmables Insdustriels. Paris, France: Dunod Technique-Bordas, 1979.

MYER8Ø Myers, W. 'Industry Begins to Use Visual Pattern Recognition.' In IEEE Computer. Vol. 13, No. 5, May 198Ø, pp21-31.

NEME63 Nemes, T.N. Cybernetic Machines, New York, N.Y.: Gordon and Breach Science Publishers Inc., 1963.

NEVI77 Nevins, J.L., Whitney, D.E. 'Research on Advanced Assembly Automation.' In IEEE Computer. December 1977, pp24-38.

NEVI78 Nevins, J.L., Whitney, D.E. 'Computer Controlled Assembly.' In Scientific American. Vol. 238, No. 2, February 1978, pp62-74, 16, 162.

NITZ85 Nitzan, D. 'Development of Intelligent Robots: Achievements and Issues,' In IEEE Journal of Robotics and Automation. Vol. RA-1, No. 1, March 1985, pp3-13.

ORDH73 Ord-Hume, A.W.J.G. Clockwork Music. London, England: George Allen and Unwin Ltd, 1973.

OWEN84 Owen, A.D. Flexible Assembly Systems. Plenum Publishing Corporation, New York, NY, 1984.

PALM65 Palmer, R.R., Colton, J. A History of the Modern World. New York, N.Y.: Alfred A. Knopf, 1965.

PAQU85 Paquette, J.J.R. 'The Next Generation Avionics Systems - A Prediction.' M.A.Sc. thesis, Carleton University, September, 1985.

PAUL79 Paul, R.P. 'Robots, Models, and Automation.' In IEEE Computer. July 1979, pp19-27.

PAUL81 Paul, R.P. Robot Manipulators: mathematics, Programming, and Control-The Computer Control of Robot Manipulators. Cambridge, Massachusetts: The M.I.T. Press, 1981.

PETR86 Petriu, E., Course Notes on Robotics and Automation.

PICH83 Piché, P.R., Krieger, M., Santoro, N. 'Oligarchy: A Control System for Flexible Manufacturing Systems.' In Proc. of the Second I.A.S.T.E.D. International Symposium on Robotics and Automation. June 22-24, 1983, Lugano, Switzerland.

PICH84 Piché, P.R., Krieger, M. 'Segmented Microprogramming: A Technique for Designing Special Purpose VLSI Processors.' In Proc. of the IEEE ICCD: VLSI in Computers. Oct. 1984, pp. 393-396.

PICH86 Piché, P.R., Krieger, M., Gauthier, C.A., Petriu, E. 'PAD: A New Tool for FMS Design.' In Proc. of 12th Annual IECON. Vol. 2, 1986, pp799-804.

PORT80 Porter, G.B., Mundy, J.L. 'Visual Inspection System Design.' In IEEE Computer. Vol. 13, No. 5, May 1980, pp40-48.

RENA79 Renaud, M., Iturralde, J.Z. 'Robot Manipulator Control.' In Proc. of the 9th International Symp. on Industrial Robots. 1979, pp463-475.

ROSE76 Rosen, C.A., Nitzan, D. 'Programmable Industrial Automation.' In IEEE Trans. on Computers. Vol. C-25, No. 12, December 1976, pp1259-1270.

ROSE77 Rosen, C.A., Nin, D. 'Use of Sensors in Programmable Automation.' In IEEE Computer. December 1977, pp12-23.

SAND84 Sanderson, A.C. 'Parts Entropy Methods for Robotic Assembly System Design.' In Proceedings of the International Conference on Robotics. Atlanta, March 13-15, 1984, pp. 600-608.

SCEN76 Van Nostrand's Scientific Encyclopedia, Fifth Edition, Edited by D.M. Constantine, Van Nostrand Reinhold Company, New York, 1976, pp1727-1728.

SHAP78 Shapiro, S.F. 'Digital Technology Enables Robots to See.' In Computer Design. Vol. 17, No. 1, January 1978, pp43-59.

SHAP79 Shapiro, S.F. 'Vision Expands Robotic Skills For Industrial Applications.' In Computer Design. Vol. 18, No. 9, September 1979, pp78-87.

SIMO80 Simons, G.L. Robots in Industry. Manchester England: National Computing Center Ltd., NCC Publications, 1980.

SKRO8Ø Skrokov, M.R. Mini and Microcomputer Control in Industrial Processes: Handbook of Systems and Application Strategies. Van Nostrand Reinhold, 198Ø.

SMIT7Ø Smith, C. 'Digital Control of Industrial Processes.' In A.C.M. Computing Surveys. Vol. 2, No. 3, September 197Ø, pp211-241.

SNYD85 Snyder, W.E. Industrial Robots: Computer Interfacing and Control. Prentice-Hall, Inc., Englewood Cliffs, N.J., 1985.

SPEC83 Special Issue on Data Driven Automation, IEEE Spectrum. May 1983, Vol. 2Ø, No. 5.

STRA77 Strait, B.G., Thuot, M.E., Hong, J.P. 'A Distributed Microcomputer Control System for a High-Energy Gas-Laser Facility.' In IEEE Computer. September 1977, pp36-43.

TSUJ8Ø Tsuji, S., Masahiko, Y. 'Industrial Computer Vision In Japan.' In IEEE Computer. Vol. 13, No. 5, May 198Ø, pp5Ø-63.

TRUX83 Truxal, C. 'The Professional Experts Are In Short Supply.' In IEEE Spectrum. May 1983, Vol. 2Ø, No. 5, pp93-94.

VAIL82 Vaillancourt, S., Krieger, M. 'Segmented Microprogramming.' In Proceedings of the 1982 Conference on Information Sciences and Systems. pp. 231-235.

VAIL8Ø Vaillancourt, S. 'Segmented Microprogramming.' M.A.SC. E.E. thesis, University of Ottawa, Sept. 198Ø.

WEIT8Ø Weitzmann, C. Distributed Micro/Minicomputer Systems - Structure, Implementation, and Application, Englewood Cliffs, N.J.: Prentice-Hall, 198Ø.

WELC8Ø Welch, J.R. 'Automatic Speech Recognition-Putting It To Work In Industry.' In IEEE Computer. Vol. 13, No. 5, May 198Ø, pp65-73.

WILL66 Williams, T.J. 'The Development Computer Control in the Continuous Process Industries.' In Proc. of the IEEE. Vol. 54, No. 12, December 1966, pp1751-1756.

WILL78 Williams, T.J. 'Hierarchical Control For Large Scale Systems-A Survey.' In I.F.A.C. World Conference Proceedings. 1978.

ZORZ63 Zorzoli, G.B., Eco, U. The Picture History of Inventions From Plough to Polaris. New York, N.Y.: Macmillan Co., 1963.

Appendix

In this Appendix, a copy of three publications written during the course of research showing the evolution of this thesis. They are listed in the Bibliography under [PICH83], [PICH84], and [PICH86].

Philippe R. Piché/Moshe Krieger
Dept. of Electrical Engineering
University of Ottawa
OTTAWA, Ontario CANADA
K1N 6N5

Nicola Santoro
School of Computer Science
Carleton University
OTTAWA, Ontario CANADA
K1S 5B6