MACRO-CAPP: A CAPP CIM INTERFACE

by

Krishnaswami Srihari

Dissertation submitted to the Faculty of the

Virginia Polytechnic Institute and State University

in partial fulfillment of the requirements for the degree of

Doctor Of Philosophy

in

Industrial Engineering and Operations Research

APPROVED:

Timothy J. Greene, Ph.D, Chairman

Wolter J. Fabrycky, Ph.D

Michael P. Deisenroth, Ph.D

Loren P. Rees, Ph.D

Marilyn S. Jones, Ph.D

August 1988

Blacksburg, Virginia

MACRO-CAPP: A CAPP CIM INTERFACE

by

K. Srihari

Timothy J. Greene, Ph.D, Chairman

Industrial Engineering and Operations Research

(ABSTRACT)

There exists today a variety of Computer Aided Process Planning (CAPP) systems that have been designed, developed, and implemented irrespective of the facility's condition and status. It is often found in practice that Computer Integrated Manufacturing (CIM) constituents such as production control, loading, sequencing, scheduling, etc. do not interact with Computer Aided Design (CAD), Computer Aided Manufacturing (CAM), or CAPP. They operate as stand alone techniques that are not inter-related in the CIM scenario. This could be overcome through increased, improved communication between CAD, CAPP, CAM, and other CIM constituents.

CAPP has to be tied into the computerization of other CIM functions. An approach in this direction is what this research presents. Its uniqueness is that it relates CAPP in a flexible manufacturing system atmosphere with scheduling, in effect relating CAPP with production control. It integrates process selection and route generation with factors such as facility congestion, work in process, flowtime, machine utilization and dynamic shop conditions.

The generation of alternate routes, and the incorporation of this technique in a CAPP system is an unique approach to the problem of inter-relating CAPP with other CIM components. This involved the design and development of software that can

model facility capacity, understand part construction, maintain and track shop status, reason through the facility capacity to arrive at possible machining sequences and job routes, and apply a heuristic to arrive at the job route through the facility. This results in the introduction and implementation of the concept of dynamic scheduling and alternate route generation in CAPP systems.

The objective in global terms was to construct a CAPP system that considers routing and production control for a FMS that consists of several high capacity, modern machines. The concepts mentioned above are combined and coalesced in a CAPP system that truly provides computerized assistance to the process planning function at a macro-level. This research attempts to create a truly integrated CAPP system within a CIM atmosphere.

Acknowledgements

It is not possible to overstate my gratitude to Dr. Timothy J. Greene for his wise and patient steersmanship, encouragement, involvement, and inspiring guidance throughout my education at Virginia Tech, especially during the time I spent as a doctoral student and during the course of this research. He accepted the awesome task of editing the manuscript of this thesis, and reduced the literary ineptness of the author to a minimum without altering the meaning of the written word.

Dr. M.P. Deisenroth has been a constant source of inspiration and guidance. I am indebted to Dr. Loren Rees for his constant support and encouragement. Drs. Wolter J. Fabrycky and Marilyn S. Jones have offered valuable assistance in background theory for which the author is grateful. The guidance and thoughts of my doctoral committee were invaluable in content. The support and encouragement that the author received from the faculty and staff of the Department of Industrial Engineering and Operations Research is also greatly appreciated.

The author wishes to express his gratitude to his parents for their continued encouragement. Without their love and support, this research would not have been possible.

Table of Contents

List of Illustrations

1.0 INTRODUCTION

1.1 Introduction To Process Planning

Process planning is the task of transforming part design specifications from detailed engineering drawings to a collection of operating instructions for part manufacturing [6,68]. It has been stated that without a plan there is no process, and without a process there is no part [2]. The creation of an efficient process plan is very important to the orderly, efficient, and effective operation of a manufacturing facility [2,9,10]. If process plans are erroneous or inefficient, then they can cause excessive scrap, extensive re-work, poor shop performance, high manufacturing costs, and extensive WIP [2,9,10].

Process planning is often considered a 'stand alone' feature that is unrelated to and uneffected by other production and manufacturing functions in an integrated manufacturing facility [2,67]. Often, a Computer Aided Process Planning (CAPP) system's interaction with other manufacturing and production functions is not considered

in it's design and implementation. This results in a narrow approach to automated process plan generation, since an efficient CAPP system supplies information in sufficient detail to help it interface with other segments of a Computer Integrated Manufacturing (CIM) scenario such as scheduling, sequencing, loading, etc. In addition, process planning systems need to interface with scheduling mechanisms in the identification and utilization of alternate routes in the planning and implementation of production floor operations [67].

Initial, raw material used in a facility may take a variety of forms that could include but need not be limited to bar stock, slabs, plates, castings, and forgings. The process planner needs to provide a list of processes that can be used to convert raw material into a finished product. Required inputs to the planning schema include [6,12,67,68]:

- Geometric features.
- Dimensional sizes.
- Tolerances.
- Materials.
- Finishes.

These inputs are analyzed and evaluated in order to select an appropriate sequence of processing operations based upon specific, available machinery and workstations [9,10]. A process plan then needs to generate operational details or outputs such as [10,67,68]:

- Sequence of operations.
- Speeds.
- Feeds.
- Depth of cut.
- Required tooling.

- Tool paths.
- Required machine tools.
- Workpiece holding devices.
- Standard time.
- Standard cost.
- Material removal rate.
- Job route.

In general, a process plan is prepared using available design data and manufacturing knowledge. The process planning function has been described as 'a subsystem that is responsible for the conversion of design data to work instructions' [68]. In order to produce good, consistent, and accurate process plans, it is important that a logical, systematic process to develop, maintain, and update plans exists. This requires the establishment and maintenance of a standard data base and the implementation of a method to process the data.

The introduction of computerization and the subsequent automation of process planning has evolved due to a substantial need caused by the lack of qualified personnel, need for consistency in planning, and the need to incorporate knowledge on continually evolving new processes [67]. Automation of the process planning function has been facilitated by the quantum improvement in computer hardware and related facilities over the last decade. It has reduced planning time, skill level needed for a planner, costs, and increased productivity. The benefits of this function have been documented by several researchers [2,8,9,12,67,68,85].

In the recent past, a major thrust has been to promote the use of CAPP as a necessary, logical interface between Computer-Aided Design (CAD), and Computer-

Aided Manufacturing (CAM) [7,8,9,10]. Current CAD systems tend to concentrate on automated design, solids modelling, drafting, and tool path generation for computer numerically controlled machine tools. CAM systems tend to concentrate on automated manufacture and robot usage [67]. Integration of these two procedures could be brought about ideally through the use of CAPP [12,67, 68]. Ideally, the functions that may need to be performed between the design of a product and it's eventual manufacture using CAM are described in Figure 1. It is obvious that CAPP is one of those functions, and does interact and relate to the others within a CIM scenario.

An ideal CIM scenario contains several important functions that take place between product design using CAD and manufacture using CAM. These functions include the procurement of inventory at the optimal time, facility loading, release of jobs to the facility, scheduling and sequencing of parts through the facility, capacity planning, Just-In-Time (JIT) techniques, materials handling analysis, cost estimation coupled with make or buy analysis, creation and loading of manufacturing instructions for a specific part for the CNC machine that will manufacture it, etc. A CAPP system should play a crucial role in the implementation of each one of these functions. It is indeed an integral part of the CIM function that interacts in a dynamic manner with the other constituents.

As perceived in Figure 1 it is obvious that in actual manufacturing facilities, two way communication needs to exist between various CIM facets. It has always been mentioned that CAPP reduces the time from design to actual manufacture. The implementation of CAPP along with Computer-Aided Engineering in Design and Manufacture (CAEDM) concepts enables the design engineer to handle complex designs, visualize results early in the actual design process, and communicate those results

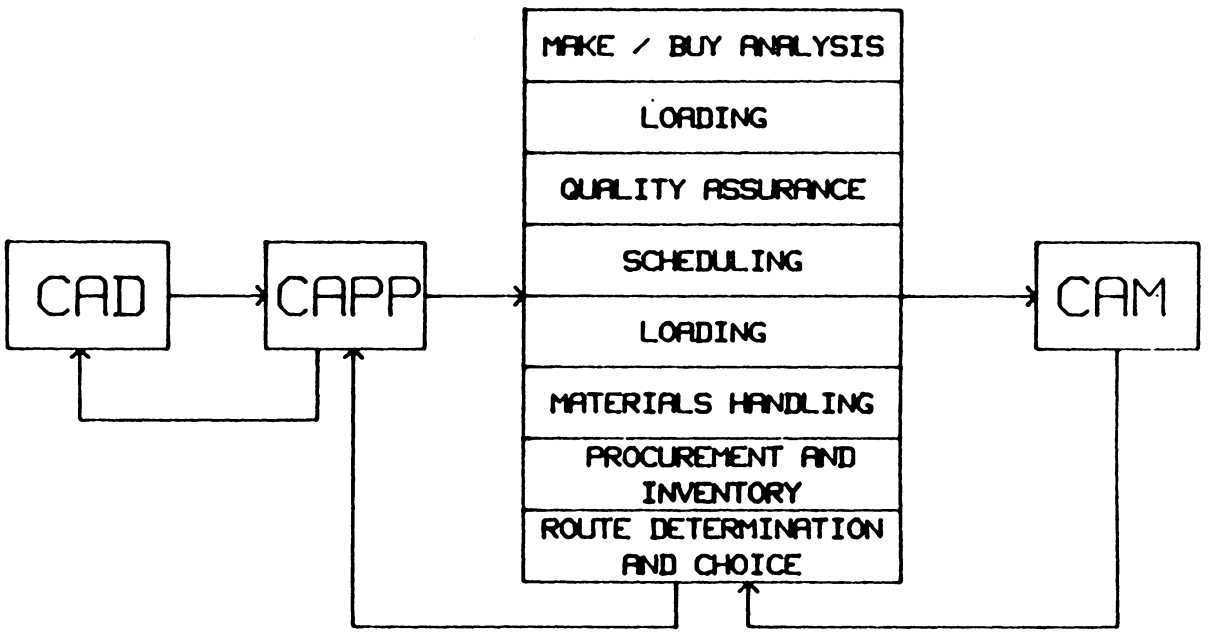


FIGURE 1 - CAPP IN A CIM SCENARIO

more effectively and efficiently. This is a tool and a technique that can be used to help achieve manufacturing goals.

The need for better communication between manufacturing facets is dictated by the need to produce more products in shorter time spans [59]. Each product must be competitive in terms of performance, quality, and cost. Computers today allow the designer to evaluate complex design problems, consider their manufacture, and understand critical factors and processes in time frames that would have been considered impossible a few years ago. Since manufacturing decisions can influence a variety of design factors, interaction and communication will help in influencing product design as also every other facet within the CIM umbrella. If critical problems should occur in any function, then compromises can be negotiated. This will help in reducing the gaps between individual CIM components.

Figure 2 illustrates the concept of concurrent engineering. It is the process of including manufacturing decisions in the actual design process. This concept, when implemented, will increase design-manufacture communication. It will help the designer understand manufacturing constraints, making the manufacturability of a part an important factor. It would attempt to cater to the ever increasing need to reduce the time between design and actual manufacture.

Current thought on the utilization and implementation of CAPP systems has concentrated, to an extent even stagnated, on simple manufacturing procedures. Processes that involved the use of complex, high capacity, modern machinery have not been considered. Individual machines have been considered, and very rarely have been combinations of machinery actually been studied with respect to CAPP implementation [9,10,67,68]. Machines often considered are lathes, milling machines, or

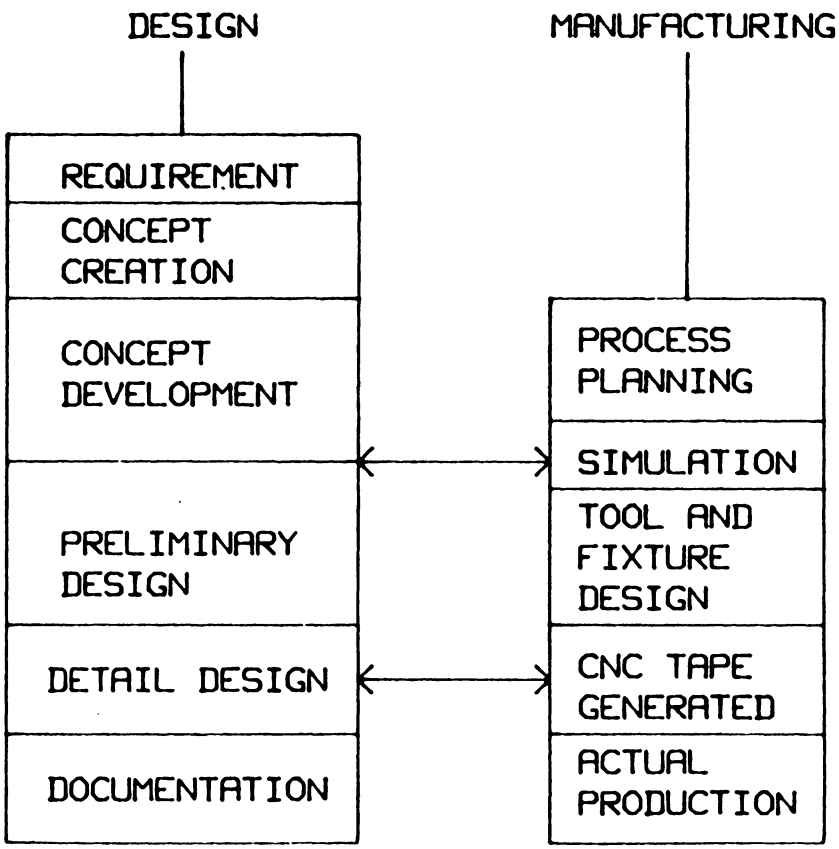


FIGURE 2 - CONCURRENT ENGINEERING

simple drill presses. Operations considered are those that can be performed on these elementary machine tools. The narrow realm of CAPP system development is one of the primary factors that has balked it's widespread application in manufacturing industry [12,67,68,85].

Integration of CAPP systems into a modern facility must necessarily involve consideration of the dynamic nature of the facility to insure an efficient and effective match [48,67]. The CAPP system must utilize this information in deciding upon the route to be taken by a job and the actual sequence of operations to be followed. This aspect of CAPP has not been widely researched, since all currently known CAPP systems are static in nature. This has resulted in the development of a surfeit of CAPP systems that are applicable to a very narrow manufacturing area, that are static in nature, and do not actually interact with other manufacturing functions. Static CAPP systems would not be efficient and effective in performing the process planning function since they do not consider actual shop conditions in deciding upon an appropriate process plan.

It is obvious that although many researchers and practitioners have promoted CAPP as a logical interface been CAD and CAM, much needs to be done to achieve this goal. CAPP is currently perceived as a post-processor to CAD and a pre-processor to CAM. The thought of integrating CAD and CAM with CAPP along with other CIM facets needs to be considered. The concept of interfacing CAD and CAM through CAPP requires researchers to move away from traditional ideas on CAPP that are narrow in perspective, to a broad outlook of CAPP within CIM. This will cause an increase in actual CAPP system implementation in manufacturing facilities.

1.2 Research Area Considered

The problem under consideration in this research addresses and uses concepts related to several component areas. They include :

- CAPP.
- Group Technology.
- Flexible Manufacturing Systems.
- Artificial Intelligence.

These topics are addressed and their relevance to this research is presented in the following sections.

1.2.1 CAPP

Over the years, process planning has evolved from the use of manual techniques through the use of computer assisted variant systems to the implementation of computerized generative systems [12,67]. This evolution is evident from a perusal of literature [67,68,85]. However, it is also apparent that most current systems deal with very simple manufacturing facilities [6,7]. Traditionally, CAPP systems have concentrated on the determination of operations that need to be performed, their sequence, and tool path determination [68,85]. None of the prototype variant systems that have been published in the literature have considered a facility where parts could be manufactured with different machines [67,85]. None have considered a variety of state-of-the-art machining centers used together to manufacture parts in various quantities. The presence of different machines that could generate similar surfaces

through different operations has apparently never been considered [67]. These Computer Numerically Controlled (CNC) machines are being increasingly used in industry. They are slowly but surely replacing traditional machine tools in the manufacturing shop.

The reason for this replacement process is quite obvious. A single CNC turning center can perform a variety of operations that could include turning, facing, boring, thread cutting, taper turning, knurling, etc. If required, it can perform several of these operations simultaneously. A single CNC turning center can replace several conventional machine tools. Typically, they can produce at a higher rate at a better quality than conventional machine tools. The CNC machine can, for example, produce a thread in more than one method that could include the use of dies, single point tools, taps, etc. Since the turrets of a CNC machine can normally hold several tools, for example sixteen, the planner has more flexibility in choosing a viable, effective, and efficient sequence of operations. This is further enhanced by the ability of the machine tool to manufacture a product using several different operational sequences.

The machining centers considered in this research have the ability to load and unload jobs automatically. These versatile machines, that are equipped with automatic tool changing equipment, can perform a very wide variety of manufacturing operations. The actual specifications, features, and operational details of the machines under consideration are described in Chapter Four. These machines, with their superlative capacities, provide very high manufacturing rates and increased shop thruput in terms of the number of products produced.

Most CAPP systems assume infinite machine capacities in determining process route [67,68,85]. This is assumption is not valid when viewed from a production con-

trol perspective. Under this perspective, it is obvious that each machine has a fixed, finite capacity. CAPP systems determine part manufacture sequences and routes oblivious of machine capacities and dynamic shop conditions. This problem is circumvented in practice by the production planner and process planner through the informal organization. They decide, for example, upon the alternate route that could be used if a particular route is congested. It is the production control function that balances committed work against capacity available to process a product. Current process planning methods do not consider the capacity of the route specified as a constraint.

It is becoming increasingly obvious that in a batch manufacturing atmosphere, it would be advantageous to the planner if multiple job routes existed to manufacture a product [37,49,86]. One reason for this is that the facility may process the same part several times over a period of time. In addition, it is also possible that the batch size required may change every time the same product needs to be manufactured. If multiple routes are identified to manufacture that part, then the route that is most suitable in a particular scenario can be used. This is possible in a robust and versatile facility in which jobs can be processed using more than one sequence of operations using different machines. A facility containing several different types of machining centers would be an ideal example. This is obvious when the capabilities of these machines are analyzed. Their capabilities include several operations that can be used by the process planner to arrive at the same end. The process planner can then decide, depending upon the current, dynamic shop status, the optimal route that should be used by the batch.

1.2.2 Group Technology

In order to function, CAPP systems need input techniques that comprehensively describe and define for the system software the products that need to be manufactured. Current CAPP systems use either Group Technology (GT) based coding schemes or CAD based input systems [16,24,40,53,62,63]. However, the use of CAD systems as input devices has not been comprehensively solved [67]. The use of CAD input devices to comprehensively describe part profiles does require more research as indicated by several researchers [68]. GT part coding techniques have been in use for several years to comprehensively describe parts. Several comprehensive part coding schemes have been developed and used in the past by both academia and industry [24,50,53,54]. The input from a part coding system to describe a part must be simple, concise, yet comprehensive in part description. It is possible that no existing part description technique can be used without adaptation in a specific facility. This may require the design of a new scheme or the adaptation of an existing scheme keeping in mind the scenario in which it is to be implemented.

A standardized coding technique reduces part multiplicity, helps recognize repeat parts, assists in standardizing shape characteristics, increases recording ease, helps group together similar parts, and promotes simplicity of use. The code used also helps in rapid information retrieval and use by system software. The part description function falls between part design and CAPP. It provides the CAPP system software with a method to recognize part characterisctics, and help in manufacturing sequence generation. It also helps in recognizing similar or identical parts.

1.2.3 Flexible Manufacturing Systems

Flexible Manufacturing Systems (FMS) are becoming more commonplace in manufacturing facilities [87]. An FMS has a variety of high capacity, versatile machinery linked together through the use of an automatic materials handling system that reaches every process station [33,86]. The entire system is integrated under common computer control [49]. However, there are no current CAPP systems that have been developed and designed specifically for an FMS. Definitely, these systems will play an important part in the FMS factories of the future.

A typical FMS combines the flexibility of general purpose machines with the high productivity of a transfer line. FMS cells usually usually consist of CNC machining centers, drilling and turning centers. These are flexible, high capacity, versatile machines. Cells usually include inspection and cleaning stations. Most current FMS's are intended for machining operations. Batch manufacture is undertaken in an FMS. The processing sequence and the machine setup vary for each batch considered. In such an atmosphere, with varying job types and lot sizes, process plans need to be generated often. The FMS's composition is such that there a fixed number of machines, the capacity, capability, and specifications of each machine are known. The advantages of considering alternate routes and job sequences in FMS scheduling and capacity planning has been considered by several researchers [49,64,86,91].

This research considers the role of CAPP in a modern FMS with several machining centers linked by an automatic materials handling system. The capacity and versatility of these facilities makes the design and implementation of a CAPP system

very complex. Such a facility is ideal to illustrate and implement the principles of generating alternate process routes and sequences [86].

1.2.4 Artificial Intelligence

Artificial Intelligence (AI) is an emerging technology that has attracted considerable attention in recent years. A simple view of AI is that it is concerned with devising programs to make computers smarter [3]. Research in AI is focussed on developing computational approaches to intelligent behavior [3,44]. It has two goals, namely to make machines and computational processes more efficient and to understand intelligence [45]. AI approaches involve heuristic programming and can easily handle dynamic situations, but cannot be used as easily for computationally intensive optimization routines [44,45]. The computer programs with which AI is concerned involves symbolic processes involving complexity, uncertainty and ambiguity similar to decision making situations that humans face constantly in the real world. AI provides techniques for flexible non-numerical problem solving. These techniques include symbolic information processing, heuristic programming, knowledge representation, and automated reasoning [72]. A system that could reason and choose the appropriate course of action can be faster, more effective, efficient, and viable than a rigid one [3,45].

Search through the state space is a fundamental concept of the problem solving process [72]. The search is performed using rules to guide the process of going from one state to another. Rules used are typically those based upon an expert's knowledge of the system's behavior. Direct algorithmic types of approaches can be used

in the search process by encoding them into rules or external routines. This provides a framework into which algorithms that are appropriate for the solution of a typical subproblem can be embedded. The control structure guiding the search through the state space is crucial to the efficiency of the problem solving process. This assumes more importance as the complexity of the problem increases [44,45].

AI has been described as a method that exploits knowledge that should be represented in such a way that [44,45]:

- It captures the generalizations, and does not require each situation to be represented separately.
- It can be understood by people who provide it. Knowledge in programs must be provided by people in terms they understand.
- It can be modified easily to correct errors and to incorporate changes.
- It can be used in a variety of situations even if it is not totally accurate or complete.
- It can be used to overcome it's own bulk by helping to narrow the range of possibilities that must be considered.

An AI system must contain a lot of knowledge if it is to handle complex problems. But as the amount of knowledge grows, it becomes harder to access the required information when needed, and hence more knowledge may be required to reach the desired solution. This results in even more knowledge to manage.

AI techniques are used for problems for which an algorithmic procedure does not exist and a search is required. The basic elements of AI are [72]:

- Heuristic Search.
- Knowledge Representation.
- Logic And Reasoning.
- AI Programming Languages.

A wide variety of possible applications of AI in manufacturing exist [72]. Several researchers have observed that CAPP is an ideal area for the implementation of AI techniques [15,44,45,56,67,72]. The increased use of AI has also resulted in the increased use of languages such as Prolog and Lisp. These languages are declarative rather than procedure oriented. They are very powerful languages that have a simpler syntax than most traditional, complex languages. Due to their relentless search and identification of possible solutions, these languages can identify all possible solutions for a specific goal.

1.2.5 Summary

This research, in addressing the problem area under consideration, combines and coalesces, among others, the concepts of CAPP, dynamic shop scheduling, multiple (alternate) routes in FMS scheduling, GT, AI, and part coding in the design and implementation of a prototype CAPP system for an FMS that contains state-of-the-art machinery. A practical CAPP system needs to be dynamic in nature to readily respond to facility status changes. It is imperative that comprehensive and concise part coding techniques be developed to help the planner and the system software in arriving at a possible, practical process plan. Process planning systems need also to consider the possibility of using more than one method or job route to process a part. Mechanisms to decide between these alternatives need to be used. Heuristics or algorithms programmed in software could be used to decide upon an acceptable route and sequence of operations for the job depending upon the performance measure(s) to be considered. The facility modelled in the construction of the proto-

type system must be realistic, and must model and be representative of a factory of the future.

1.3 Problem Statement

There exists today a variety of CAPP systems that have been designed, developed, and implemented irrespective of the facility's condition and status. There are systems that have been developed for individual pieces of machinery completely oblivious of the surroundings and situations in which these would be working [68,85]. As FMS's become more commonplace, it is imperative that CAPP systems be designed considering their characteristics. In practice, very rarely do CAPP systems perform as an integrating element between CAD and CAM [67,68]. They do not consider dynamic facility status in identifying job routes [48]. Possible alternate routes or operations that the job can use through the facility are not considered in arriving at optimal processing sequences and routes. There does not exist a prototype CAPP system, as evident from existing literature, that considers and addresses the implementation of these concepts and ideas in its design and development. A CAPP system that serves as integral part of a CIM scenario needs to be developed.

Although much has been said about the importance of CAPP as a link between CAD and CAM, much needs to be done to make this a reality [9,10]. This research attempts to achieve this goal. It is often found in practice that constituents of CIM such as production control, loading, sequencing, scheduling, etc do not interact with

CAD, CAM, or CAPP. They operate as stand alone techniques that are not interconnected, inter-related, or interwoven in the CIM scenario.

This is obvious from the techniques and methods used in design, process planning, and production control. Design, for example, specifies that very fine tolerances need to be held. These tolerances may not be critical in the actual application and use of the part. This requirement could cause tremendous problems in manufacture that could be overcome if superfluous design constraints were removed. Process planning on the other hand assumes that tolerances once set cannot be changed. This results in additional processes being scheduled in scenario's in which they are unnecessary. This is a problem that could be overcome through increased, improved communication between CAD, CAPP, CAM, and other CIM constituents. In practice, this communication rarely exists.

While modularized, stand alone software exists for both production control and process planning, they are not tied together. They operate independent of each other, often causing unacceptable plans or routes when viewed from either a process planning or a scheduling perspective. There is lacking in application and use an integration of CAPP with other CIM components. The automation of the process planning function has to be tied into the computerization of other CIM functions. An approach in this direction is what this research seeks to present. The uniqueness of this research is that it relates CAPP in an FMS atmosphere with scheduling, in effect relating CAPP with production control. It relates and integrates through software process selection and route generation with factors such as facility congestion, WIP, flowtime, machine utilization and dynamic shop conditions. The generation of alternate operations and routes, and the incorporation of this technique in a CAPP system

is an unique approach to the problem of inter-relating CAPP with other vital CIM components.

1.4 Objective

A prototype CAPP system that services a modern FMS facility consisting of multiple machining centers was researched and developed. The system contains in software the capacity, characteristics, and capability of each machine. The system assimilates part characteristics, features, and other relevant details prior to devising feasible machining sequences to generate the finished part from the raw material. The system analyzes possible alternate methods to produce the same product, and suggests the optimal machining sequence and job route chosen through the implementation of a heuristic. The prototype system developed integrates through software CIM constituents such as CAPP, sequencing, and scheduling.

The heuristic used by the system to decide upon the optimal sequence of operations and route through the system considers the flowtime the job would require if a particular route is used through the system. Flowtime minimization is therefore the major performance measure. The overall logic of the CAPP system is illustrated by Figure 3. This insures that at each stage of operations, prior to actual job release, the facility's dynamic situation is considered in the routing decision.

The software developed for this prototype system contains an inference mechanism that outputs to the user information such as:

- Machines used in this facility.

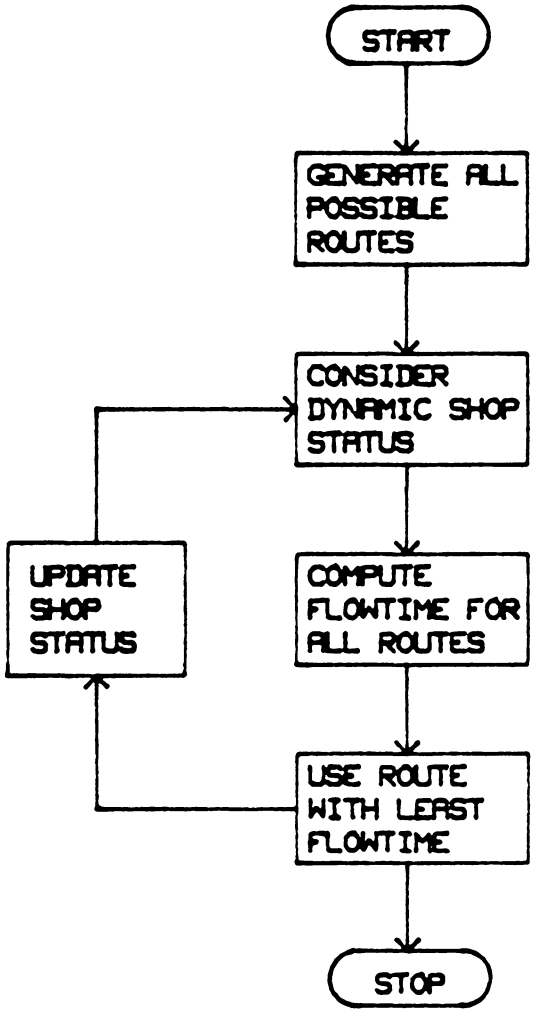


FIGURE 3 - HEURISTIC IMPLEMENTATION

- Machine specifications.
- Material types considered.
- Suggested speeds, feeds, etc for each material type.
- Job types considered.
- Operations that could be used to generate a specific surface.
- Surface finishes possible in this facility.
- Machines that could be used to generate a specific surface.
- Process plan with alternate routes and processes.
- Process plan suggested for use considering dynamic shop status.

This information is envisaged as a tool that could be used to educate a new process planner or a new employee on the actual capacity, specifications, and capabilities of the facility under consideration.

This research considered a pseudo-facility, an FMS with five machining centers. Each machining center is a versatile, computer controlled machine that can perform multiple operations. They are similar to the most modern machinery currently produced. The system considered is described in detail in Chapter Four. Jobs arrive to the system at predetermined times using known arrival rates. A GT coding system is used by the process planner to input information on part parameters. The system has the individual machine's capacities and capabilities stored in software. The process plan for the part is devised through an intricate planning procedure that mimics the rational thought of a human when executing the process planning process correctly. The outputs include feeds, speeds, depth of cut, sequence of operations, job route, and machining times. Due to the versatile nature of machinery, it is possible that alternate routes and job sequences may exist. This possibility is analyzed, and alternate routes that can be used to manufacture the part are studied. The most

appropriate route depending upon the dynamic shop condition is determined through the use of a heuristic.

There has been very little research, development, and implementation in the area of CAPP with respect to FMS systems. This research needed to initially design, develop, and implement a custom GT coding technique that works efficiently in this specific application. It is used as the input technique to describe parts comprehensively. This involved the design and development of software that can model facility capacity, understand part construction, maintain and track shop status, reason through the facility capacity to arrive at possible machining sequences and job routes, and apply a heuristic to arrive at the optimal path through the facility. This results in the introduction and implementation of the concept of dynamic scheduling and alternate route generation in CAPP systems. System software then analyzes relevant inputs such as material, quantity, and part profile to arrive at possible plans.

The objective in global terms was to construct a CAPP system that considers routing and production control for an FMS that consists of several high capacity, modern machines. It involved fulfilling the sub-objectives such as the design of a GT part description code, design and development of a prototype CAPP system that generates alternate methods and routes to produce a part, implementation of a dynamic shop status module, and the use of a heuristic to choose between alternate routes in an FMS. These objectives were then combined, coalesced, interwoven, and fused in the design and development of a CAPP system that truly provides computerized assistance to the process planning function from the part description stage to the actual manufacture in the facility. This research attempted to create a truly integrated CAPP system within a CIM atmosphere.

1.5 Dissertation Overview

The first chapter of this dissertation introduces the process planning function. A brief overview of CAPP is provided. The research area, its relevance and importance are described in general terms. The problem to be addressed and the objectives of this research are described. The techniques to be used and the system to be modelled are described briefly.

A comprehensive review of the relevant literature is provided in the second chapter. This is divided into topics that are of interest and importance to this research. These are: process planning methods, GT input techniques, CAD input systems, prototype planning systems, dynamic CAPP systems, AI in CAPP, alternate routes in FMS scheduling, and CAPP application areas. This includes a comprehensive review of all process planning systems that are found in the published literature.

The third chapter describes the methodology used in this research. The construction and development of a unique, custom designed part coding system, the software structure, and the heuristics used to actually decide between the alternate routes available to process a part, the strategy used to track system status, and the simulation module are described. The software was coded in Prolog. The rational, reasoning, and the advantages of this choice are explained. A picture of the problem solution strategy is provided.

Chapter Four describes the pseudo-system under consideration. This chapter addresses the research assumptions, actual shop construction, machine descriptions, machine capacities and specifications, and other relevant shop con-

struction details. This chapter describes the shop's manufacturing capacity, the processes that can be performed, and the materials that can be processed. For each material type are given the possible speeds, feeds, and depths of cut. The method used to utilize these data in computing machining times and horsepower is described.

The software used in the process planning function is described in Chapter Five. The role of Prolog in the software schema is discussed. The design, structure, and the reasoning mechanism used are described. Stored in Prolog are shop capacity, capability, process information, machining data, machine specifications, etc. The technique used by the software to decipher the inputs is presented. The inference mechanism that reasons through to arrive at required outputs such as alternate routes and methods to produce a part, detailed process plans, and appropriate machining parameters is described. This chapter also considers possible auxiliary outputs that can be derived through use of this system.

The sixth chapter addresses the technique used in maintenance of the dynamic shop status in software. This software segment maintains and updates shop status, and applies the heuristic to arrive at the best route for the job through the system, and suggests the route to be used. It performs the computation of the machining times, the job flowtimes, etc. The organization of this segment of the CAPP system is decribed in this chapter.

The seventh chapter describes experimental results and process plans derived through planning test parts through the prototype system developed. This chapter describes in detail the outputs generated for these parts, and their significance. It

helps analyze the utility of this CAPP system, and the importance of the approach used by this research to apply CAPP in an FMS.

The eighth chapter presents a summary of this research effort. It discusses the outputs generated, importance of this CAPP system, the significance of generating and choosing between alternate routes in an FMS, and the importance of using this approach to CAPP. It details the accomplishments of this research. Also presented in this chapter are numerous enhancements and possible expansions to the system developed. Ideas for further research within the spectrum of CAPP are presented.

1.6 Conclusion

The importance of CAPP in a modern manufacturing facility cannot be over estimated. CAPP provides a direct link between design and manufacture. It reduces the time spent between part design and actual manufacture. Its obvious advantages as stated by several researchers make it impossible to ignore [8,9,10,67,68,85]. This research implements CAPP concepts in an FMS. This includes the design of part input techniques, design of appropriate software, system design, implementation, testing, and validation.

The rapid introduction of FMS systems in modern facilities, and the invaluable position that these systems would enjoy in the factories of the future make it a very important area to design and implement CAPP systems. This research designed, developed, and implemented a CAPP system for an FMS with multiple machining cen-

ters. Dynamic shopfloor conditions are considered in deciding upon the actual sequence of operations and the final job route. These conditions and the logic to arrive at the most appropriate solution were built into system software.

2.0 LITERATURE REVIEW

2.1 INTRODUCTION

Process planning techniques available include manual, computer-assisted generative, and computerized variant methods [12,68]. Topics germane to process planning and this research that are reviewed include:

1. Process Planning Methods.
2. Profile Input Using CAD Systems.
3. Profile Input Using Group Technology Coding.
4. AI In CAPP Systems.
5. Prototype Process Planning Systems.
6. CAPP Application Areas.
7. Dynamic Process Selection Techniques In CAPP.
8. Alternate Routings In FMS Scheduling.

It is evident from the literature that while some topics are addressed adequately, some have not been [67]. The review indicated several areas which have been re-

searched but still require additional development, and those that have apparently been neglected. Several topics are proving to be stumbling blocks in CAPP system development and implementation. It is also possible that current research has avoided tackling some crucial problems [68,71]. Both will be highlighted herein.

It is obvious that considerable effort is being invested in the integration of CAD and CAM through CAPP [8,9,10,85]. This should reduce the time for a job to proceed from the drawing board and conceptual design stage to actual production. It should also cause designers and process planners to work together, designing parts with ease of manufacture as a primary consideration. It should also cause the development of a unified data base that could be accessed by the process planner and the design engineer [10,12]. It would be ideal if the CAPP system developed a comprehensive process plan, determined the actual route and machines to be used, and then produced machining instructions. This could then be converted into detailed manufacturing (machining) instructions in a format that could be directly loaded into the controller of a computer numerically controlled machine. This would help in the development of a totally automated factory, and help close the link between CAD and CAM.

The recent, rapid development and implementation of knowledge based expert systems will certainly have its effect on CAPP. There have been a few research efforts in that area [13,14,44,45,72]. CAPP is an ideal area to use expert systems [44,72]. The AI approach with its elegant data representation and retrieval techniques, coupled with sound decision logic programmed into system software should have widespread applications in CAPP in the future. Applications could include processes such as metal forming, welding, casting, lesser used processes such as

broaching, and traditional CAPP applications such as turning and drilling [67]. A major hurdle in the building of expert systems will be the gathering of reliable data in order to create good, comprehensive knowledge bases.

This literature review also shows that CAPP systems have to expand their horizons to serve a greater variety of manufacturing industries. When new applications are sought, it is almost certain that problems will arise that will result in the development of new techniques and methods to help in CAPP development. Current CAPP systems concentrate on machining processes, especially in processes involving turning and hole generation [67,71]. Recent CAPP system applications have included operations such as cold forging [58], and deep drawing processes [19]. Widespread application and increased use of CAPP systems would make their installation more viable financially, and help to increase productivity levels in all manufacturing industries.

It is obvious that there are numerous problems to be addressed in CAPP, even at the macro-CAPP level as dealt with above. The problems will certainly multiply when studied at the micro-level.

2.2 Process Planning Methods

There are essentially three possible different approaches to process planning [12]: manual, computer-assisted variant, and computerized generative. It may be

more appropriate to use a particular approach under a given set of circumstances [12,68].

2.2.1 Manual Process Planning

The traditional approach to process planning has been to manually examine an engineering drawing and develop a detailed process plan based upon knowledge of process and machine capabilities, tooling, materials, and shop practices [9,12]. The process planner, given a specific material, must endeavor to build a list of processes for converting the given raw material into its desired final geometry [68]. Manual process planning leans heavily upon the planner's knowledge and experience. It requires well trained, experienced personnel who are well versed in shopfloor practice [12]. A manually derived process plan will always exhibit the personal preferences and prejudice of the individual.

Chang [9,12] mentions the following disadvantages of manual process planning:

- The requisite experience for a manual process planner requires years of hard work to accumulate.
- The experience thus developed may not apply to new processes.
- Experience gathered would represent approximate data only, and would not be exact.

These disadvantages along with those already mentioned has lead to investigating automating the process planning function.

2.2.2 Variant Process Planning

A computer assisted, variant process planning system requires a catalog of standard process plans to be stored in a computer system [12]. It uses the similarity of parts to retrieve process plans. A plan that could be used to manufacture a family of parts is called a standard plan. The standard plan is stored in the database with the coding number as its key. The degree of detail on the plan can be varied to match shop needs. The part being planned is classified and coded using GT concepts. With the part's attributes coded, any standard plan that is similar to the the part being examined is retrieved. The plan must then be checked, edited, and corrected at a Visual Display Terminal (VDT) to convert the standard plan into a plan for the new part [9].

The variant process planning method has been addressed in great detail by Chang [9,12], and Steudel [68]. Both authors state the advantages and disadvantages of this method. One drawback is that an experienced process planner is required to construct, maintain, modify, and edit standard process plans that are retrieved. This process planning method uses the computer as a tool to assist manual process planning [68]. The planner's knowledge and experience are still the key factors in determining the quality of the resulting plans. A variant process planning system performs well if there are a small number of part families, and the environment remains static, without changes in machines, tools, or materials [67].

A variant system has two operational stages: a preparatory stage and a production stage [67]. The preparatory stage requires that parts be coded, classified, and grouped into families. Standard plans are created for each family, and are stored in a database, indexed by family codes. This is a labor intensive process. The pro-

duction stage occurs when the system reaches operational readiness, and new components can be planned. The incoming part is coded, and the software structure of the system is used to retrieve an appropriate standard plan. This plan is then edited, altered, and modified to suit the needs of the specific part. Any changes made at this stage will not effect the standard plan stored in the system. There are several variant systems that have been developed including : AUTOPLAN [79], CAMI-I CAPP [39], GENPLAN [74], MIPLAN [62,63], etc.

2.2.3 Generative Process Planning

Chang [12] defines generative process planning as a system that synthesizes process information in order to create process plans for parts automatically. This technique uses an automatic, computerized system consisting of decision logic, formulae, technology algorithms, and geometry-based data to determine decisions required to create a process plan to transform a part from its rough to finished state [9,12]. A generative system, upon receipt of model description, must generate a process plan without human intervention. This requires that manufacturing knowledge be captured and encoded into software efficiently.

Steudel states that a generative process planning system consists of two major components: a geometry based coding scheme to convert physical features and drawings into computer interpretable data, and software consisting of decision logic, formulae, and algorithms to compare job requirements to machine availability and capability [67].

The system must make relevant decisions regarding process selection, process sequencing, machining parameters, etc. A major obstacle being faced is the transformation of part data into computer readable format. This requires that the software capture process planning logic, use an efficient input mechanism, and incorporate a comprehensive, unified database in the system. Currently the term 'generative process planning system' is applied less rigorously than defined above. Systems with built-in logic are often called generative systems. An ideal generative system would be a turnkey system that contains all the decision logic programmed in software.

The advantages of generative process planning include [9,12,68]:

- Consistent process plans can be generated very rapidly.
- Plans for new parts can be derived as easily as for existing parts.
- Generative process planning does not require human intervention.

Generative process plan design is a very complex process [85] requiring the long term investment of man, machine, and time. Generative process planning techniques can be further divided as those that use forward planning, and those that use backward planning [68]. Forward planning involves modifying the workpiece until it attains the features required by the finished product. Backward planning involves starting with the finished product, and filling it in to obtain the shape of the unmachined workpiece. Each machining process is considered a filling process.

The input format in a system affects the ease with which the system can be used [67]. Systems with very complex input modes are difficult to use. Besides, the translation of part geometry from a CAD model to a specific input format may be tedious and difficult to automate [9]. Input formats used include GT codes and specially designed part description languages. The language can be designed to input all

necessary information about the part. A CAD model, as an input device to a generative system, would reduce the human element in the process planning procedure [67].

Decision logic is an important aspect of generative process planning [9,42]. It directs the flow of control in software. It determines how the process and procedures are selected. It requires that process capabilities and part requirements have to be adequately matched. Several researchers have addressed the role of decision logic as embedded in software control [9,12,42,67]. Types of logic structures as used in process planning are: decision trees, decision tables, and artificial intelligence [42,44,72]. The knowledge representation is linked directly to the logic structure used.

Several sophisticated, generative process plans are currently available. Most of them concentrate upon a specific manufacturing process, or can be used only for a specific part family. They include among others APPAS [90], ACAPS [65], GARI [15], and AUTOPLAN [79].

2.3 Profile Input Using CAD

Computer aided design has been described simply as the use of a computer in the design process [67,68]. Although a wide variety of applications qualify as CAD, a narrower view would be to consider CAD within the bounds of computer graphics as applied to the design process in general, and specifically to CAPP. The major focus in this respect has been on mechanical part design as an input to a CAPP system.

The early CAD systems were developed in the late 1950's and the early 1960's. Current CAD systems concentrate upon solids modelling, NC part programming, engineering analysis of objects, etc.

TIPPS was one of the early systems that used CAD input devices [9]. This system tried to transform a design display into a format that was acceptable to the process planning system. It linked directly to the CAD data base. The components in a CAD system could be post-processed into a format that was acceptable to TIPPS. TIPPS allowed the user to manipulate both two and three dimensional displays interactively. TIPPS also allows for interactive identification of part surfaces. A menu driven identification subsystem with graphics capability is used to identify machined surfaces. Types of CAD data stored are geometric data and surface data which are used to identify a machined surface.

A CAD design method oriented to manufacturing problems based upon the 'Delta Volume To Removal' (DVR) technique allows the user to define tolerances, surface finish, and other relevant information [9]. This method requires a substantial amount of information from the user after the solid model has been created. This method requires the user to modify the blank part created by a solids modeller through standard machining operations until all the surfaces have been created and the tolerances, finish conditions, and all other information has been defined. Eventually the solid model is available for use, and the machined surface information is also stored in easy to read files. The part is now seen as a series of machined surfaces connected by geometrical and positional relationships with micro-geometrical and technological features only. This separation is manifested only in the data that may be of interest in machining and control.

The system provides the user with three orthogonal views and an axonometric view to ensure the correctness of inputed information. All surfaces stored in the system are arranged as separate files. A separate technological file and a geometrical file are stored. The second stage involves the input of technological data, both positional and dimensional tolerances. At the end of this stage, the system makes all the cutting operations needed to create the solid model of the finished part. This system stores all the information for process planning and automatic generation of NC code.

A system to automatically decide upon the cutting parameters for machining complex parts that adjusts feedrate depending upon the cutting load based upon offline simulation using solids modelling was developed by Wang [80,81]. The system consists of three units: a solid modelling module, machinability database, and feedrate automation module. The solid modelling information is used by an offline simulation system to compute a feedrate for every NC instruction such that metal removal is maximized subject to the cutting force and other constraints. The system uses an initial baseline speed and feed from the database. Average values for horsepower, cutting force, and tool deflection for each motion are computed. The feedrate is adjusted to a maximum such that constraints set by the user are not violated. The offline simulation helps to experiment with a variety of cutting conditions. The adjusted feedrates will avoid tool breakage and reduce machining time.

Three dimensional models have been used to extract geometrical and surface information from a part [82]. The part is stored as a set of data, and a boundary representation structure is used. This requires that the faces of the part are bounded regions defined by edges which are in turn defined by their end vertices. The

boundary representation allows easy extraction of data about surfaces. A comparison of the final part and the initial part is used to identify the surfaces that need to be machined. In a boundary representation, since the part is described in terms of lower level entities such as face, edge, and vertex, a higher level structure is defined based upon these entities. Forward chaining is used to recognize part features using simple rules.

When features are recognized, they need to be analyzed to determine the machining precedences. A sequence is determined after analyzing possible constraints, and reducing the possible sequences due to the infeasibility of the constraint set. The constraints are further subdivided into absolute constraints caused by part design, and process constraints caused by technology. The CAD interface analyses only the geometric requirements. The reduced set of precedences are analyzed later by the process planning system to determine a feasible sequence. The system then determines the feasible approach directions to machine a part. Feasible directions are chosen based upon feasibility, locally and globally, and good machining practice. This system has been implemented for cylindrical surfaces.

ICAPP [20] uses a direct access to a product model database through the use of the Initial Graphics Exchange Specification (IGES). IGES permits the exchange of product descriptions between different CAD systems. IGES is a product definition and data exchange standard that contains information relevant to the drafting of objects that includes both geometric and non-geometric information. IGES requires pre- and post-processors to form a system. The post-processor must convert the IGES formats of entities into a format that is suitable for use by the CAPP system. This requires that data transmitted by the IGES system be transmitted into a common database. A

graphics routine then transforms this data into an identical drawing as in the CAD system. The third step provides the actual CAPP interface with the required information from the IGES data. The use of IGES solves the problems of data transfer between different CAPP systems. It enhances the flexibility and capability of CAPP software.

Several systems use CAD interfaces to describe parts and define their geometric and surface specifications. Among the early systems to use a CAD interface was TIPPS [9]. Other systems include AUTOPLAN [79], ICAPP [19], SHAPES [25], XPLANE [77,78], BITCAPP [21], POPULAR [61], TURBO-CAPP [83], SAPT [43], EXCAP [88], and KAPPS [32].

2.4 Profile Input Using GT

Group Technology (GT) has been identified as an essential element in the integration of CAD and CAM through CAPP [54]. As a manufacturing concept, GT has attracted considerable interest [50,62]. The original idea proposed by Mitrafanov is being used in a variety of applications [50]. GT identifies and exploits the similarity of parts and operational processes in design and manufacture [53]. GT classification systems help the manufacturer realize the total spectrum of parts under production. It has helped in eliminating design duplication, promote job standardization, speed up design retrieval, and assist variety reduction. In addition, the economic advantages of GT have been a primary cause for its extensive use. This is true especially in a batch type manufacturing environment, since batch manufacturers deal with a

wide product range [53]. It is estimated that 60 to 80 percent of all manufacturing is batch oriented [53]. Batch manufacture results in a multitude of parts being produced, and requires a large number of process plans to be stored. Under such circumstances, GT would represent an ideal method to integrate manufacturing and design. GT based component classification improves knowledge about components in production, tool requirement estimation, machine utilization, and helps in machine selection in the purchase of new machinery [24]. GT also helps in part variety reduction.

A part family in GT may be defined as a group of related parts that are similar in some specific aspects [24]. They could have similar geometric shapes or share similar processing requirements. Parts that are dissimilar in appearance may be grouped as a part family because of common production operations. As the similarity between parts and the frequency of their production increases, so does the financial viability of implementing GT [24].

Part classification and coding is an invaluable aspect in the successful application of GT techniques [24,51]. Classification assigns parts to predefined classes, and coding is the allocation of symbols to the classes defined in the classification. Part classification could be design oriented, product oriented, or a compromise between the two.

In a design oriented coding system, the similarity between product shape is taken as an entity to form part families [24]. Major shape oriented families are further subdivided using shape features either individually or in combination; arranging them in their order of importance or incidence [24]. A code is then devised such that it identifies the part comprehensively and provides it with an unique identification code. In

a production oriented system, the similarity in manufacturing processes is used to identify possible part families. In this system, a principal term of reference is the parts' structure, since parts with similar shape features will, to an extent, use similar production operations. This is not however a necessary condition, since a production oriented system is satisfactory if it is able to group parts into families that require similar manufacturing processes. Since a combination of both a design oriented and a production oriented system may not be completely effective by themselves, systems that combine both techniques have been devised. Such systems use a supplementary code in addition to primary design data to describe design characteristics.

Coding allocates symbols to the classes defined in a classification [26]. Coding can be alphabetic, alpha-numeric, or numeric. The code structure can be hierarchical or have a chain-type structure. Coding and classification systems use three basic parameters to classify components: shape, function, and manufacturing operations and tooling [24]. A combination of these can also be used in classification. The shape parameter indicates the overall shape and proportions of the part. It indicates the presence of features such as holes, threads, and gears. The function parameter indicates the component's function such as pulley, gear, etc. The manufacturing operations parameter is aimed at the rationalization of production methods and to help in process planning. It will change as manufacturing techniques change or improve. Some systems use two or more of these parameters in combination to describe a component.

Two systems used widely in industry are the Opitz and the Brisch systems [53]. The Brisch system is design oriented, but the Opitz is a compromise between design

and production. The Opitz system uses a nine digit code, supplemented by a four additional digits to reflect the specific organization's needs [50]. The Brisch system uses a design oriented, hierarchical coding structure [53]. It uses a basic code of four to six digits. The user can add specific secondary polycodes to augment the primary monocodes.

Most CAPP systems use GT classification and coding techniques to retrieve standard plans, identify part surfaces, and organize appropriate manufacturing processes. Systems that use GT include MIPLAN, GENPLAN, AUTOCAP, AUTOPLAN, and ICAPP. The use of GT began in process planning with the development of standard process sequences for a family of parts as in generative process planning. Variant process planning systems also employ GT techniques. In these systems, GT is used to retrieve appropriate manufacturing processes depending upon the GT code through complex decision logic [10,67].

Ham states that a practical and efficient GT classification system must meet the following requirements: be all embracing, mutually exclusive, based upon permanent characteristics, specific to the user's requirements, adaptable to change, easily computerizable, and must have the potential to be used everywhere in an organization [24]. It is hypothesized that GT provides the standardization essential for the optimal use of CAPP [62,63].

2.5 CAPP Using Artificial Intelligence

Recent advances in the field of Artificial Intelligence (AI) demonstrate that practical AI based systems are possible [33,44,72]. Expert systems based on AI concepts are being used to solve specific problems that previously required a human expert. Although a wide variety of possible expert system applications exist within the purview of industrial engineering [72], Nau and Chang [44,45] state that a primary candidate for utilizing expert systems would be the process planning function in a manufacturing shop.

An expert system is a specific computer system that uses a knowledge base to solve specific problems that would normally require a human expert [72]. An expert system contains: a knowledge base, an inference procedure, and a working memory or a global data base [44,45]. A human expert usually collaborates in knowledge base development. Domain knowledge is stored as production rules. The knowledge base is called the rule base, and the inference technique is the rule interpreter.

Chang and Nau [44] state that automating the task of process planning could be done best through the use of AI expert system techniques. An expert system must organize information about each part surface. The system must then get the information that would be required about each surface. This information could be obtained from a CAD system or directly from the user. It would be better to design an interactive system that would ask the user only for required information in order to avoid the storage of redundant data. Such a system could also be interfaced with other parts of the CAPP system as they are developed.

Knowledge representation allows pieces of knowledge to be added, changed, modified, or removed from the system without having to modify other segments of the system [72]. Such a system allows spatial knowledge to be encoded into the system, and once the system was operating, it could be augmented by additional information. A list of processes could be created for a surface to be machined. The prerequisites and restrictions of each process must be considered before it is performed. A software control structure must search through the knowledge base until a satisfactory process is identified. The system must interact with the user until it has determined the process to produce every surface. The process plan as determined by the system is then printed out.

Wolfe and Kung describe the development of a generative process plan using an expert system methodology [87]. The system aims to extract form features of a part from a CAD system. These features, along with details about tolerance and surface finish, are placed in a data base which is used by an expert system to generate a process plan. In the prototype system which is still under development, features from the solid model input are extracted using software. Surface finish would be entered by the planner interactively. It is hypothesized by the authors that when the expert system is developed, it will use a knowledge base, a form feature data base, and an inference engine which would combine the information stored in the form feature data base and the knowledge base.

Perhaps the earliest plan generator to be developed using AI techniques was GARI [15]. It uses a knowledge base to automate the planning of the machining sequence for metallic parts. GARI did show that as the number of production rules increased, the possibility of conflicts between them also increased. This would require

appropriate conflict breaking methods to be enunciated. In addition, it is also possible that a rule may also have an exception which may lead to delays due to tedious conflict resolution.

An expert system was developed by Hummel to generate process plans to machine a part in one fixturing setup at a machine tool [29]. Hummel notes that in a totally automated factory atmosphere, a process planning system must be matched to a specific machine tool. He advocates the use of localized process planning capabilities for each control module in a distributed computing atmosphere. A system such as this would require a hierarchical network of independent modules whose planning spectrum would reflect the span of control at that level in the hierarchy. Each module would have a local knowledge base and a set of requirements that would be matched with the capabilities of an individual machine tool. It is imperative that each module in the hierarchy be able to access the knowledge bases of other modules in the system. The machine tool planner developed could work as a segment of a larger system, responsible for generating machining instructions required to process a part in a single fixturing setup at a machine tool, or as a stand alone system, dedicated to a single machine [29].

Hummel states that although this expert system has not been evaluated and tested, it has successfully planned for a wide variety of simple manufacturing operations [29]. The part programs generated by the system have been reliable within reasonable limits. However, this system is still in the early stages of development. As the system is expanded, capabilities and production rules could be added to the system. The system could evolve into one that is capable of planning complex tasks typically found in industrial situations.

SIPS uses AI techniques to perform generative process planning [46]. It considers each part to be a collection of machineable features. For each one of these features, it generates a sequence of machining processes that can be used to generate that feature. SIPS has been extended to perform tool selection, select cutting tools, and calculate process parameters. SIPS uses hierarchical knowledge clustering to organize information about surfaces in a taxonomic hierarchy. Each process in the hierarchy is represented by a frame. This system can read data from a file, or it can run interactively.

It is obvious from a survey of the literature that the systems mentioned above are only a small number of the actual prototype AI systems developed. Some other systems developed include IPROS [58], TURBO-CAPP [83], SAPT [43], XMAPP [31], KAPPS [32], and EXCAP [88].

In a review of AI approaches to the process planning function, Zdeblick states that the systems of the future will be more dynamic, flexible, and intelligent than they are today [92]. The successor to an automated system will be those that can make intelligent decisions such as what the most recent expert systems, for example, in the production scheduling, process planning, and facilities design areas can do. The successor to intelligent systems will be 'learning systems' that can monitor actual production experiences and feed them back to the system. This feedback will become the teacher. The systems will be able to learn from manufacturing mistakes and therefore improve performance. The advantages of a learning, self adapting system will include more accurate time and cost estimates, improved productivity, ability to monitor processes, less variability, more reliability, and less reliance on humans.

Subramanyam and Lu described the AI approach to process planning as a hierarchical structure that consists of two major branches: manufacturing logic, and part representation with the control structure providing a link between them [69]. The manufacturing logic consists of three stages: part comprehension, macro planning, and micro planning. Part representation consists of part description schemes, transitional methods, and geometric modelling. It is possible that an AI based process planning schema could be broadly classified as described above.

With the advent of intelligent manufacturing systems and AI based CAPP systems, the interface between the two has assumed importance [72]. There must be two-way communication between them. This will facilitate the generation of feasible plans for parts that the facility needs to produce. The CAPP system requires substantial quantities of data in different levels of detail. Chryssolouris and Gruenig describe a conceptual model that can be used to link an intelligent facility management system with the AI based CAPP system [13].

Joshi et al attempt to integrate a CAPP system with a CAD interface using an AI approach [36]. The CAD interface is developed with respect to automated feature recognition, the determination of tool approach parameters, and deciding the precedence relationships between features. Using a hierarchical structure results in the development of high level plans that are general in nature, and lower level plans that are very detailed in nature. At each intermittent level, sufficient detail is added to the plan. The first two levels form the CAD interface. It is at the third level that machine specific information is introduced into the system. The NC tape is generated at the final stage.

The CAD interface uses an available solid modeller to describe the part. A forward chaining, data driven recognition scheme is proposed in this paper to identify part features such as slots, pockets, steps, etc. The feasible approach direction is determined using available global and local information. The output from the CAD module is the input to the CAPP software. Appropriate machining processes are identified and selected. Process capabilities are mentioned in frames. A detailed set of rules are used to identify available manufacturing procedures, and precedences. A process plan is then created.

2.6 Process Planning Systems

There exist today both variant and generative process planning systems. Process planning systems have historically been developed by and for a single organization or in an academic atmosphere at an university. The following sub-sections will attempt to describe several CAPP systems that have been presented in the literature, and have concepts applicable to this research.

2.6.1 Automated Process Planning And Selection (APPAS)

Automated Process Planning And Selection (APPAS) is a generative system designed exclusively for parts that could be produced on machining centers. It is able to plan all processes that can be executed on these versatile machines. APPAS requires that needed process and planning options be defined by user input to the

planning program [90]. Chang [9], in a review of APPAS, states the primary difference between APPAS and other generative planning systems is the method used to describe surfaces. APPAS uses a special code to provide the technical data and information that describes each surface. A long string of attributes is used to describe each surface to be machined. The appropriate machining processes are selected through a complex decision tree structure.

Required inputs to APPAS include: surface geometry of the rough part, geometry of the finished part, and the manufacturing processes that could be considered in developing the sequence of operations, taking part tolerances and surface finish into consideration. This generative technique has the capacity to plan multiple passes and processes to machine a surface to the designated requirements. Outputs from APPAS include feed, speed, depth of cut, length of cut, machining time data, and tool particulars.

2.6.2 AUTOPLAN

Vogel and Adlard [79] describe AUTOPLAN, a CAPP system that incorporates a generative infrastructure to prepare a process routing sheet along with additional details such as jig and fixture selection, material and stock determination, etc. AUTOPLAN does not claim to introduce a computerized process planning system, but instead it helps the planner at crucial stages of process plan creation. The four major modules are:

1. Historical information retrieval from the database system using group technology coding techniques.

2. Interfacing graphical planning aids with the system to help the planner create tooling layouts, develop graphical work instructions, etc.

3. Generative process planning using the group technology coding to generate operation sequences, and tooling and machine recommendations.

4. Optimizing machining processes conditions to achieve minimum cost and maximum production rate.

AUTOPLAN's software is tied into a database that updates all files in the system and checks them for commonality. It is possible to interface this database with other corporate databases. AUTOPLAN attempts to combine a number of process planning activities into a single software system. It utilizes graphical capabilities to help the planner to visualize part and workpiece outlines, create turret layouts for specific parts, and simulate graphically numerically controlled machine operations. Process parameter optimization is also addressed.

2.6.3 CAM-I CAPP System

The CAM-I CAPP system is essentially a database management system. It provides the structure for a data base, appropriate retrieval logic, and interactive editing capability [39]. No fixed coding scheme is recommended, instead each user is encouraged to devise and use his or her own scheme. This is to promote the better adaptation of the plan to the user's specific needs and facility requirements. The objective stated prior to the development of CAM-I CAPP was the total automation of the manufacturing process planning, and possible interfacing of CAD and CAM in the future. CAPP could possibly be adapted to a variety of manufacturing processes.

While a centralized planning master source to maintain valid, current plans is essential, CAPP requires minimum computer resources.

The CAPP system relies on GT coding to retrieve parts where a family class criterion is used to retrieve a standard plan. The system allows for a thirty six digit alpha-numeric code input. The standard sequence file accessed by the code is retrieved, and if needed, is edited by the user. Hence, GT coding has been used in CAPP to retrieve standardized process plans. It was estimated in 1980 that over four hundred CAPP systems were in use throughout the U.S.A. [39].

2.6.4 Automated Coding And Process Selection - ACAPS

Emerson and Ham developed ACAPS in an academic atmosphere [17]. It is a semi-generative process planning system that operates in a time sharing mode using GT to code and group similar parts together. It has a modular structure, and process planning in this system is performed in several steps. Coding is the first step and is done based upon part geometry, overall dimensions, and material specifications. The software induces interactive coding of part surfaces. In the second step, parts are grouped as 'families'. A check is then carried out to insure that available machine tools have the capacity to carry out the required machining operations in the third step. Based upon the part's code, ACAPS suggests a machining operation sequence in the fourth step. Finally in the fifth step, an economic analysis of the production process is carried out based on existing quantitative data on which a 'make or buy' decision could be based.

The authors stated in 1982 that ACAPS was in its exploratory stages and could then handle holes, slots, plane surfaces, and turned surfaces. The authors expected to extend this system to gear production. The user is required to supply surface descriptors such as straightness, roundness, surface finish, threads per inch, presence or absence of chamfers, and any other relevant data depending upon the surface configuration to be machined.

2.6.5 Interactive Process Planning System For Prismatic Parts (ICAPP)

ICAPP was developed to plan parts that would be manufactured on milling, drilling, and boring machines [29]. It is a feature-oriented process planning system where eight different geometric operations can be processed. Operations necessary to produce a part are selected by the system from the part's feature type, dimensions, and tolerances. Input information required for processing includes: general ICAPP system, technical, job, and process planning information. The system calculates cutting speeds, operation times, and expected job production time. The system takes into consideration material hardness and cutter specifications in these calculations. Input to this system is interactive with respect to the part description. It is possible to edit the process sheet prepared by the ICAPP system.

2.6.6 AUTOCAP

AUTOCAP is a system that was developed for interactive process planning of turned parts to aid the unskilled planner in producing a high quality, accurate plan

quickly [25]. It was designed to help small and medium scale industries to improve process planning procedures at minimum cost using a shopfloor-based mini-computer system. The process planning sheet produced includes tool selection, speeds, feeds, number of cuts, operations, handling, and estimated total machining time.

AUTOCAP requires the planner to decide which surfaces are to be machined. Data then has to be input about the part such as material type, cutting tool material, and maximum permissible roughing cuts. This procedure requires that the planner have basic knowledge of process planning. The planner then identifies the first feature to be machined, part dimensions prior to and after the cut, and desired accuracy and surface finish. The computer then generates the planning information including number of cuts required, finishing cuts needed, machining and part handling times. This information is given on a planning sheet which can be edited as required. It is designed for use by the production planning personnel, and this system has produced consistent and accurate plans faster than a manual process.

2.6.7 MIPLAN

MIPLAN is another system in which GT is used to construct a process planning system [62]. GT techniques are used to group and classify part families in this system. MIPLAN is described as a special purpose, interactive word processor, which allows a planner to construct and edit process plans from standard text material or from special instructions. The planner constructs a plan with the aid of a graphics terminal. The final plan is printed only when a satisfactory sequence of operations

has been developed. MIPLAN provides for storing plans that can then be retrieved at the appropriate time.

A planner using MIPLAN has four different options: a process plan can be created from scratch, an incomplete plan can be retrieved, a retrieved plan can be edited and used, or a plan for a part with similar coding can be used. Once the plan is finalized, then the planner can store it. It can then be retrieved and used by personnel who are not well versed in the art of process planning. Four different files are generated by the user, and they work in unison with the logic modules of the planning system.

2.6.8 GARI - A Problem Solver

GARI is a process plan generator that uses an expert system to automatically create process plans [15]. This system used large amounts of subjective and specialized knowledge programmed into the system to determine process plans for machined parts. GARI consists of a knowledge base and a general purpose problem solver. Knowledge is stored in system software as a collection of production rules. The process plan is generated from the inputed part model which must give a description of the part mentioning the holes, grooves, notches, etc. and provide all relevant technical and geometric data. This information is used to determine the required cuts. Each available machine along with its capabilities must be described to GARI.

A process plan is generated after a series of iterations and successive plan refinements. Each successive iteration in the plan generation procedure produces a

new set of assertions that apply to a set of potential cuts. The system iteratively generates assertions by applying production rules. A contradiction between assertions is immediately processed by the system. The system discards the previous procedure and updates the current solution. Software for GARI was written in Maclisp, and was tested on substantially complicated industrial parts. All generated plans were said to be acceptable. It is possible that as the system grows, the number of conflicts would increase thereby requiring the adoption of more complex conflict resolving techniques.

2.6.9 GENPLAN

GENPLAN is a generative process planning system used at Lockheed-Georgia to create a comprehensive process plan when supplied with a unique part classification code [74]. It is capable of quickly creating the work instructions for the manufacture of aircraft parts and subassemblies. The system stores in software the shop's capabilities, rules of manufacturing as observed at the plant, and manufacturing process details. It also captures the skill of the experienced aircraft technicians, combines this with state-of-the-art technology, and creates precise plans.

GENPLAN specifies the machine tools and the processes needed to produce the part after it analyzes part geometry, material type, and several other important parameters, prior to specifying the appropriate process. This includes determining the cutting tools required, sequence of operations, and machining times. The system can also produce the required tool orders for manufacturing a specific part. Two versions

of this system are used: assembly GENPLAN and fabrication GENPLAN. Both use the same structure, but different codes are required.

The coding system used at Lockheed was specifically designed to meet their needs. This code describes the geometry and the manufacturing characteristics of the part. System software then synthesizes this code to formulate an optimal manufacturing plan. After the plan is found to be acceptable, it can be printed or stored for future reference. Lockheed estimates that GENPLAN reduced costs in six areas: process planning, scrap, shop labor, material, tooling, and work-in-process inventory [74]. It also reduced the time required to produce a new plan. Lockheed proposes to integrate GENPLAN with a drafting system (CADAM) and a system to plan and allocate manpower resources to meet a specified schedule criteria. A totally integrated system, it is hypothesized, would go a long way in improving productivity.

2.6.10 SHAPES

SHAPES is a software package that develops approaches to integrate design and manufacture through interaction between CAD and CAM systems [25]. The processes, tooling, and machine tools to be used depends upon the inputed part profile. This system requires part information to be entered in terms of 'SHAPE' elements [25]. The elements are described in terms of machined features and raw material forms. The routines required to construct the part given these data exist in system software. The system then selects the required tooling and machines.

Information on the material to be machined and part size are used to determine the machine tool to be used. The capabilities and constraints of each machine tool

available for use are entered into the system. The appropriate machine tool is selected along with the collets, chucks, and centers to hold the workpiece. All the tools needed to carry out operations on specific turning centers are arranged in a library. Tools are chosen depending upon their job's shape, the material to be machined, and the required finish desired.

This system uses part geometry as the starting point to generate a process plan. The system could be further extended to generate programs to operate CNC machines. The modular nature of SHAPES makes it easy to enhance the system with additional modules. The system, in its present configuration, plans the sequence of operations for a part for a specific manufacturing facility.

2.6.11 A Totally Integrated Process Planning System (TIPPS)

The Totally Integrated Process Planning System (TIPPS) is a generative process planning system developed by Chang [9] that uses a CAD interface. Part description without this is apparently tedious and error prone. Model decomposition and shape identification are two apparently major problems that TIPPS addresses [9]. Model decomposition deals with the problem of separating the original shape into the sum or difference of other shapes. Chang [9] uses a simple algorithm to help in shape identification. TIPPS matches a facility's process capability with the processes required to machine a part. The required processes needed to machine the part are identified by TIPPS, and a process plan is devised. Each process is programmed into the system using Process Knowledge Information (PKI), a language used by TIPPS to describe each process. Process capabilities are built into the system through a

series of 'If .. Then ... ' statements [9]. TIPPS uses backward chaining, generative process planning.

A combination of AI and decision tree approaches are used in decision making in TIPPS. The processes required, as stipulated by the system, are matched against process capabilities. A search termination message is issued when the required plan is identified. The system selects the required processing tools. However, if no feasible solution can be identified, and no available process satisfies the requirements to machine surfaces, a warning message is issued to indicate that no satisfactory process sequence is available.

Input to TIPPS is in the form of a CAD boundary representation. Technical specifications including tolerances and surface finishes are needed. The surfaces to be machined are marked. Process determination and sequence selection are automatic. Machining parameters such as feed, speed, and machining time are determined. TIPPS may not be implemented directly on the shopfloor, but is a feasible generative process planning approach interfacing CAD with CAM using CAPP.

2.6.12 TOJICAP - A System For Rotational Parts

TOJICAP is a CAPP system for rotational parts that attempts to combine the advantages of variant and generative approaches [14]. TOJICAP retrieves the standard process sequence through part family classification and code, and the detail of each individual operation is generated automatically from the database.

The system was developed to be planner oriented, and any decision made by the system can be modified by the planner. The planner can use the system effectively without prior programming knowledge. The system develops process sheets acceptable to the shopfloor. This system stores standard plans and retrieves them when they are appropriate for a new part. If a standard plan does not match the manufacturing requirement of a part, the system can then be used interactively to generate a new plan. The planner then has to input every operation code, and plan the process, step by step, using the interactive module provided in the system software. This approach is more time consuming than the variant approach. TOJICAP does calculate the relevant cutting parameters, machining time, part production time, and estimates production cost.

2.6.13 MICRO-CAPP

MICRO-CAPP is a variant process planning system that runs interactively on a microcomputer [51] where all system commands are two letter mneumonics. The user communicates with this system to use the various features provided in MICRO-CAPP in real-time mode. Coding and classification of parts for the purpose of segregating them is done through GT techniques. Part surfaces are coded interactively in this system, and it is believed that this approach reduces possible errors in coding part surfaces. The software stores the part family descriptions in matrix form.

2.6.14 Geometric Programming In CAPP

Geometric Programming In CAPP is a microcomputer based effort that hypothesizes that this technique can be used to select the optimal machining parameters for a variety of machining operations [41]. A case study is used to explain this technique. The process plan can be devised with one of two possible objectives: minimizing processing cost or minimizing production time. However, this system requires the user to provide information on the machine tools' capabilities, that is, speed, feed, and horsepower. System software then attempts to attain the objective function, determining the speed, feed, and depth of cut. This procedure is then repeated for each step in the sequence of operations required to produce the part.

Geometric programming is a new technique used to optimize non-linear equations with or without constraints [41]. The constraints specified can be non-linear. The software used in this system might go through a maximum of twenty iterations. This process planning system does not determine the sequence of operations nor does it determine what machines are to be used. Instead, the planner has to make these vital decisions. Although this is intended to give a human touch, it does mean that only highly competent process planners can use this system.

2.6.15 XPLANE

XPLANE is a prototype knowledge based expert system that attempts to automatically select both machining operations and cutting tools based on a pre-selected combination of machine tool and tool sets for box type products [77]. XPLANE is a

CAPP system that serves as one component of an integrated process planning module that combines product modelling, process planning, and scheduling. The user interface utilizes a multi-window environment with a mouse as an input device. A database interface embedded in software helps access information irrespective of data category. A boundary representation solid modeller is used to describe the part to be produced to the system. Features that can be recognized by the system include hole, pocket, step, plane, slot, and sculptured surface. Machine tool selection is currently possible, and a module to design and select adequate holding devices such as jigs and fixtures is under development.

Machining operations are selected based upon unstructured knowledge stored in a knowledge base. XPLANE is able to select the best solution from amongst a set of possible solutions. It is possible to change and update the knowledge base. XPLANE contains facts related to the available machine tools, and rules that control the inference procedure. Software selects the sets of possible machining operations, cutting tools, and the machining sequence. A backward chaining method that incorporates the graph search method is used. A heuristic procedure is used to help identify the optimal solution. The knowledge-base editor helps the planner change, add, delete, and reactivate a deleted rule. It is suggested that systems such as XPLANE be used in small batch manufacture atmospheres.

2.6.16 PROPEL

PROPEL is a feature oriented process planning system that uses AI techniques to produce process plans for orthomorphic and non-orthomorphic prismatic parts for

a specific workshop whose capabilities and capacities are known. System input includes: part features, relations between features, facility description, and capacity [73]. The system uses knowledge stored in two separate knowledge bases to develop a process plan that consists of machining operations, tools and machines required, and machining cuts.

Part description is brought about through a generalization of manufacturing features. A manufacturing feature is a tool trace of a simple tool such as a boring head, or a multiple tool such as a set of milling cutters. Geometrical data is regarded as secondary information with respect to features and is coded as relationships among features or as attributes of features. Workshop description includes description of machine and tool types. The system decomposes the main problem into a set of subproblems that are then solved independently of each other. The sub-solutions are then combined to arrive at the global solution. The process plans developed by this system were verified by experts and have been reported to be satisfactory.

2.6.17 Semi-Intelligent Process Selection (SIPS)

SIPS uses AI techniques to generatively select machining operations for part creation [46]. It considers each part to be a collection of machinable features. For each feature, it generates a sequence of machining processes to be used to create that feature. This is achieved by reasoning about the intrinsic capability of each manufacturing operation. SIPS can perform process planning at the macro and micro levels. SIPS has been extended to determine the actual tooling required, and to calculate the feeds, speeds, etc.

SIPS uses backward chaining in problem solving. The selection of cutting tools is performed after process selection is done. The proper tool material has to be selected to ensure optimal production costs. The search technique used by SIPS mimics the reasoning capacity of a manufacturing engineer. This system can read data from an input file or can run interactively, and it can produce alternate production plans for a product. Currently, this system can be used for parts made of wrought aluminum. Future prospects include extending it to other material types as well.

2.6.18 BITCAPP

BITCAPP is a semi-generative system developed for rotational parts. It consists of six modules: input, drawing modification, process generating, parameter calculation, drawing generation, and file output [74]. Each part consists of a set of primitives that are in some relative positions with respect to each other. If a standard drawing file similar to the part is available, it is recovered, and adapted to suit the part being planned. If no standard drawing is available, then a part drawing is generated using primitives directly from part models. The drawing is then stored in a file.

The inputed primitives are analyzed to develop a process plan using backward chaining concepts. The time required for each operation, and the machining parameters are determined considering part material, surface finish desired, tool life, productivity, etc. Theoretical optimums for each parameter are determined. They are then compared with the capacity of the machines available, and the actual values of the cutting parameters are determined. The operation drawing that illustrates the

actual process involved in transforming the raw material into a finished part is generated.

2.6.19 POPULAR

POPULAR uses a two dimensional CAD drawing as input to the system [61]. To this is added interactive input that the user needs to provide. The user designates shop type, and defines the actual machines that exist in the shop, their capabilities, and their capacity. The system evaluates all the information provided, decides upon the operations to be performed, and prepares NC data for each machine. POPULAR can process data for turning, drilling, boring, grinding, and gear cutting.

The output created includes cycle times for all processes, a process planning sheet for each process, a process planning sheet for each part, and a NC machine control output. The system decides which machine to use, and the required tooling. An important aspect of this system is that the operator needs to approve all the important decisions that are made through software. The operator can override system decisions, and is responsible for all system output.

2.6.20 IPROS

IPROS is an interactive variant process planner that produces output that includes : process description, work orders, material orders, parts lists, and progress reports [57]. Main system modules are: process planning, time and cost calculations,

progress control, company data maintenance, and document printing. Auxiliary functions include a feature to search among the process plans in the database for suitable process plans, a search for drawings in the database, facility to input data from the shop into the system, analysis of plan provided, and maintenance of data specific to the facility.

This system can communicate with three data bases: order dependent drawing database, order dependent processes, and standard process. IPROS is reported to produce over 10,000 process plans per year with over 70,000 operations. An experimental system to integrate it with an MRP system has been prepared but not yet tested. It is hypothesized that the training for the personnel involved has been minimal. IPROS use improved facility efficiency.

2.6.21 TURBO CAPP

TURBO CAPP contains five modules: machine surface identification, a process selection and sequence, NC code generation, knowledge acquisition, and data base management [83]. Surface identification includes geometric and technological features. The process selection and sequence is identified by using a knowledge base and an inference engine. The knowledge base stores data on available production facilities, rules to be used in sequencing, and rules for knowledge manipulation. The inference engine uses a backward chaining system to arrive at logical process sequences.

The NC part processor creates a NC program for each planned workpiece based upon its geometric features. The NC codes created are stored in a generic format,

and software can be created to transform the code into machine dependent NC programs. The knowledge acquisition routine assimilates necessary information that may be required from the user. The tolerances and finishes required are absorbed through the use of a tolerance input routine. The machines in the facility are described to the system through a machine description routine. The process manipulation routine allows the user to add new processes or delete existing routines. The data base is managed as a relational database in which all relevant details regarding a part are stored. It is proposed that this system can be extended for FMS use.

2.6.22 SAPT

SAPT is a part of the DESIGNER expert system that was developed for conceptual design of manufacturing machines, process planning, and production control [43]. It can handle both prismatic and rotational parts. The system's main structure is made up of three parts: pattern recognition, functional logic, and optimization. Parts are classified into families, decomposed, and compared with stored knowledge to achieve geometrical pattern recognition. The next step is process identification in order to manufacture the product given its inputed characteristics. The appropriate processes are chosen subject to optimization criteria specified in software.

Knowledge organization for process planning is similar to human reasoning and the designer's cognitive process. The main reasoning strategy operators used are: part recognition, decomposition, coupling, and sequencing. While the acquisition of new knowledge is efficient, the entire system is still under development, but has been partially tested in an industrial setting.

2.6.23 CORE-CAPP

CORE-CAPP is a semi-generative, rule based system using GT concepts that can be adapted to specific facilities [38]. The system consists of the following modules: family searching, GT database, detailed part information input, part geometry display, main process plan generation logic, process planning rules, time standard data, process planning database, and report generation.

Each part category contains several part families that use similar plan generation logic structures. The logic can guide software through the stored knowledge to arrive at the appropriate procedures. The rule base contains rules that can select the appropriate processes and machines, and calculate the machining times. Rules are either dependent on part families or on operations. The main logic accesses the relevant rules depending upon the part profile. The rule base communicates with the time standard calculation algorithm software, the process planning database, and the main process planning generation logic. The system performs a three step procedure: GT coding, part family searching, and automated process planning. GT code is assigned to the new part interactively. The part family is then identified, and software logic identifies the actual process plan. The plans developed compared well with manually produced plans. Consistency and standardization were attained.

2.6.24 XMAPP

The XMAPP system treats product information as constraints on products' properties [39]. The dimensions of a product, for example, impose geometric constraints

on product shape. Solids modelling is used by this system to input part information to software. Form features that define the part are used to select appropriate machining operations. The operational sequence is determined based upon the relationship between form features.

The actual system is divided into three modules: user interface, process planning, and product management. The user interface provides graphical output, multi-window management, and menu interaction functions. The process planning module consists of three independent knowledge based expert systems. The product model management module performs the basic product modelling functions and helps in manipulation of form features and dimensions. This system designs the initial workpiece, and then determines the machining, clamping, and positioning surfaces. The next stage involves deciding on the actual machining operations and their sequence. This system has been experimentally tested.

2.6.25 KAPPS

KAPPS is a system that can be subdivided into four subsystems: CAD interface, decision making subsystem, knowledge base, and know-how acquisition system [32]. The decision making subsystem uses input data from a CAD system to recognize part shape. This is used to determine process parameters such as machine tool selection, cutting conditions, and fixture selection. Know-how and knowledge are represented in software as frames.

Knowledge acquisition into the system is interactive. The system cannot directly detect contradictions within its knowledge base because it creates a separate file for

each part. Depending upon the input, the process plan is determined using a forward chaining approach. The system has been tested experimentally.

2.6.26 EXCAPP

EXCAPP is a knowledge based system developed for rotational parts [88]. It can recognize, through a two dimensional CAD system, an object's surface that can be machined by a single tool. The feature recognition rules are hierarchical in nature, with the top level testing for all occurrences of the feature, and the lower level testing for individual features. It uses backward chaining procedures to plan the operational sequence. The knowledge base contains rules that are not redundant. It allows the addition, deletion, and modification of rules, and a rule change can effect several other rules.

When the system is installed in a specific facility, information specific to that facility has to be added to the knowledge base. This is kept separate from general information stored in the software. The user can override EXCAPP's decisions. The system supports a simple rule base editor. It has been interfaced with GKS to help system portability and to provide an advanced graphics environment.

2.7 CAPP Application Areas

The literature review indicates that areas of CAPP application are very restricted [67]. When new applications are sought, it is almost certain that problems will arise that will result in the development of new techniques and methods to help in CAPP development [71]. All CAPP systems that have been developed in academia and in industry, and reported in literature, have been developed for application in discrete part manufacturing atmospheres [67,68]. In a continuous industry such as chemical industries, plastics and polymer manufacturers, there exists no reported CAPP system [67].

Even in discrete part manufacturing, current CAPP systems have concentrated on processes such as hole generation using drilling or milling, or on surfaces generated by turning [71]. Several prototype systems such as TIPPS [9], APPAS [91], SHAPES [25], ACAPS [18], ICAPP [29], and TOJICAPP [93], that model processes such as turning, milling, drilling, and boring exist. These processes, though widely used in industry, do not constitute a complete, comprehensive list of processes that are used. They are often considered because the transformation of data and information about a part and its surface into information that the CAPP system can understand and utilize has been solved for these processes.

CAPP research has not considered other machining processes such as thread cutting, grinding, broaching, shaping and planing [67]. Information regarding the cutting capabilities and characteristics of these machines are available. It requires researchers to determine how the part profile can be inputed to the system, and that

information transformed into parameters that CAPP system software can identify. Software must then generate appropriate and detailed work instructions. This will require that information on the processes such as thread cutting etc. will have to be coded into system software. If CAPP can be used in a wider range of manufacturing processes, it will generate more widespread use of such systems.

Whenever parts are produced in sufficient volume using processes such as drill presses, they have to be clamped and located using jigs, fixtures, or holding devices. This requires that jig and fixture design be a part of any process planning system. Some systems do address the basic problem of workpiece holding and positioning, but none do so in sufficient detail [71].

There are no existing CAPP systems that have been developed to serve non-traditional machining processes. This includes processes such as electron-beam machining, electro-chemical machining, abrasive-jet machining, ion-beam machining, plasma-arc machining, and laser-beam machining [85]. Very few process planners deal with non-traditional processes such as those mentioned above. CAPP systems would be especially useful in this area because these non-traditional machining processes are not used very often, and very few planners are conversant with their characteristics.

Literature also indicates that there is considerable effort being spent upon the use of AI techniques in the development of CAPP systems. Several researchers have said that it would be ideal to develop knowledge based expert systems to aid the process planning function. However, a major drawback in the difficulty involved in the development of comprehensive knowledge bases [44,45].

Current process planning systems often deal with individual machines or processes [67]. They do not consider modern shopfloor machines such as turning or machining centers which can process several parts simultaneously using a variety of process. These machines will be used in the 'high tech' factories of the future. They will also provide the planner with more than one method to process a part. This will help the planner develop alternate job routes for each part, and adopt the route that is most advantageous after considering the performance measures that need to be optimized.

2.8 CAPP And Dynamic Process Selection Techniques

The primary objective of most process planning systems has been to develop a sequence of operations based upon technical and economic criteria [12,68]. This approach is static in nature and does not consider dynamic facility conditions. A versatile FMS would require a CAPP that is dynamic in nature, since it could possibly produce parts using several job routes using alternative machines and processes. Dynamic process selection must ensure the ability to to generate alternative, feasible plans using different routes.

Dynamic process selection involves the ability to decide which process would be used to manufacture a part, taking into consideration current facility status. This technique also assumes that alternative processes exist to machine a part. This means that a part that can be processed on a particular class of machines can also be processed on a different class of machines using significantly different manufac-

turing procedures. This technique can be used in facility configuration decisions to help decide the number of machines that are needed in the facility, whether a smaller number of faster, sophisticated machines or a larger number of slower and cheaper machines could be used.

Nof and Barash state that production rate and machine utilization are two criteria to be monitored in a facility [48]. It was found that when dynamic allocation of either the original plan or the alternative were undertaken, the production rate and the machine utilization improved. However, product quality may be dependent upon the manufacturing process used. In a particular facility, since most parts are not manufactured just once, but repetitively, it is a sound decision to prepare alternate process plans for each part. Nof and Barash illustrated the advantages of dynamic process selection with an example that demonstrates that this process resulted in a better production rate and machine utilization.

2.9 Alternate Routings IN FMS Scheduling

An FMS has been defined as a set of machine tools, material handling equipment, and WIP storage facilities that are under common computer control [64]. Wilhelm and Shin studied the influence of alternate routings in FMS performance [87]. Although an FMS does provide flexibility in terms of machine versatility and the ability to process a variety of jobs and batches, it can provide additional flexibility through the use of alternate operations to process a product implemented through the system's computer control. An alternate route can be used to avoid overloaded machines, use

under-used machinery, reduce machine queues, reduce job flowtimes, mitigate machine bottlenecks, and balance machine utilization [86]. These objectives assume importance considering the investment made in FMS installation and operation.

The flexible design of a production planning system has evolved due to the need to develop and produce product variants, reduce WIP and delivery times, and improve product quality [85]. The last few years have seen the development of JIT and OPT techniques to help in the effective management of flexible production systems. A high degree of automation has also been introduced. The data considered in the system are the materials to be processed and the resources applied to process the materials. A comprehensive data base is required to keep track of the flexible production system [91].

Wilhelm and Shin state that it is possible that when a job is planned, no alternate operations exist, alternate operations are directed dynamically as in dispatching in traditional jobshops, alternate operations are planned ahead to avoid bottlenecks, or alternate operations can be planned and directed dynamically [86]. An FMS with four machining centers was modelled. Three types of parts, each with four operations, visiting all four machining centers was planned. Operational sequences and machining times are given. The performance of the facility under all four control methods were evaluated. Performance measures evaluated include machine utilization, product cycle time, storage space used, and number of carts used.

An FMS's computer control gives it the capability to collect, retrieve, and utilize on-line information, helping it in making adaptive decisions [91]. Yao concentrates on the routing flexibility afforded by an FMS. It is possible that a part may follow several routes which differ in terms of operational sequences and machine combinations. A

measure called 'route entropy' is used to evaluate routing flexibility. This factor is used to determine job routes and processing priorities.

Kimemia and Gershwin used a network flow optimization approach to decide on optimal part routing through an FMS modelled by a queueing network [37]. The solution technique used generates routes for each job, and decides upon the rate at which jobs should be dispatched to job routes. The strategy used does not require the enumeration of all routes ahead of time. The techniques used in this effort require the computers used in FMS control to make effective routing decisions. The FMS is modelled as a 'M' machine facility producing 'N' part types. A mathematical programming approach is used. Decomposition techniques are used to break up the optimization problem into path generating sub-problems and a coordinating master problem. The result produced by the solution strategy is the proportion of parts that must be manufactured by each of the available paths. Performance measures monitored included queue lengths, machine utilization, and production rate.

Loading and scheduling in an FMS is addressed by Kripa Shankar and Tzen [64]. They attempt to minimize system workload unbalance and system unbalance coupled with job lateness. Four different dispatching rules were tested. AN FMS loading strategy can aim to balance machine processing times, minimize job movements, balance workloads, fill the tool magazines as densely as possible, or consider two or more of these objectives [64]. This research addresses the loading problem in an FMS with the bi-criterion of balancing workload among machining centers and meeting job due dates. A heuristic technique was compared with an exact, mixed integer non-linear mathematical model. A simulation model was used to compare the performance of different dispatching rules on shop performance.

Research on alternate route determination and utilization in an FMS has proved that system performance, efficiency, and effectiveness would improve if this technique were used [86]. It demonstrates the advantage that an FMS possesses in terms of productivity due to effective computer control and information use when compared with traditional jobshops [64]. Since these conclusions are based upon specific test problems, further research is needed. Performance measures such as machine buffers, WIP, machine breakdowns, etc need to be investigated. It is obvious that within a CIM framework, an FMS provides extensive quantities of information that need to be effectively used.

2.10 Conclusion

This literature review indicates that the last two decades have witnessed tremendous growth in the development of CAPP as a viable interface between CAD and CAM. The desire to improve productivity and decrease costs have developed the need to reduce the time spent by a part from preliminary design to actual manufacture. It is evident that enhancements in available computing power coupled with reductions in computer hardware costs have made it more cost viable to use CAPP. Recent trends in CAPP development indicate that the advent of knowledge based expert systems has greatly altered the scope of this research area. It is forecast that expert systems can be used to aid process planning involving most manufacturing processes.

It is certain that the near future should witness substantial efforts by researchers focusing on CAPP. Researchers in industry and academia strive to produce a generative process planning system that could work with very little human intervention. It is certain that in the future, research would employ an interdisciplinary approach involving experienced manufacturing process planners, design engineers, and computer scientists. A great deal of effort is required before a comprehensive CAPP system that truly interfaces CAD and CAM is developed.

3.0 PROFILE INPUT AND GENERAL OPERATING PROCEDURES

3.1 Introduction

Although the last two decades have witnessed the emergence, and increased implementation and utilization of CAPP systems in both academia and industry, current systems tend to operate independent of other CIM components [68]. Often, while different control strategies and mechanisms are used to control and manage various CIM components, the lack of communication between them can cause the facility to operate under conditions that are not optimal. It is imperative that a global strategy be used in order to ensure that CAPP systems work along with other control strategies to ensure effective and efficient system implementation, application, and utilization.

Since CAPP is a relatively very new research and application area with few systems actually in use, most of the developed systems are prototypes dealing with very narrow areas of process planning [67,71]. The prototype developed herein integrates CAPP with other mainstream CIM elements, simultaneously increasing its scope and application area. While several researchers have commented upon the importance of considering the dynamic shop situation in system control [37,64,87], this research implements that concept in a CAPP system.

A typical FMS cell combines the flexibility of general purpose machines with the high productivity of a transfer line [64]. FMS cells consist of CNC machining centers which may include: milling, drilling, or turning centers. All of these are flexible, versatile, high capacity machines [87]. Cells usually include inspection and cleaning stations. Most current systems are intended for machining and very few are set up for forming or welding. Typically, each FMS has its own materials handling system that can move all possible material types.

It is with this background that the importance of CAPP systems for FMS cells has to be understood. Batch manufacture is undertaken in an FMS. The processing sequence and setup of the machines varies with part changes. In such an atmosphere, with varying job types and lot sizes, process plans have to be generated or regenerated often. The FMS's composition is such that there are a fixed number of machines, the capacity, capability, and specifications of each machine being known. To maximize the return on capital invested in an FMS, it is necessary to take full advantage of the capabilities. Several researchers have documented the importance and the advantages of considering alternate routes in FMS scheduling and capacity planning [37,64,87].

A shop specific system that can generate quick, consistent plans for an FMS cell has been created. It takes into consideration the dynamic shop condition, and investigates its effect on the efficiency of particular sequence. Since it is possible that machines with similar processing capacities may exist, it is possible that several alternate routes can be generated. The optimal route needs to be chosen depending upon the performance measure(s) to be optimized.

This research's thrust was to integrate CAPP with other CIM constituents such as production planning, scheduling, sequencing, and loading. The goal was to develop, design, and implement a prototype CAPP system for an FMS that reinforces and helps in comprehensive CIM implementation. The integrated prototype system designed and implemented performs the following tasks: suggests a comprehensive part description method, recognizes the part to be planned for, determines if it has been encountered before, retrieves an existing plan or produces a new one, checks on current shop status, suggests a job route using a heuristic to decide between possible alternatives, and updates the shop information based upon the route suggested. The research's basic infrastructure is described schematically in Figure 4. It is obvious from this schema that the overall research objective was to design and implement a prototype CAPP system that achieves integration between the process planning function and various CIM facets through software information processing.

A software structure implemented in Prolog and FORTRAN, controlled through REXX was used in developing this prototype system. While Prolog performs the task of identifying the routes possible, FORTRAN is used to implement the heuristic to decide between the available alternate routes considering the dynamic shop status using the minimization of flowtime as the performance measure. REXX is used to

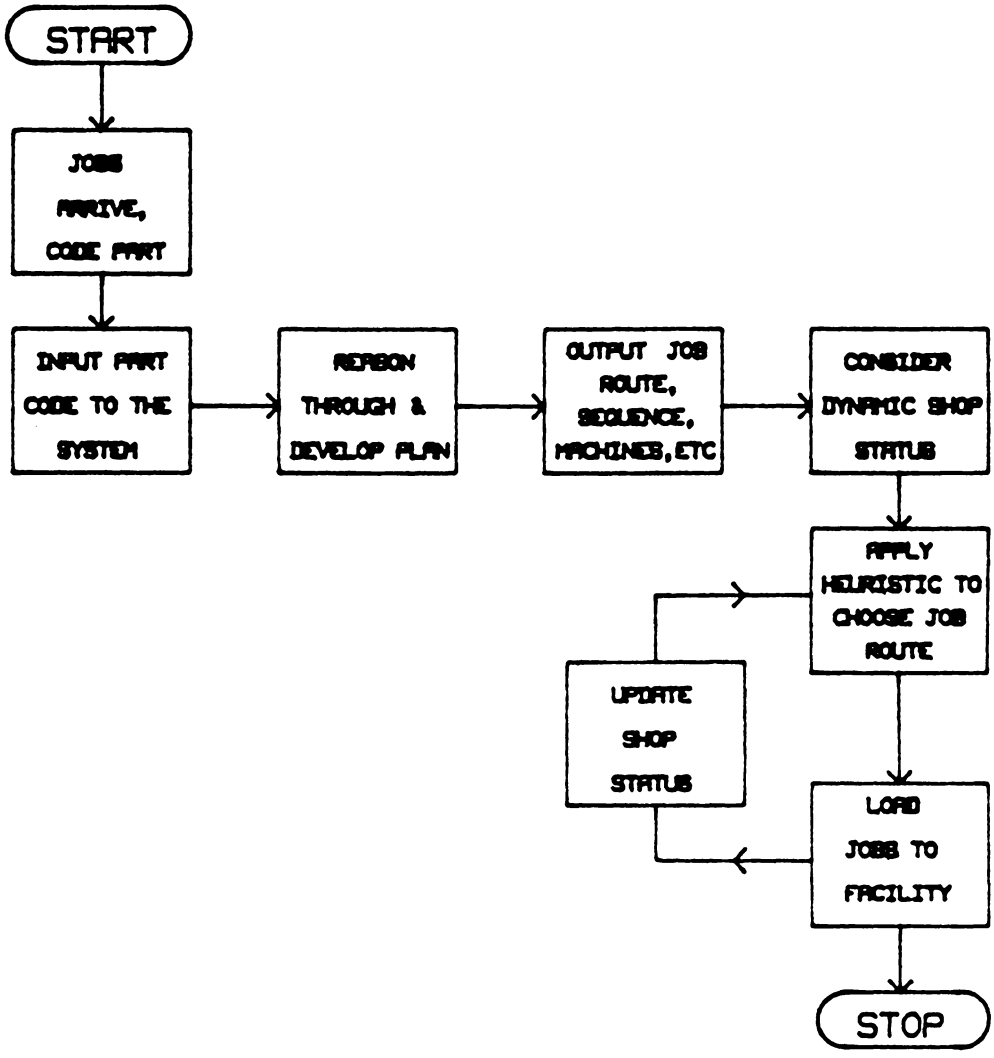


FIGURE 4 - BASIC RESEARCH INFRASTRUCTURE

control the actual execution and working of the software. The capacities and capabilities of each machine in the pseudo-facility, manufacturing hierarchies, precedence relationships, profile recognition methods, and properties of materials to be considered are stored in software, coded in Prolog. FORTRAN executes the numerical calculations to compute manufacturing times, consider shop status, evaluate times along various routes, and suggest job routes based upon the heuristic implemented. The methods used to implement each one of these steps in system software is described in detail herein.

3.2 Part Profile Input

In order to describe the part to be machined to the CAPP system, a custom designed, GT coding technique was designed. The part classification technique used facilitates a uniform method to describe parts to the software. Parts considered in this research are made of a single material, have a homogenous shape such as castings, half finished objects, and sintered components, and can be separated into new kinds of single parts by a destructive process. Welded, soldered, attached, or assembled parts are not included among the single components.

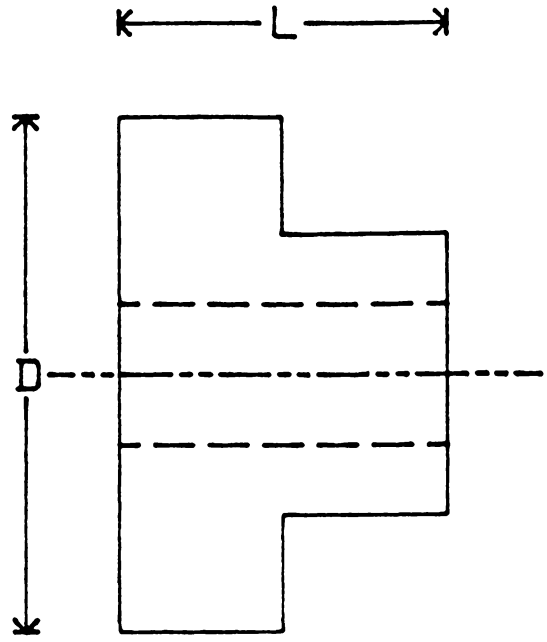
When a part is to be coded, its drawing has to be referenced. Each part detail then has to be described using the code designed for this purpose. The geometrical code utilized describes the part's final and initial shape, material type, tolerances, and dimensions. The part's initial shape is the least circumscribing cylinder or rectangular prism, oriented according to the axis of the main shape of the part. For ro-

tational (cylindrical) parts, the overall shape is given by a cylinder with dimensional ratio of Length (L) to Diameter (D). For cylindrical parts without deviations, the geometrical axis of this cylinder coincides with the rotational axis of the part. Non-rotational parts are enclosed in a prism of least volume that is described by the length of it's edges A, B, and C arranged such that A > B > C. An example using the code designed for this research for both a rotational and a non-rotational part is provided in Figures 5 and 6.

A standardized and uniform part description method incorporating a systematic technique helps in reducing the multiplicity of part components, recognition of repeat parts, standardization of shape characteristics, recording ease, overall effort minimization, collection and grouping of similar parts, and promotes simplicity of use. New products with similar designs can be planned for easily. The methods implemented to describe a part makes it possible to seek out the information needed concerning a part quickly.

The part description function falls between the design and the process planning stage. The coding process must define and describe a part and fit the description into a central drawing index. This index provides the software with a mechanism to recognize the presence of similar, or identical parts. The system designed and implemented promotes part standardization, uniform process planning, and manufacturing sequence generation. Figure 7 provides an outline of the coding technique used.

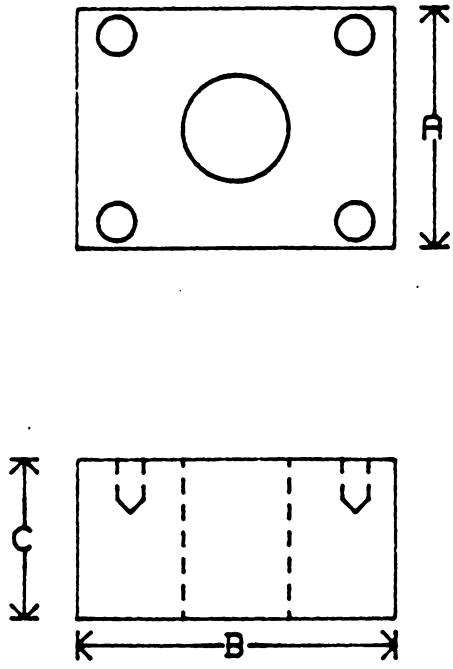
The hybrid code used contains five digits, with each digit representing a specific detail of the part under consideration. The first digit of the five digit code designed and used for this prototype describes the component class. This digit indicates if the part under consideration is rotational or cubic. If the part is rotational, the number



CODE

ROTATIONAL - 2
OUTER_STEPPED_ONE_SIDE - 2
INTERNAL_SMOOTH - 2
EXTERNAL_PLANE_NONE - 1
HOLES_NONE - 1

FIGURE 5 - CODING FOR A ROTATIONAL PART



CODE

CUBIC - 4

CUBIC_RECTANGULAR - 1

INTERNAL_SMOOTH - 2

EXTERNAL_NONE - 1

HOLES_RELATION_ONE_DIRECTION - 2

FIGURE 6 - CODE FOR A PRISMATIC PART

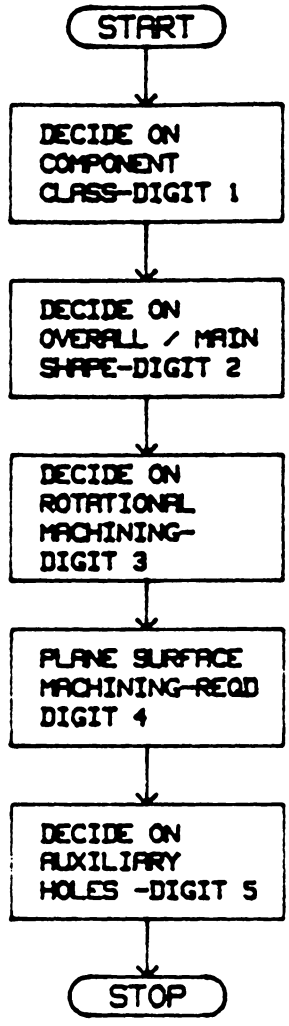


FIGURE 7 - CODING TECHNIQUE OUTLINE

used in the first digit describes to software if its L/D ratio is less than 0.5, between 0.5 and 3, or greater than 3. For flat parts, it differentiates between plates, cubes, and long parts. The A/B and A/C ratios are used for this purpose. If the A/B ratio is less than 3, and the A/C ratio is greater than 4, then the part is flat. If the A/B ratio is greater than 3, the part is long and narrow, and if the A/B ratio is less than 3, and the A/C ratio is less than 4, the part is cubic. This digit helps the software decide the machines on which the individual parts should be processed. If this digit indicates that the part is cubic, then it should be machined on a machining center. If on the other hand it is rotational, then it should be machined on a turning center. The significance of each number that is used in the first position in the five digit code is described graphically in Figure 8.

The second digit in the five digit code describes the part's overall external shape. It is used in conjunction with the first digit of the five digit code. This is because the significance of each second digit number is dependent on whether the part under consideration is rotational or cubic. For rotational parts, it signifies if the part has a smooth outer shape, whether it is stepped to one end or to both, or if a thread or groove exist on the part. If the part is cubic, then this digit differentiates between parts that are rectangular with no external deviations, rectangular with threads, and parts with grooves. The significance of digits used in the second position is described in Figure 9.

The third digit in the code describes the part's internal shape. If it is rotational, it can have either no internal features, internal thread, internal grooves, stepped to one side, or stepped to both sides. For cubic parts, it can have either no internal rotational work, one principle smooth bore, a principle smooth bore being stepped to

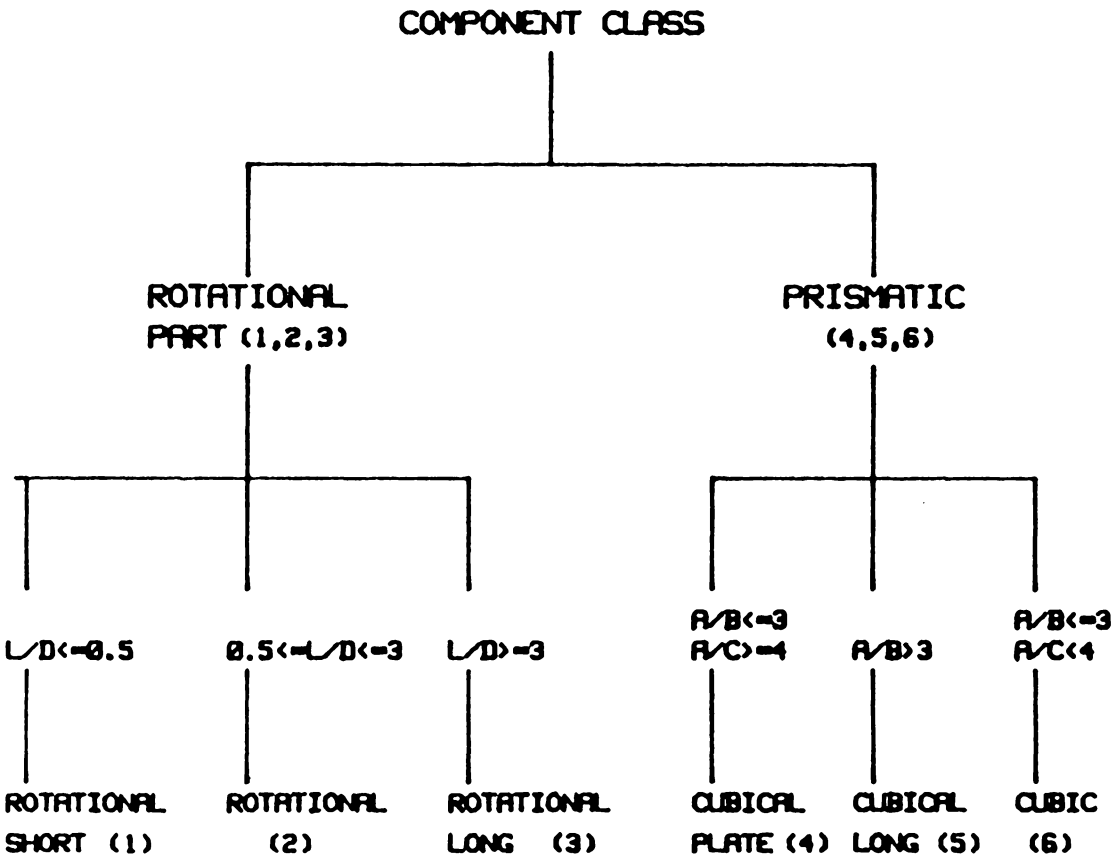


FIGURE 8 - DIGIT ONE-SIGNIFICANCE

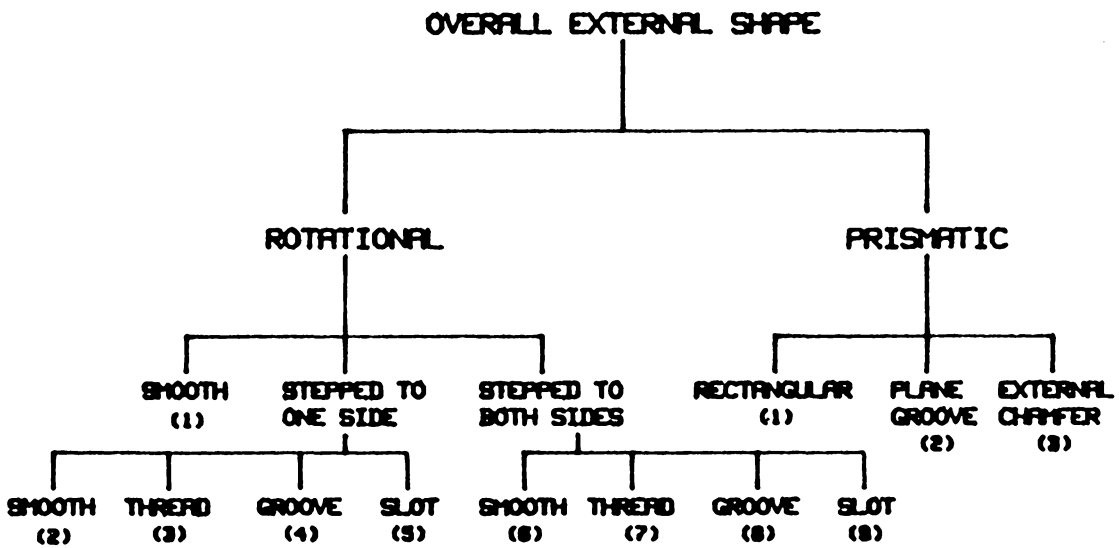


FIGURE 9 - DIGIT TWO - SIGNIFICANCE

one or both ends, two parallel bores, several parallel bores, or non-parallel bores. The significance of the numbers used as the third digit in the code is described in Figure 10.

The fourth digit represents the plane surface machining required for the part. If it is rotational, it might require no external machining, external plane surface curved in one direction, external plane surface related to one another by graduation around a circle, external groove or slot, and external groove and/or slot and/or groove. If the part is cubic, this digit stands for either no external machining required, functional chamfers, one plane surface, stepped plane surfaces, a groove/slot, or stepped plane surfaces inclined to each other at right angles inclined and/or opposite to each. The significance of the numbers used in the fourth digit of the five digit code is described in Figure 11.

The fifth digit in the code describes the presence of holes in the part. If it is rotational, it describes either absence of holes, presence of axial or radial holes related to each other by a drilling pattern, or axial or radial holes unrelated by a drilling pattern. For cubic parts, this digit describes parts with either no auxiliary holes, holes drilled in one direction only or in multiple directions, related or unrelated by a drilling pattern. The significance of the number used in the fifth digit of the five digit code is described graphically in Figure 12.

The part profile input technique used includes some non-numerical inputs to complete the comprehensive description of the part. The interactive inputs include material type under consideration, dimensions required for each feature to be machined, surface finish, tolerances, and the initial form of the part to be machined. The part dimensions required are inputed in inches. The material types considered in-

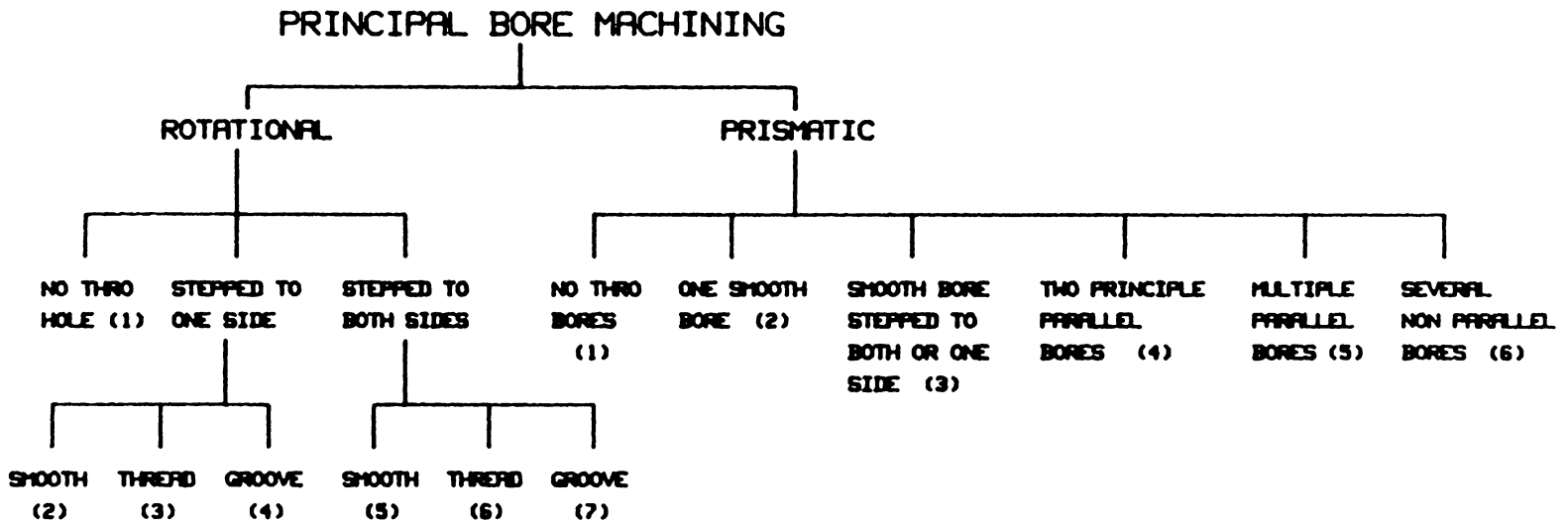


FIGURE 10 - DIGIT THREE - SIGNIFICANCE

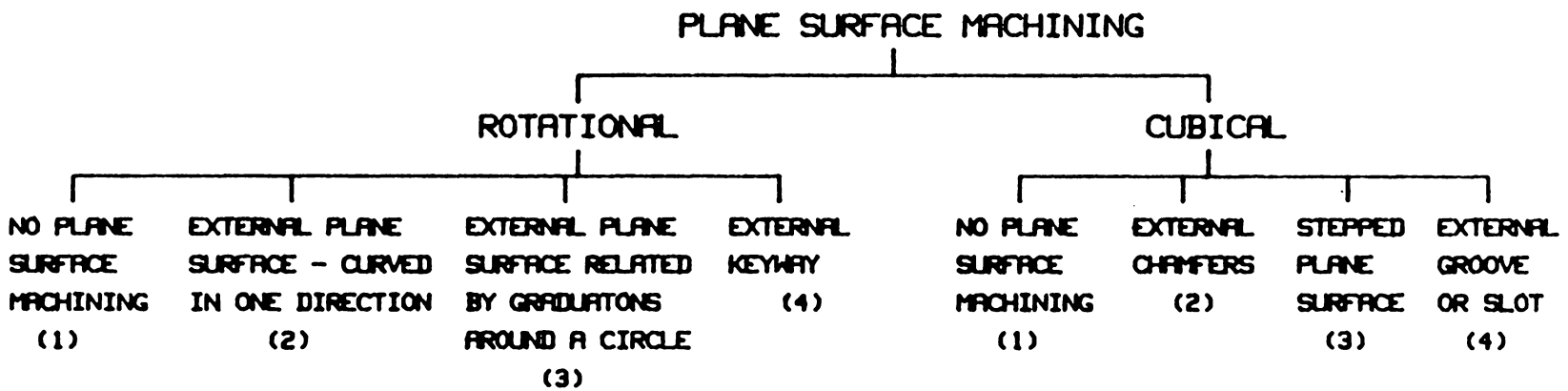


FIGURE 11 - DIGIT FOUR-SIGNIFICANCE

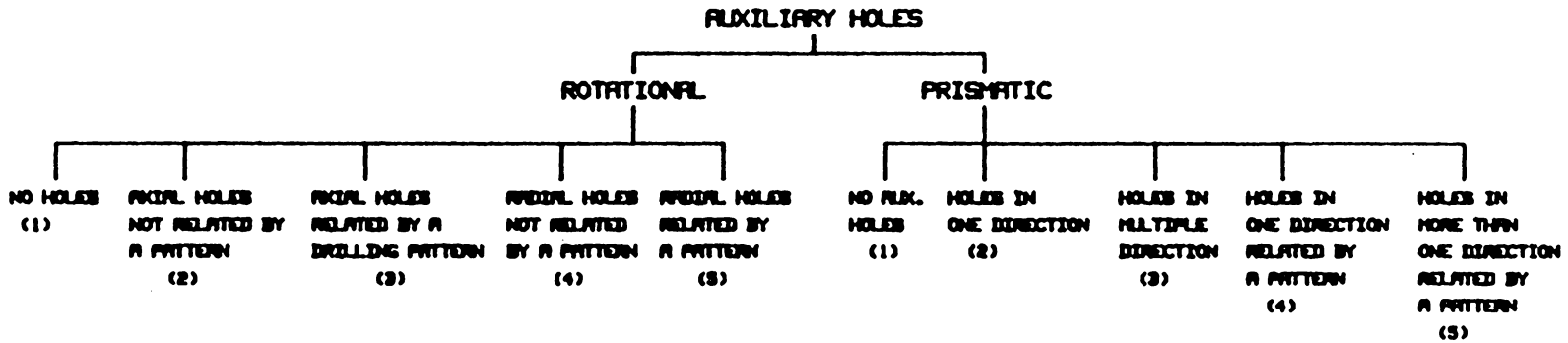


FIGURE 12 - DIGIT FIVE - SIGNIFICANCE

clude aluminum, copper, brass, cast iron, and stainless steel. The initial forms considered include slabs, bar stock, plates, and cubes. The surface finish can vary from 16 through 125 microinches. The tolerances desired are also interactively inputed. The tolerance and surface finish desired effect the operations used to machine a part and the final finishing operation used. The interactive inputs used are described in Figure 13. These interactive inputs are in addition to the actual five digit part description code used. They help to comprehensively describe the part to the system in a concise manner.

This prototype system assimilates this input along with other relevant inputs in order to develop a feasible process plan. Each digit inputed to the system by the user is employed by the software to decide an appropriate sequence of operations and machines. The input technique helps in standardizing the inputs made to the system by the user. The inputed code also helps to determine if the same part has been encountered before. This assists with retrieving a stored plan to ensure that the planning process is not repeated.

This research considers parts with features that can be described by the code and the other interactive inputs devised herein. It considers both rotational and prismatic parts. For rotational parts, features considered include external parts stepped to one or both sides, external chamfers, internal parts with through holes that are smooth, stepped to one side or both sides with a variety of surface finishes. It is possible to have the external surface machined with a variety of plane surfaces that includes surfaces related by graduations around a circle. Also considered are holes drilled through the part that are either radial or axial in direction. It could be single or multiple holes that may or may not be related by graduations around a circle. For

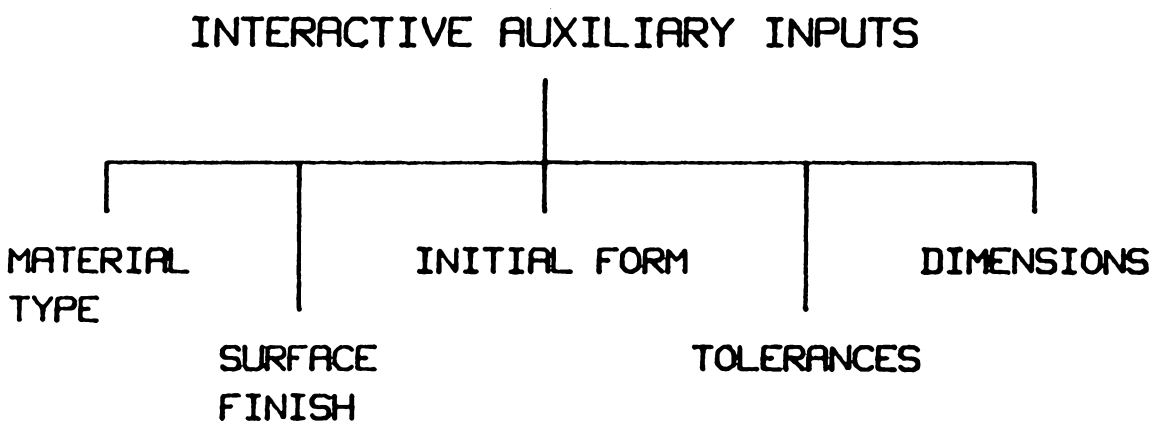
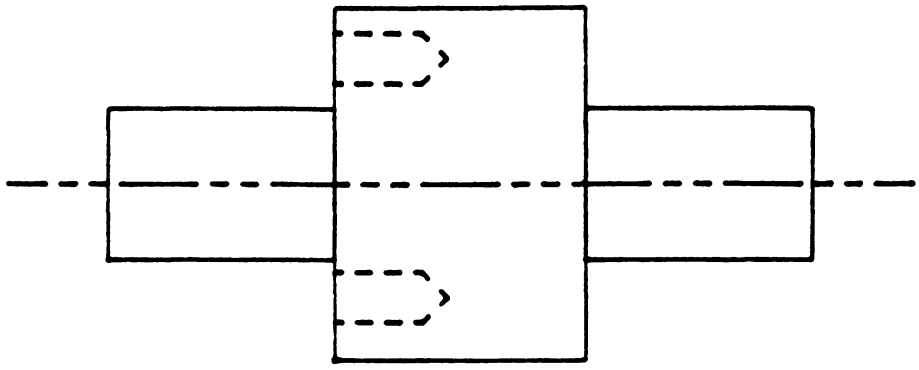


FIGURE 13 - INTERACTIVE SYSTEMS INPUT

prismatic parts that require external surface machining, the internal features considered in this prototype are similar to those considered for rotational parts. External plane surface machining is also considered. Presented in Figure 14, 15, and 16 are some examples of rotational and prismatic parts with a variety of features that are considered in this prototype design.

Parts, both rotational and cubic, with deviations are not considered. A part without deviations has only one axis of rotation, with the geometric axis corresponding to the axis of rotation. In a part without deviations, the cross section at any point is circular or annular or a combination of both. Rotational parts with external tapers and external shape elements are not considered. A shape element performs a definite function and poses a definite requirement on production. Examples of shape elements are external grooves for sealed rings, external tapers, etc. Parts that become rotational after metal forming processes are not considered. Internal tapers, external splines, external polygons, any type of gear teeth, and internal plane surfaces are not considered. Rotational parts that are bent to shape or rounded are considered as rotational parts with deviations.

Cubic parts with external shape elements are not considered. Internal annular grooves for cubic parts are not considered. External curved surfaces, guide surfaces, stepped plane surfaces, and any type of gear teeth are not considered. Parts with a bent axis are not considered in this prototype. Figure 17 and Figure 18 present some part features that are not considered in this prototype. It is evident from the description of features of the part family considered by this system that although all possible parts and features are not considered, a very wide and representative variety of features and part types are taken into consideration.



CODE

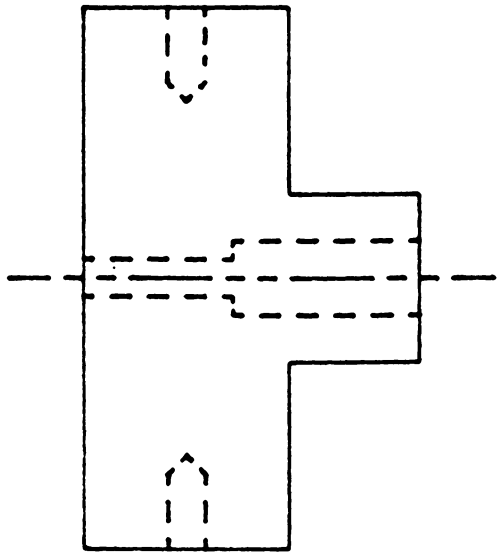
ROTATIONAL_LONG

OUTER_STEPPED_BOTH_SIDES

PLANE_NONE

HOLES_AXIAL_RELATED

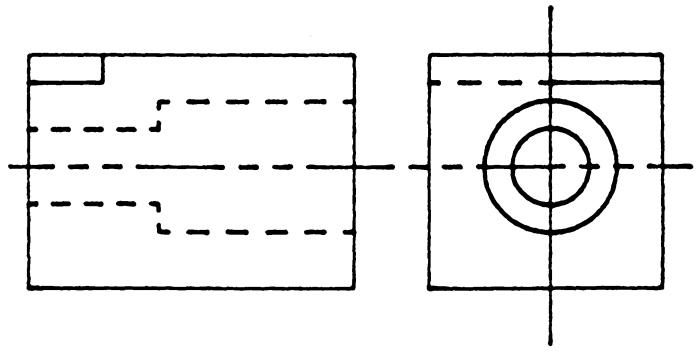
FIGURE 14 - A ROTATIONAL PART CONSIDERED BY THE SYSTEM



CODE

ROTATIONAL_SHORT
OUTER_STEPPED_ONE_SIDE
PLANE_STEPPED_ONE_SIDE
PLANE_NONE
HOLES-RADIAL

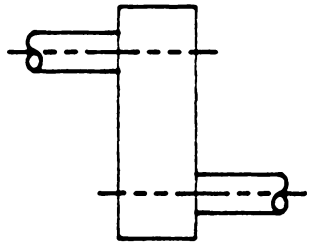
FIGURE 15 - A ROTATIONAL PART CONSIDERED BY THE SYSTEM



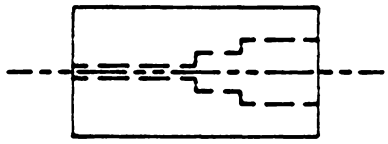
CODE

CUBIC - 6
CUBIC_RECTANGULAR -1
INTERNAL_SMOOTH_STEPPED_ONE_SIDE - 2
EXTERNAL_STEPPED - 3
HOLES - NONE

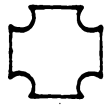
FIGURE 16 - A PRISMATIC PART CONSIDERED BY THE SYSTEM



MULTIPLE AXIS ROTATIONAL COMPONENTS

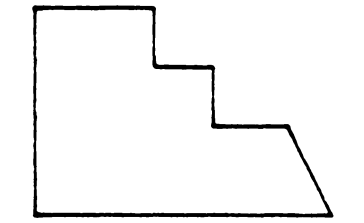


INTERNAL MULTIPLE INCREASE STEPS

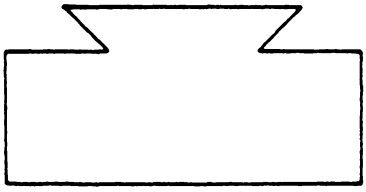


EXTERNAL SPLINES OR POLYGONS

FIGURE 17 - SOME ROTATIONAL FEATURES NOT CONSIDERED BY THE SYSTEM



PLANE MULTIPLE STEPS
INCLINED SURFACES



EXTERNAL GUIDE
SURFACES

EXTERNAL CURVED
SURFACES

FIGURE 18 - SOME PRISMATIC PART FEATURES

NOT CONSIDERED BY THE SYSTEM

It needs to be emphasized at this point that the five digit code developed provides information in sufficient detail for the prototype CAPP system. If commercial codes are used, it is possible that they may contain excessive information resulting in the input and storage of redundant information, or insufficient information for this application resulting in system failure. If however codes that provide sufficient information are used, they could be adapted for use in MACRO-CAPP with relative ease.

3.3 Process Planning Technique

After the part has been coded, the code is input interactively to the system. It is through software logic that the CAPP system recognizes if the part has been encountered on any prior occasion. If not, the system devises a plan for the part that is plausible and practical considering the part's characteristics, quantity to be manufactured, shop capacity and capability, and shop status. The output produced includes the processes, manufacturing parameter details, sequence of operations, and job route.

The process plan is devised through software coded in Prolog. The detailed program listing is presented in Appendix A. Stored in software is information regarding the construction of the facility, the capacity of each machine tool in the facility, the processes and procedures that can be executed on each machine, the materials that can be processed on each machine in the facility, and information regarding speeds, feeds, and other metal cutting parameters. Details regarding these characteristics and machine details are presented in Chapter Four. Also stored in software are al-

ternate procedures and methods that can be used to achieve a particular end. If, for example, a thread needs to be cut, it can be generated using a single point tool, a threading die, a tap, etc. This information helps and enables software to arrive at all possible alternate processing sequences.

It is through Prolog that the input code that describes the part is deciphered by the prototype CAPP system. Each digit in the inputed code and the interactive information is used by software to decide upon the processes that are needed to transform the raw material to the finished product. The reasoning strategy required for each segment of the manufacturing process is represented in the clauses in the software. The production rules are designed to assimilate the information input on each part. The reasoning and deduction mechanism, information storage and use, input and output techniques, part description and information assimilation methods, alternate route development, and all other relevant system development and implementation details are presented and described in detail in Chapter Five.

The first step is to determine what needs to be done to transform the raw material to the finished product considering factors such as material type, quantity, machines available in the facility, processes available, and finish and accuracy desired. This involved the design and implementation of a forward chaining technique to arrive at the raw form of the part from the final form as input to the system using the custom designed code.

Once the processes that are required to transform the part from its initial to final form are decided upon, then the machines that are capable of performing the processes required are identified. It is possible that several methods and/or machines may exist to machine a part from its initial to final state. Software generates all the

methods/sequences possible to process a part identifying all the feasible job routes through the system.

While all the steps stated up to now are performed through Prolog, FORTRAN is used to maintain shop status information. The role of FORTRAN is multi-faceted. While Prolog generates all possible routes and sequences that are possible, the information on current shop status is maintained using FORTRAN. The information generated through Prolog is written into an output file that is read by FORTRAN. The sequences and routes suggested are evaluated using appropriate formulae and techniques to determine the actual processing time that is required if a part batch were to be processed using each and every alternate route possible. This takes into consideration the processing time, shop congestion, Work-In-Process (WIP), and dynamic shop conditions.

The route that minimizes flowtime through the shop is chosen. The shop status is then dynamically updated assuming that the batch under consideration used the route suggested by software. This updated shop information is used to determine the optimal route considering all possible alternative routes when the next part batch is considered.

The role of REXX in the software structure is to coordinate the working of Prolog and FORTRAN. The output from Prolog contains all possible sequences that can be used. This is read in by FORTRAN to compute the flowtimes for each route possible considering current shop status. The interaction between the two languages, their loading, compiling, and execution is controlled by REXX. When a plan has to be devised for a part, REXX executes the Prolog portion of the software that generates the sequences, processes, and routes possible. REXX executes the FORTRAN segment

of the software using the output generated by Prolog as an input to this software segment. This reduces the human element in the software execution. An aggregate flowchart of the software structure in Prolog is provided by Figure 19. Figure 20 provides an aggregate flowchart of the FORTRAN software structure. The overall software mechanism is presented in Figure 21 which illustrates the role of REXX with respect to FORTRAN and Prolog.

3.4 Conclusion

This chapter provides a description of the techniques and methods that are used in this research to integrate a CAPP system devised for modern, versatile, and robust FMS with other CIM functions. A custom designed code is used to describe in detail to software the actual part construction. This information is used to generate all possible manufacturing sequences for the facility under consideration. The optimal route among alternatives available is chosen using a heuristic implemented through software. A software structure that involves a combination of Prolog and FORTRAN controlled through REXX is used.

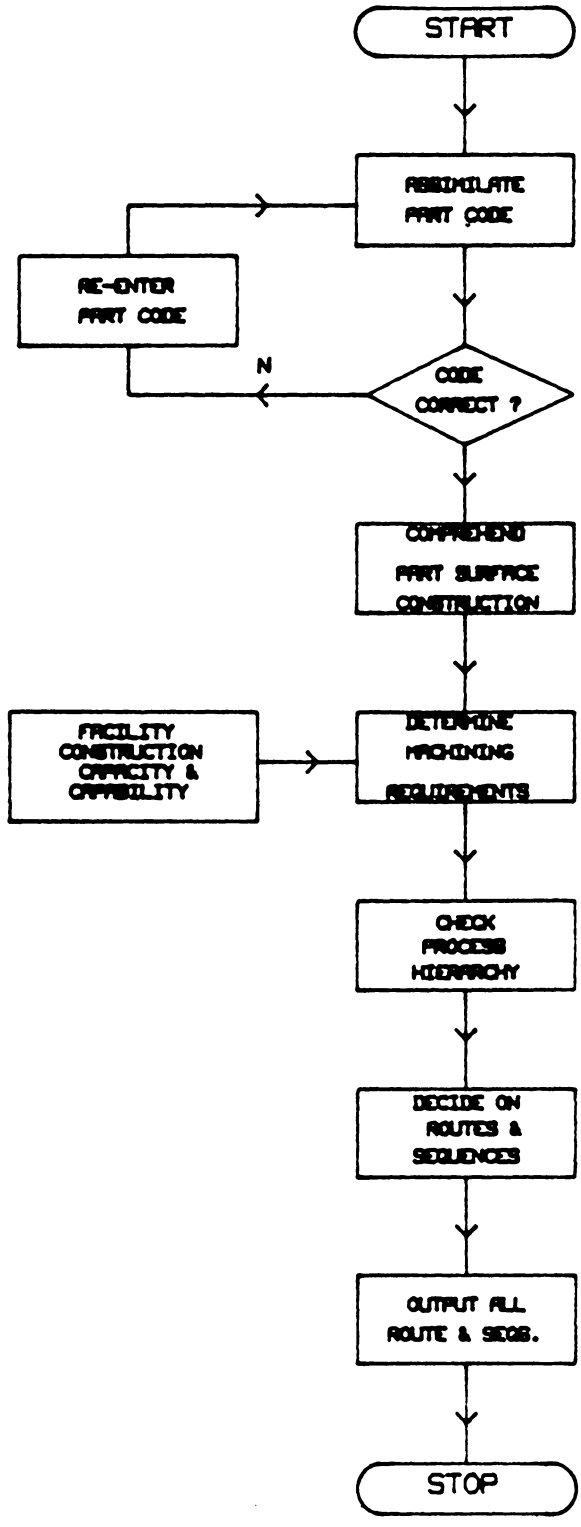


FIGURE 19 - PROLOG SOFTWARE LOGIC OUTLINE

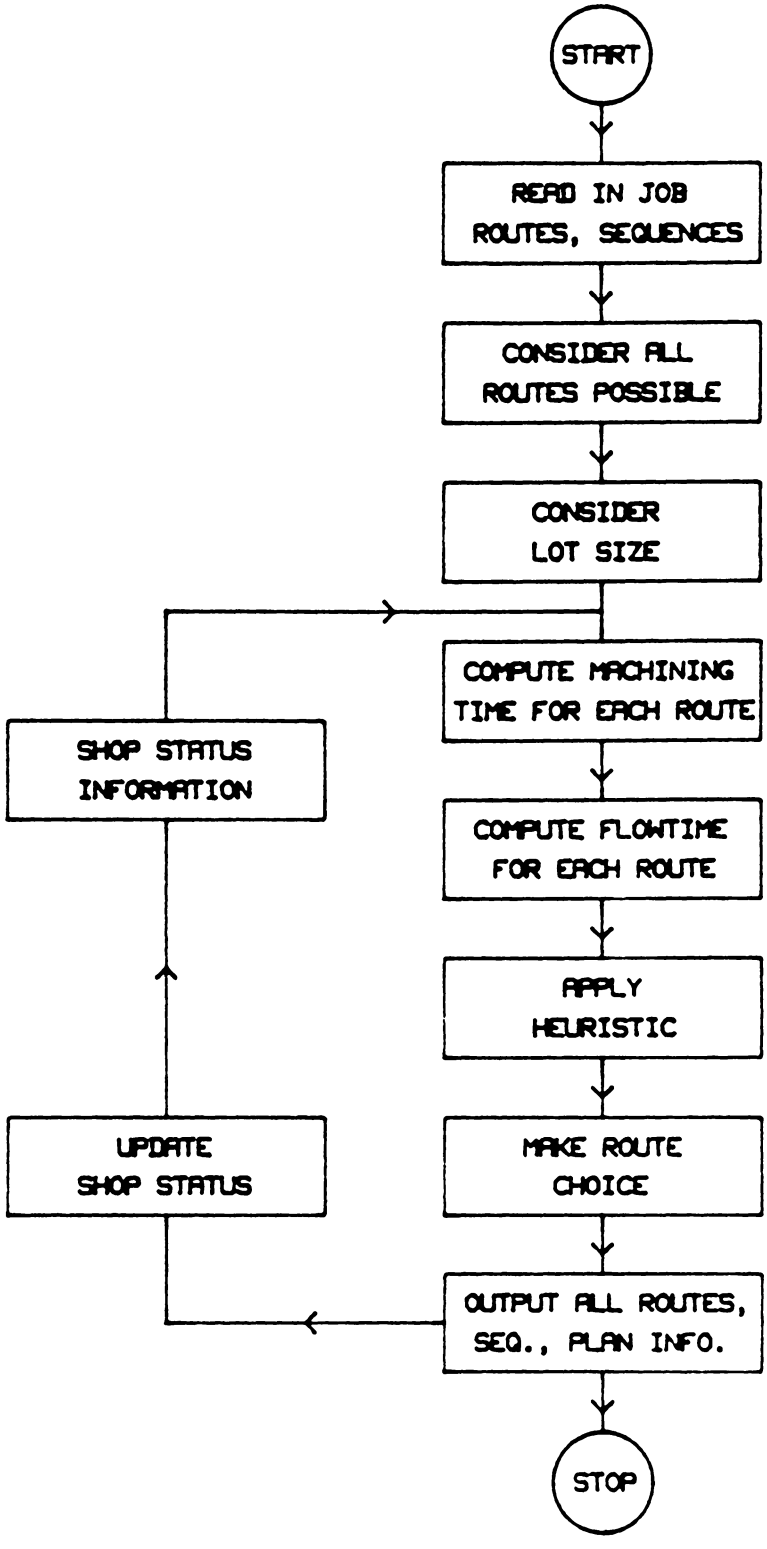


FIGURE 20 - FORTRAN SOFTWARE STRUCTURE

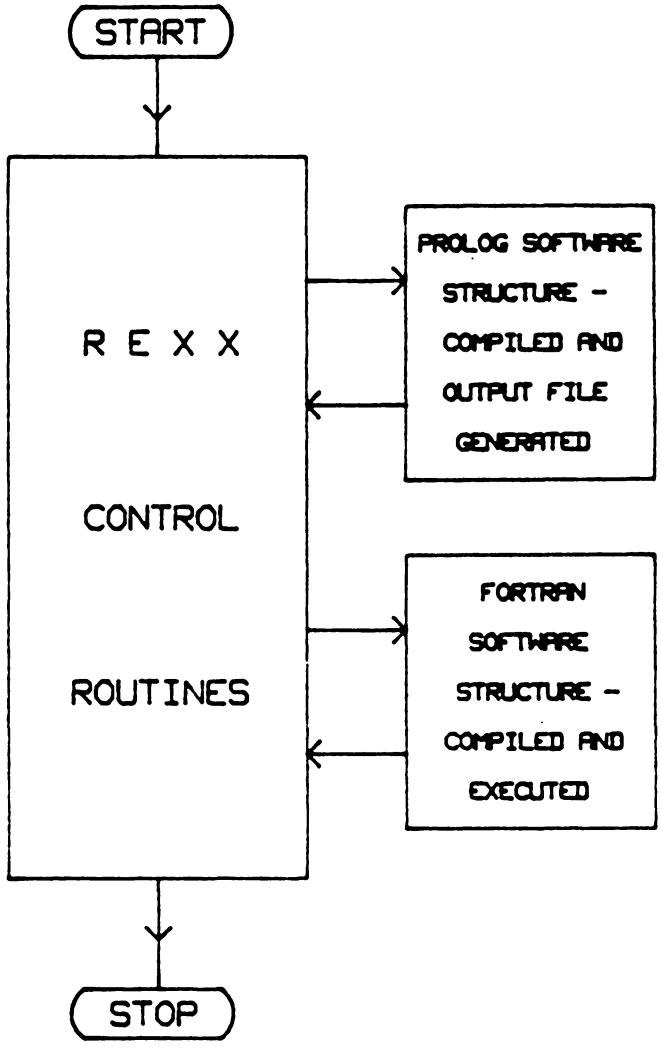


FIGURE 21 - REXX CONTROL STRUCTURE

4.0 SYSTEM DESIGN AND SPECIFICATIONS

4.1 Introduction

The system used in this research realistically models a feasible, representative, modern FMS facility. Assumptions are stated to make explicit the conditions that this facility operates under. They are sub-divided into job related, machine related, system related, control related, and process related assumptions. These assumptions help to accurately define and describe a system that is robust and resembles an actual, practical system closely. This ensures that the techniques and methods described, developed, and implemented were tested in a realistic atmosphere. Also described in detail herein is the type of system considered, methods used to model job arrival to the system, and the methods used to regulate job flow through the system.

This facility contains five state-of-the-art machining centers that are similar to machinery that would be used in the factories of the future. The capacity, capability,

and the specifications of each machine are described herein. The operations possible on each machine, the materials that can be machined, available speeds, feeds, and depths of cut are described. Also described are methods used to compute machining times, and horsepower required to machine jobs.

4.2 Facility Type Considered

The facility type considered for this research is a Flexible Manufacturing System (FMS) with five machining centers. The machines are Computer Numerically Controlled (CNC). The machines are linked together by a common, automated, materials handling system. The entire facility is integrated through common computer control. The flexibility of the CNC machine tools with automatic materials handling and computer controlled production management gives the system the flexibility of a batch environment with the efficiency of a mass production unit.

The job flow within the shop is entirely dictated by the machining sequence required to process the part and facility status. This indicates that the job could proceed to any machine with the required capability. The net flow in the shop is multi-directional; that is, jobs can follow a multitude of paths. Multiple entry and exit points exist for the facility. Jobs after being processed are transferred to the next machine that it has to be processed on through a computer controlled material handling system. The facility can handle very large to small batches. This versatile system patterns a modified flowshop/jobshop atmosphere.

4.3 Assumptions

These assumptions attempt to comprehensively describe the system that was considered in this research. These assumptions are under five headings: job related, system related, control related, machine related, and process related.

4.3.1 Job Related Assumptions

1. Each job is independent of any other.
2. Different job routes can be generated depending upon the processes that need to be performed.
3. Processing times are computable and are deterministic.
4. Batch sizes are known.
5. Batch sizes are discrete.
6. Batches are made up of identical jobs grouped together.
7. Batch sizes are not always the same.
8. Job due dates are not considered in scheduling and sequencing jobs.
9. Job setup times are known, and independent of prior conditions.
10. Arrival rate of jobs to the facility is known.
11. Job mix arriving to the shop can be controlled vis-a-vis the arrival rate of each job type.
12. A job may need to be processed on one or more machines.
13. Jobs can wait for processing in infinitely long queues.

4.3.2 Machine Related Assumptions

1. No machine failure is anticipated.
2. No maintenance is needed.
3. Machine capabilities, capacities, and specifications are known.
4. Machine setup time is known.

4.3.3 System Related Assumptions

1. The number of machines in the facility is known.
2. Job flow direction is determined by its sequence of operations.
3. The system is dynamic in nature.
4. System status at any point in time is known and continuously monitored.
5. A start-up period is allowed to enable the system to approach steady state.
6. Materials handling time is negligible.

4.3.4 Control Related Assumptions

1. No batch splitting or lap phasing is allowed.
2. No job interruption is allowed.
3. Jobs are not accelerated or expedited.
4. No scheduled slack time is considered.

4.3.5 Process Related Assumptions

1. A job, on entering the facility, may require more than one process to transform it to the finished product.
2. The processes that can be performed in the facility are the sum of the processes that can be performed by the individual machines.
3. The processes that can be performed on a specific machine are known.
4. A process is performed on a machine only after it has been confirmed that the machine has the requisite horsepower to perform the job.

4.4 Machine Specifications

The facility consists of five CNC machines: three turning centers, and two machining centers. No two machines are completely identical. Given below are details of the machines' specifications and capabilities. All machines are capable of processing all the different part materials that arrive at the facility requiring work. Material types that are machined and the data associated with material types are described later in this chapter.

4.4.1 Machine 1 - Turning Center - Type 1

Machine One is a multi-axis, CNC, machine with two, eight tool turrets. It can perform the following operations:

- Turning.
- Facing.
- Drilling.
- Boring.
- Thread Cutting - Internal and External.
- Taper Turning.
- Knurling.
- Cutoff and Parting.

The specifications of this machine are :

- Maximum cutting diameter = 8.00 inches.
- Maximum cutting length = 24.00 inches.
- Speed = 40 to 4000 rpm.
- Speed increments = 1 rpm.
- Maximum torque = 1143 foot pounds.
- Maximum Number of tools = 16.
- Tools Per Turret = 8.
- Number of turrets = 2.
- Maximum Horsepower = 30 hp. at 100 % efficiency.

4.4.2 Machine 2 - Turning Center - Type 2

Machine Two is a multi-axis, CNC, machine with two, six tool turrets. Six tools could operate upon the inner diameter, and six on the outer diameter. It can perform the following operations:

- Turning.
- Facing.
- Drilling.

- Boring.
- Thread Cutting - Internal and External.
- Taper Turning.
- Knurling.
- Cutoff and Parting.

The specifications of this machine are :

- Maximum cutting diameter = 8.00 inches.
- Maximum cutting length = 20.00 inches.
- Speed = 32 to 4000 rpm.
- Speed increments = 1 rpm.
- Maximum torque = 1072 foot pounds.
- Maximum Number of tools = 12.
- Tools Per Turret = 6
- Number of turrets = 2.
- Maximum Horsepower = 20 hp. at 100 % efficiency.

4.4.3 Machine 3 - Turning Center - Type 3

Machine Three is a multi-axis, CNC, machine with one turret. It can perform the following operations:

- Turning.
- Facing.
- Drilling.
- Boring.
- Thread Cutting - Internal and External.
- Taper Turning.
- Knurling.

- Cutoff and Parting.

The specifications of this machine are :

- Maximum cutting diameter = 12.00 inches.
- Maximum cutting length = 20.00 inches.
- Speed = 23 to 3000 rpm.
- Speed increments = 1 rpm.
- Maximum torque = 1143 foot pounds.
- Maximum Number of tools = 12.
- Tools working on inner diameter (maximum) = 12.
- Tools working on outer diameter (maximum) = 8.
- Number of turrets = 1.
- Maximum Horsepower = 30 hp. at 100 % efficiency.

4.4.4 Machine 4 - Machining Center - Type 1

Machine Four is a 20 h.p, multi-axis, CNC, horizontal spindle machining center. It is equipped with automatic work changing, automatic tool changing, and automatic chip removal. This particular machine is equipped with a 45 tool automatic tool changer. It can process non-rotational parts with the following operations :

- Drilling.
- Tapping.
- Side Milling.
- Face Milling.
- Plain Milling.
- Slitting.
- Shell Milling.
- Pocket Milling.

- Reaming.

The specifications of this machine are :

- Maximum part weight = 2000 pounds.
- Maximum part size = 32 inches * 26 inches * 26 inches.
- Speed = 20 to 4000 rpm.
- Speed increments = 1 rpm.
- Table index possible = 1 - 360 degrees.
- Maximum Horsepower = 20 hp. at 100 % efficiency.

4.4.5 Machine 5 - Machining Center - Type 2

Machine Five is a multi-axis, CNC, vertical spindle machining center. It processes parts that are non-rotational in construction. It is equipped with automatic work changing, automatic tool changing, and automatic chip removal. It can handle bigger, heavier jobs that its counterpart, machine 4, mentioned above. This particular machine is equipped with a 45 tool automatic tool changer, and can perform the following operations :

- Drilling.
- Reaming.
- Tapping.
- Internal Threading Using Dies.
- Side Milling.
- Face Milling.
- Plain Milling.
- Slitting.
- Shell Milling.
- Pocket Milling.

The specifications of this machine are :

- Maximum part weight = 4500 pounds.
- Maximum part size = 40 inches * 40 inches * 40 inches.
- Speed = 20 to 4000 rpm.
- Speed increments = 1 rpm.
- Table index possible = 1 - 360 degrees.
- Maximum Horsepower = 40 hp. at 100 % efficiency.

4.5 *Material And Cutting Parameter Information*

The facility considered can machine a very wide variety of material types. Materials considered include: aluminum; brass; cast iron - soft, medium, and hard; stainless steel; copper; three types of steel - less than 0.3 percent carbon, 0.3-0.6 percent carbon, and greater than 0.6 percent carbon. All the materials mentioned above can be machined on all the CNC machines under consideration.

The speeds used to machine parts depends entirely upon the material under consideration. Speeds, feeds, and depths of cut are important in determining the actual horsepower requirement for processing a job, and for the processing time computations. The following speeds, for drilling, milling, and turning operations, are the maximum recommended for purposes of this research. If carbide tools are used, it is possible that higher machining speeds could be used.

- Aluminum = 550 feet per minute.
- Brass = 280 feet per minute.
- Copper = 280 feet per minute.

- Cast Iron - Soft = 125 feet per minute.
- Cast Iron - Medium = 125 feet per minute.
- Cast Iron - Hard = 90 feet per minute.
- Stainless Steel = 90 feet per minute.
- Steel - Less Than 0.3 percent Carbon = 110 feet per minute.
- Steel - 0.3 - 0.6 percent Carbon = 80 feet per minute.
- Steel - Greater Than 0.6 percent Carbon = 50 feet per minute.

The feeds used in the actual machining processes depends upon the material. The following feeds are used in this research for drilling, reaming and turning.

- Aluminum = 0.045 inches per revolution.
- Brass = 0.045 inches per revolution.
- Copper = 280 feet per minute.
- Cast Iron - Soft = 0.025 inches per revolution.
- Cast Iron - Medium = 0.025 inches per revolution.
- Cast Iron - Hard = 0.018 inches per revolution.
- Stainless Steel = 0.020 inches per revolution.
- Steel - Less Than 0.3 percent Carbon = 0.020 inches per revolution.
- Steel - 0.3 - 0.6 percent Carbon = 0.018 inches per revolution.
- Steel - Greater Than 0.6 percent Carbon = 0.015 inches per revolution.
- Finishing Feeds For All Materials = 0.010 inches per revolution.

For milling operations, feed is expressed in terms of inches per tooth. The following feed values are the maximum recommended for this research.

- Aluminum = 0.003 inches per tooth.
- Brass = 0.003 inches per tooth.
- Copper = 0.002 inches per tooth.
- Cast Iron - Soft = 0.003 inches per tooth.
- Cast Iron - Medium = 0.003 inches per tooth.
- Cast Iron - Hard = 0.002 inches per tooth.

- Stainless Steel = 0.001 inches per tooth.
- Steel - Less Than 0.3 percent Carbon = 0.001 inches per tooth.
- Steel - 0.3 - 0.6 percent Carbon = 0.0005 inches per revolution.
- Steel - Greater Than 0.6 percent Carbon = 0.0003 inches per tooth.

The time required for a cut is the length of cut divided by feedrate. The time per cut is used to determine the amount of time required to manufacture a part. This is then used to calculate queue lengths, in time units, at each machine. This information is also used to update shop status information. The horsepower required for a particular machining operation is computed as a function of feed, speed, material type, depth of cut, inverse of machine efficiency, and a constant whose value depends upon the application and the material type. Before any part is machined, it is verified if the machine that is to be used possesses the required horsepower to complete the job.

4.6 Machining Operations Possible

Analyzing the description of the machines modelled in the pseudo facility under consideration, it is evident that the following are the machining operations that are possible in the facility. They are :

- Turning.
- Facing.
- Drilling.
- Boring.
- Reaming.

- Internal Threading Using Dies.
- Thread Cutting - Internal and External.
- Taper Turning.
- Tapping.
- Side Milling.
- Face Milling.
- Plain Milling.
- Slitting.
- Shell Milling.
- Pocket Milling.
- Knurling.
- Cutoff and Parting.

The list of operations possible in the facility along with the specifications and capacity of the machines considered help decide the job types, surfaces, and job sizes that can be machined.

4.7 Conclusion

It is obvious from the discussion and machine descriptions mentioned above that the facility considered has comprehensive machining capabilities. The facility assumptions define the scenario under which the system operates. The assumptions, while delineating the actual scenario which is to be used, reveal an experimental system that is robust and versatile. The system used is representative of a factory-of-the-future that would comprise several CNC machining centers with very high machining capacity and capability. The machines described can machine a variety of

material. Parameters and other relevant information associated with machining operations are known. This information is used by software to help in the process planning procedure. It is crucial to the determination of operations required to machine a part, the sequence of operations, and the choice of the actual machinery to be used.

5.0 PROLOG'S ROLE IN THE SYSTEM

5.1 Introduction

Prolog has been used extensively in this research. A very wide variety of the features and capabilities available in Prolog have been implemented. The following sections attempt to introduce the concepts and techniques used. They help understand the reasoning behind the choice and usage of Turbo Prolog in the prototype system's development. The simple construction, easy and flexible syntax, ability to backtrack, depth first search strategy, debugging and trace facilities, use of lists and string structures, interactive input methods developed, and the ability to interact with other languages are dealt with. Every capability provided in Prolog that has been discussed in the following sections was used in the development of this prototype. It is also evident from this discussion that Prolog with it's qualitative approach to reasoning and automatic derivation of alternate options and solutions is an appropriate tool to use in this prototype development.

A multi-faceted role is played by Prolog in the implementation and working of the prototype system. This software segment contains objects, their relationships, part description input and comprehension structure, forward chaining plan reasoning structure, and the process plan output mechanism. It is a storehouse of facts and rules, using the rules to help solve the problem through an inference engine that decides on the process plan, and moves from one state to another within the decision state space.

In the recent past, there has been a phenomenal drop in computer hardware costs and a tremendous increase in the cost of software generation [72]. This has influenced the development of new programming tools to help make the programming procedure easier [44]. Prolog is one such language developed after years of research. Several AI application programs and expert system shells are written in Prolog. Prolog finds all possible feasible solutions to a problem. The Prolog system organizes how required computation is carried out. It can help a programmer describe a problem in an efficient and well structured manner.

5.2 Prolog's Applications

Prolog can be used for a large variety of applications. It can:

1. Produce prototypes for virtually any application program.
2. Control and monitor industrial processes using the access provided to computer input ports.

3. Translate languages, either natural human languages or from one programming language to another.

4. Implement dynamic relational databases.

5. Construct expert systems and expert system shells.

6. Construct natural language interfaces to existing software.

7. Construct symbolic manipulation packages for equation solving, differentiation, integration, etc.

8. Be a part of theorem and artificial intelligence packages in which the reasoning capabilities are used to test different theories.

9. Identify multiple solutions to a specified goal through built in capacities to perform a depth first search and output all possible solutions.

5.3 Prolog And Other Languages

Prolog is a descriptive and declarative language that describes a problem rather than how a computer must work to solve a problem. The problem is described through software in three steps:

1. The names and structures of the objects considered in the application.

2. Names of relations known to exist between the objects.

3. Facts and rules used to describe this relationship.

The description in Prolog is used to specify the desired relation between the input provided to the system and the output generated. Apart from some initial declarations, a Prolog program consists of logical statements in the form of facts or rules. It can use the facts and the rules to make deductions through a built-in inference en-

gine. Most of the unique features present in Prolog involve procedural interactivities with Prolog's declarative kernel.

Languages such as Pascal or Basic are procedural in nature. In these languages the rules and facts need to be explicitly stated. The language has to be used to direct the compiler in searching for a solution, where to look, and when to stop. The distinction between a declarative and a procedural language is a reason that Prolog can be used to develop AI applications, especially when it is compared with other languages.

Execution of Turbo Prolog is controlled automatically. During execution, the system tries all possible sets of values that satisfy a given goal. Backtracking is used to identify all possible solutions using the rules, facts, and conditions provided. Prolog has a syntax that is comparatively simpler than most traditional languages such as FORTRAN. It is hence easier to learn and use. Besides, it is a very 'powerful' language, and typically uses ten times fewer lines than a comparable Pascal program. This is because of the built in pattern recognition facility and the simple and efficient method used to handle recursive structures.

Turbo Prolog is compiled, but allows interactive program development. A programmer can test individual modules of a program at any point. The goal can be altered at any point without adding new code. This would be similar to being able to try any procedure in FORTRAN after the program has been compiled. This brief overview of some of the capacities and unique features of Prolog in general and Turbo Prolog specifically that are used in this research make obvious the advantages of Turbo Prolog over conventional languages such as FORTRAN for CAPP. This is also

evident if other application programs written in Prolog are compared with similar efforts, for example, in FORTRAN.

5.4 Features of Prolog

Features of Prolog that were used in this research effort are :

1. Objects and relationships.
2. Rules and backtracking.
3. Backtracking control through cut and fail.
4. Recursion.
5. Functors, lists, and string structures.
6. Input and output techniques.
7. Trace and include facilities.
8. Interaction with other programming languages.

These facilities are tools used in this application to help in advanced software environment development.

5.4.1 Objects And Relationships

At its most elementary level, Prolog deals with objects and the relationships between them. An object is anything represented symbolically on a computer. The simplest method to combine an object and a relationship is to use them to define a fact. The relationship forms the predicate and the object is called the argument. An

example of a simple fact definition would be machine_1 (facing) in which machine_1 (symbol) is the predicate and facing is the object or the argument. Standard object domain types are symbols, integers, real numbers, characters, and strings.

It is possible to define compound objects that contain other objects. They are regarded and treated as a single object. The domain declaration for compound objects differs than that for simple objects. It is also possible through Turbo Prolog to construct compound objects that go down several levels. This would represent a tree structure. An example is machine_1 (facing, material (aluminum)). This compound clause could indicate that machine 1 could perform the facing operation on an aluminum part. Compound objects have been used in this software effort. They increase the software's efficiency, and reduce the length of the code.

Simple and compound objects have been used in a widespread manner in this prototype system. They define, among others, the processes that can be performed on a machine, material types that a machine can process, specifications and capacity of a machine, surface and job types that can be processed on a specific machine, and operations that can be used to generate specific surface profiles and finishes. They have been used to store relevant pieces of information that can be used in the process planning function in an elegant and efficient manner. This is evident from a perusal of the actual software code presented in Appendix A.

5.4.2 Rules And Bactracking

Rules are used to define the actual relationships between objects and to state facts. A typical rule says that something is true if some other things are true. Rules

take Prolog beyond the status of a searchable dictionary or database into a logical, thinking machine. Rules are are also used to help in the inference process to move from one state to another in the decision space. There are several process planning systems that are rule based in nature [67].

When a simple or compound goal is specified in a Prolog application program, system software tries to attain a solution to the first goal and continue to the next sub-goal only after the first goal is satisfied. If a particular sub-goal is satisfied but the next state fails, then Prolog backtracks to the previous sub-goal, uses another root of the tree structure to search for another possible solution. Prolog continues its search for solutions to the next sub-goal until all sub-goals have been satisfied.

Once one set of solutions have been identified, Prolog backtracks to the previous sub-goal and attempts to use another root of the tree structure to identify another possible solution. The backtracking facility coupled with the depth first search strategy allows for all possible solutions to be identified.

The rules and backtracking facility available in Prolog have been used extensively in this software developmental effort. Rules have been used to reason through the knowledge base and apply manufacturing procedures through an appropriate forward chaining mechanism to arrive at the appropriate process plan. The ability of Prolog to backtrack and perform a directed and controlled search procedure prompted the use of Prolog in this research. Alternate route identification, identification of appropriate alternate processes, determination of machining hierarchies and procedures, are through the use of rules and the capability to backtrack. These features have also been used, for example, in the identification processes possible on individual machines, their specifications and capacities.

5.4.3 Backtracking Control Through Cut And Fail

When a program written in Prolog is asked to satisfy a particular goal, it initiates a depth first search. If in the search process a dead-end is encountered, the program backs-up or backtracks to the previous node to find another branch of the tree structure to search. The program then searches through every branch of the tree structure that might emanate from that node in a depth first manner in order to identify possible, satisfactory solutions. This is repeated for every node of the tree structure. It is possible that this may cause the software to spend excessive amounts of time in retrieving the required information. In order to reduce the search space and avoid a combinatorial explosion of possible solutions, cut and fail commands can be used to direct the logic, prevent and curtail search or provoke and promote search, in order to use computing time and effort efficiently. Figure 22 presents a tree structure and helps in describing a depth first search process executed in Prolog with the cut and fail commands.

The cut command behaves essentially as a built-in predicate with no arguments. It is very useful in reducing the search tree to a manageable size. The depth first search technique adopted by Prolog can be controlled through the cut command. It freezes the bound variables at their current values. It prevents the search from back tracking up to a point before the use of the cut in the goal driven search process. The cut command is also used when a short, deeply interrelated set of clauses would print out multiple answers when only one is needed. This command is said to make software reflect good programming style and make software more readable. This creates slower programs, but they are easier to read.

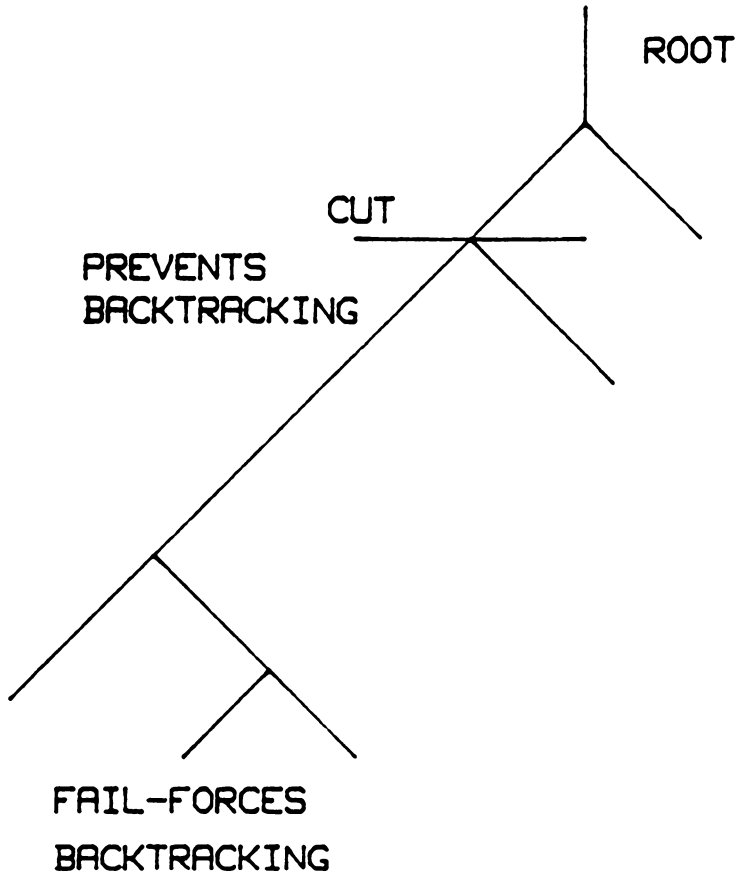


FIGURE 22 - DEPTH FIRST SEARCH STRATEGY

The fail command is, in a sense, the opposite of the cut. It forces backtracking and can never be satisfied. Everytime the fail predicate is encountered, Prolog is forced to backtrack to search for new solutions until a cut is encountered or the first sub-goal is reached. In effect, Prolog searches through all the nodes and branches of the tree structure until all possible solutions are identified or a cut curtails and prevents further search. The cut and fail commands help the software designer to effectively control the search procedure. Fail forces backtracking as surely as the cut eliminates it.

Cut and fail have been used in this research to identify multiple solutions to manufacture a part. They have been used to backtrack through the clauses and objects in a directed search to arrive at multiple, relevant solutions to a specific manufacturing goal. The presence of this facility made Prolog an ideal language to use in this research.

5.4.4 Recursion

Recursion is used to describe operations that call themselves as part of a process. Recursion occurs when relations are described with the help of the relations themselves or when compound objects are a part of other compound objects, that is, they are recursive objects. A common example of recursion is the factorial operation. It is possible for a recursive operation to get out of hand if there is no tangible end point. The factorial operation will stop when one is reached. However, it is essential that all other operations that are recursive in nature have similar limits imposed.

A stack is used to save the values and variables mentioned in the search for the recursive endpoint. If the definition requires too many levels of recursion, then the stack could overflow. Data would then be lost, and an error would result. When a recursive expression reaches its limit, it can find an endpoint value that will work back through the many levels of parentheses to identify the final result. In Prolog, a recursive predicate definition marks the temporary exit and return points along the way. When an end point value is identified, it then works up the chain to the beginning.

Recursion can be used to great advantage with lists that need to be inspected, pulled apart, and put back together. Lists are the basic data structure in Prolog, similar to the use of arrays in Pascal. Hence, the manipulation of lists through the use of Prolog recursive structures assumes importance. Recursion can also be used when the number of elements in an object is not known in advance.

5.4.5 Functors, Lists, and String Structures

The ability to use and build data structures is fundamental to any practical application program. Turbo Prolog allows for complex arguments but also has a number of built-in predicates that operate upon strings as if they were lists. These features require the user to understand the use of lists, functors, and strings in practical application programming.

A simple relationship and its objects are often written in Turbo Prolog as a relationship (object, object). This can be also expressed as a predicate (argument, argument). Representation of knowledge can be simplified through the use of complex

objects. The use of complex objects allows for more detail to be added to clauses, and simplifies the code used. In such as scenario, when an argument itself is a predicate, then it is called a functor. The structure of a functor is predicate ((argument), functor (component, component)). However, in the use of functors within complex clauses, too many levels would result in program reading difficulty and debugging. It is also possible that the components of a compound object can itself be a compound object.

5.4.6 Input And Output Techniques

The write predicate is used to output data and values. It can be called with an optional number of arguments. These arguments can be constants or bounded variables. The standard predicate 'nl' is used to in conjunction with the 'write' predicate to cause continuation from a new line. The write predicate does not provide much control over the output format especially in the case of objects like lists. It is possible to write to the screen or to a printer.

To input information into a Prolog software system, the standard predicates for reading can interpret whole line characters, integer, real, and character values from the keyboard, and through files. The capacity to read lines of characters, integers, real, and character values from the keyboard, and from a disk file are used in this software developmental effort. The interactive nature of the software designed, the need to enter part descriptive code and auxiliary information, and the need to output comprehensive process plans, and interface with other software languages has caused the need to use all possible input and output features available in Turbo Pro-

log. Information is written out to the screen as also to multiple files. Information input to the system is through the keyboard and files.

5.4.7 Trace and Include Facilities

The trace facility is provided to help with software debugging in the event of logic errors. The trace command is provided at the beginning of a program. When this directive is added, the program is examined through every single step, calling one predicate, making one match, or writing one line at a time. At the end of each step the system explains what has been done and waits for further instructions on what needs to be done. The trace directive shows the goals and subgoals that system software attempts to match up in the trace window.

The 'shorttrace' objective provides you with less information. It is also possible to use the trace facilities to observe the working of certain predicates while leaving the other predicates to run normally. The trace directive does not consider the use of the write predicate in software. There are other operators and predicates that receive special consideration by the trace directive. The trace facility proved to be of immense importance in helping develop this software.

5.4.8 Interaction With Other Programming Languages

The software developed in this research needed to interact with other programming languages. The languages it interacts with are FORTRAN and REXX. It is re-

cognized by Prolog users that although it is an excellent tool for many purposes, it is easier to use FORTRAN, for example, for certain numerical calculations. Besides, it is possible that software subroutines that perform some of the necessary computations may exist in some other language. They may need to be interfaced with Prolog software. The current languages that can be interfaced with Turbo Prolog are Pascal, C, FORTRAN, and assembler.

To inform the Turbo Prolog that an external language subroutine is used, a language specification is appended to the global predicate declaration. Turbo Prolog makes the interfaced language explicit to simplify the problems of record activation and parameter format, calling and returning conventions, segment definitions, linking, and initialization. Turbo Prolog also provides the programmer access to basic I/O system routines. The ability of Turbo Prolog to interface with other languages has been used in this software developmental effort, albeit through the use of REXX.

5.5 *Software Structure*

The software developmental structure is divided into the following sections:

1. Domain declarations and predicate definitions.
2. Facility capacity stored in clauses.
3. Part profile input and comprehension.
4. Process and procedure inference techniques.
5. Forward chaining process planning mechanism.
6. Process plan output technique.

Domain declarations, predicate definitions, representation of the facility's capability as clauses, and the part profile input and comprehension modules form the system input and description section of Macro-CAPP. The process inference techniques and the forward chaining process planning mechanism describe how the system functions in arriving at an acceptable process plan. The plan output structure describes in detail the information output mechanism used. The software structure used is described by a flowchart in Figure 23. Appendix A contains a detailed listing of the software used.

5.5.1 Domain Declarations And Predicate Definitions

The first module of Prolog software consists of domain declarations. They comprise module one as indicated in Figure 23. They describe to the system compiler if a variable is an integer, real, character, string, file, or a symbol. This software structure specifies the domains to which objects in a relation belong. For example, a part's surface finish, tolerance, etc. would be given an integer domain. However, the material type would be identified as a symbol. A sub-module of software illustrating some domain declarations used is shown in Figure 24. This software module requires the perfect understanding of the role a specific variable, since this is the factor that decides on the actual domain type that is assigned.

Predicate definitions are used to specify the objects used in a relationship, and the actual domains of these objects. They comprise module two of system software as indicated by Figure 23. Predicate definitions are used to demarcate the number

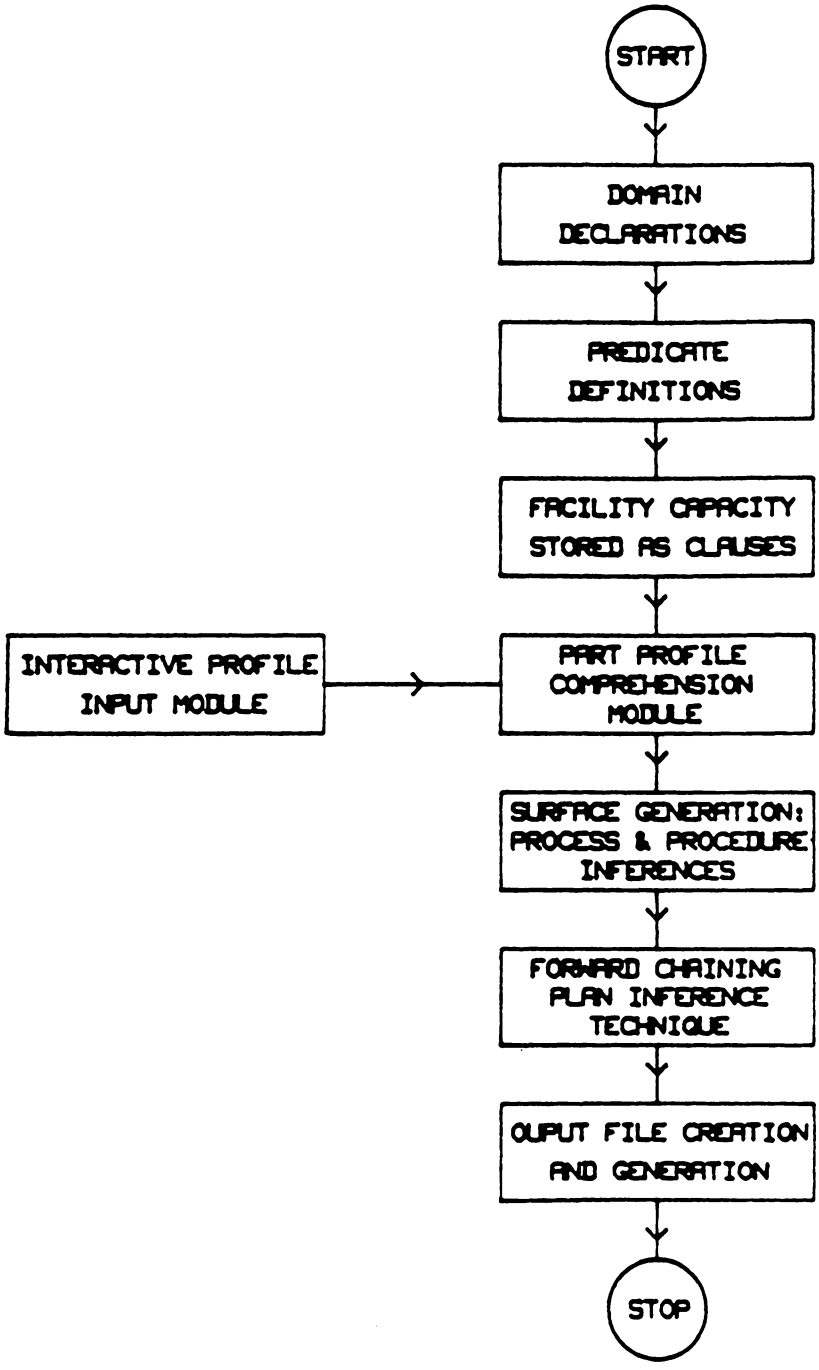


FIGURE 23 - PROLOG CAPP SOFTWARE STRUCTURE

```
DOMAINS

        Comp_Class = Integer
        Ext_Shape = Integer
        Int_Shape = Integer
        Plane_Surface_Mach = Integer
        Aux_Holes = Integer
        Dimension = Integer
        Tolerance = Integer
        Surface_Finish = Integer
        Mach_Identifier = Integer
        N = Integer
        file = myfile
        mach_id = Symbol
        operation = Symbol
        Init_Form = Symbol
        Material_Type = Symbol
        maxheight = Symbol
        maxdia = Symbol
        mindia = Symbol
        material = Symbol
        maxlength = Symbol
        minspeed = Symbol
        maxspeed = Symbol
        speedincr = Symbol
        type = Symbol
        name = Symbol
        width = Symbol
        jtype = Symbol
        jobdes = Symbol
        number = Real
        maxturrets = Symbol
        maxhp = Symbol
        maxtools = Symbol
        maxatc = Symbol
```

Figure 24. Some Domain Declarations Used

of objects, if any, in a variable. Predicates can also consist of a name only. Multiple predicate declarations are also possible.

Some of the predicates used in this software developmental effort are shown in Figure 25. They are used to define the variables used in each software module. Simple and compound clauses are used in software. The predicate definition could be, for example, delay (integer) where delay has a single object that is an integer. It could also be just a name, such as read_comp, a variable that contains no objects related to it. Every variable used in software needs a predicate definition. This would include the part description and profile input module, the clauses used to describe shop structure and capacity, variable names used in the process plan inference mechanism, the file structure used in recording part shop characteristics and capacity, and the file and information output mechanism.

5.5.2 Facility Capability Stored In Clauses

The facility capability module, module three in Figure 23, stores the capacity of each machine with respect to a variety of factors. The actual capacity and capability of each machine has been described in detail in Chapter Four. Information stored in software includes the actual job types that can be handled on a specific machine. The job types are either rotational or prismatic. There are three classes each of rotational and prismatic jobs. Prismatic jobs can be machined on the machining centers only. The rotational jobs are machined on turning centers. Milling operations needed for both job types are performed on machining centers.

```
Predicates

        delay (integer)
        read_comp
        readaux
        run
        mach_operations
        go
        start
        surface_finish (integer)
        mach_cap (mach_id,jtype)
        mach_use(mach_id, name)
        machtype (mach_id, name)
        machine (mach_id, operation)
        mach (maxdia, number)
        mach (maxlength, number)
        mach (minspeed, number)
        mach (minspeed, number)
        mach (maxhp, number)
        mach (maxtools, number)
        mach (maxturrets, number)
        mach (speedincr, number)
        mach_material (material, type)
        jtype (jobdes)
        thread (symbol)
        outer (symbol)
```

Figure 25. Some Predicates Used

This software module helps the user and the software understand the actual system capacity, job types, materials, surface finishes, surface profiles, etc. that can be machined in this facility. They are designed to help the process planning inference procedure work in an efficient manner. Each sub-module that is a part of the facility capability module contains simple clauses that are used to help store information since they allow the user to very easily add, delete, or amend information stored. They also make the code very easy to read and understand.

The first software sub-module stores the description of the outer surface of jobs that can be machined in the pseudo facility considered. It is necessary to identify the surface types that can be machined on each machine modelled in the facility. This helps the rule based reasoning mechanism and the depth first search procedure to identify the machines on which a specific job can be processed. The outer profile of jobs machinable by the facility includes threads, smooth surfaces, grooves, slots, outer surface stepped to one side, outer surface stepped to both sides, outer surface with plane surface machining or without plane surface machining, or with plane surface chamfers. These facts are stored in software as simple clauses.

The construction of this software sub-module, modelled as simple clauses, is as follows:

```
outer (smooth).
outer (groove).
outer (stepped_one_side).
```

Translated into words, this indicates that surfaces such as smooth external surfaces, grooves, and surfaces stepped to one side can be machined in this facility. This example indicates the use of simple, concise clauses on a large scale in information storage in this software module.

The next software sub-module stores in simple clauses the internal surfaces that can be machined in the pseudo facility. The internal surfaces include smooth holes, holes stepped on one side or on both sides, internal slots, internal multiple parallel bores, or non-parallel bores. This is used for both rotational and prismatic parts. The clauses used to store the internal surface machining capacity of the facility are presented in detail in Appendix A. The construction of the software sub-module that stores the internal surface machining capacity of the machining centers in the facility is as follows:

```
internal (none).
internal (smooth).
internal (stepped_one_side).
```

Translated into text, this indicates that jobs with no internal surfaces, smooth internal surfaces, or those stepped to one side can be machined in this facility. These clauses indicate the construction of the software used in the storage of internal surface machining information.

The pseudo facility considered can also machine external plane surfaces for both prismatic and rotational parts. The purpose of this sub-module is to store the actual external plane surface types that can be machined. This is done in a manner that is easy to read and modify. The external plane surface can be plain, have a single or a multiple curve, have an external keyway, an external groove, or a slot. These facts are stored in software simple clauses. An example of the code would be as follows:

```
plane_surface (external_single_curve)
plane_surface (groove).
plane_surface (none).
```

This code states that plane external surfaces with single external curves, grooves, or jobs requiring no external plane surface machining could be processed in the facility modelled.

It is possible that holes may need to be drilled into the part. The holes could be axial or radial in orientation, related by graduations around a circle in the radial or axial direction, not related by graduations around a circle in the radial or axial direction, or could be related by graduations around a circle but multidirectional in orientation. The description of each scenario is stored in the form of simple clauses in software. An example of the clauses would be as follows:

```
holes (none).
holes (axial_unrelated).
holes (radial_unrelated).
```

Translated into text, this code indicates that parts with no holes in them, with axial holes or radial holes not related to each other in position, could be machined in this facility. The purpose of each sub-module of this software module is to describe in a comprehensive manner the actual types of holes that can be considered by this CAPP system.

The software also contains, stored as simple clauses, the material types that can be machined in the facility. The materials that can be machined are copper, brass, aluminum, three types of iron, three types of steel, and stainless steel. Details of the material that can be machined are presented in Chapter Four. The clauses that describe material type are similar in construction to the other information storage sub-modules. It has a simple, easy to understand, concise construction. They are developed in a manner that would help the reasoning mechanism. An example of these clauses would be as follows:

```
material (brass).
material (copper).
material (aluminum).
```

Translated into text, this means that this facility could machine brass, copper, or aluminum parts.

The facility under consideration can machine surfaces to a finish of 32, 64, and 128 micro inches. This facility capacity information is presented in clauses to enable the reasoning mechanism to identify if a particular surface finish can be attained, and use that information in the process planning process. An example of the software structure used would be as follows:

```
surface_finish (32).
surface_finish (64).
```

Translated into text, this indicates that the surface finish values that could be considered by this facility are 32 or 64 microns.

The software clauses described above contain information stored as simple clauses that describe in sufficient detail the capability of the entire facility. It does not distinguish between individual machining centers. However, this information is also needed to help decide the actual machines on which a particular job could be processed. Information on individual machine capabilities, specifications, material types, job types, surface types, etc. need to be stored. The following software sub-module constitutes the second part of the facility capacity module, module three in Figure 23.

Each machine considered as part of the facility is described as clauses in software. The machines used are three similar but not identical turning centers, and two machining centers, one being vertical and one horizontal. Machines one, two, and five are designated as turning centers, and machines three and four are designated as machining centers. The horizontal spindle machining center is designated as machine three, and the vertical spindle center is designated as machine four. The characteristics, specifications, and details of each machining center are presented in Chapter Four.

Each machine is given a specific identifier in software. The machine type is also stored. An example would be as follows:

```
mach_type (mach1, turning_center).
mach_type (mach2, turning_center).
mach_type (mach3, horiz_spindle_mach_center).
mach_type (mach4, vert_spindle_mach_center).
mach_type (mach5, turning_center).
```

This set of clauses identifies 'mach1', 'mach2', and 'mach5' as turning centers, 'mach3' as a horizontal spindle machining center, and 'mach4' as a vertical spindle machining center. This software sub-module is used by the reasoning mechanism to identify the machines, relate a machine type to a machine, and use that information to decide upon actual job routes, machining parameters, machining times, and make other decisions in the process planning procedure.

The next step is to describe the actual machining processes that can be performed on each machine in the facility. This is achieved through the use of simple clauses that state the actual processes that can be performed on each machine. They are used by the CAPP system to determine and decide which machines have the capacity and capability to perform a required operation. For example, it can be used to determine which machines can perform turning of a rotational part or which can perform a partoff operation. This information is used to describe in detail each machining center and to determine if a part can be processed on a particular machine, compute machining speeds and times, and determine the operations executed on each machine. The working of individual inference and deduction procedures is described in detail in subsequent sections of this chapter.

The inference procedure designed and implemented also helps determine the operations that can be performed on each machine within the facility. This inference

module can be used, for example, to identify the operations possible on machine 4, the vertical spindle machining center. A sample output is presented in Figure 26.

An example of the code that is used to store information on the actual machining processes that can be performed on a specific machine is presented below.

```
machine (mach1, knurling).
machine (mach1, turning).
machine (mach3, side_mill).
machine (mach5, drilling).
```

When translated into text, this would mean that turning and knurling could be performed on machine one, side milling on machine three, and drilling on machine five.

The next sub-module contains the machining capabilities of each machining center modelled in the FMS. The factors stored include the maximum diameter that can be machined for turning centers, maximum job width and height for machining centers, maximum job length, minimum and maximum speed that can be attained, number of turrets, number of cutting tools, speed increments, and the maximum number of tools that can be used on a machining center. This information is used to describe in detail each CNC machine and to determine if a part can be processed on a particular machine.

The information is stored as simple clauses that are easy to read, understand, amend, and add to or delete information. An example of the code used in this sub-module is as follows:

```
mach (maxdia1, 8.0).
mach (maxlength2, 20.0).
mach (minspeed2, 32.0).
mach (speedincr3, 1.0).
mach (maxhp4, 20.0).
```

```
The first machine is turning_center
mach1 turning
mach1 facing
mach1 drilling
mach1 boring
mach1 reaming
mach1 knurling
mach1 external_threading
mach1 internal_threading
mach1 taperturning
mach1 partoff
```

Figure 26. Operations Possible On A Vertical Machining Center

This indicates that the maximum diameter that can be machined on machine one is eight inches, the maximum job length that can be processed on machine two is twenty inches, the minimum speed that machine two can work at is thirty-two r.p.m., etc. The specifications of each machine are identified and related to specific machine identifiers in this software sub-module.

Also stored in software is a set of clauses that define the materials that can be processed on each machine. This module is used to deduce the material types that can be machined in the facility in general and specifically on individual machines. The forward chaining inference procedure uses this module in deciding upon possible job routes, since the material type is a factor in deciding the machines on which a job can be processed.

A portion of the software used in storing this information in the prototype system is presented below.

```
mach_material (mach1, copper).
mach_material (mach1, brass).
mach_material (mach1, aluminum).
mach_material (mach1, iron_soft).
```

Translated into text, this software segment would indicate that machine one, a turning center, could machine copper, brass, aluminum, or soft cast iron.

Also stored in the form of clauses in software is information on the machines that can be used to perform a particular process. This can be, for example, the machines that can perform turning. This would help identify to the user and the CAPP inference mechanism the machines that can be used to perform a specific operation. An example of the software coding used in this sub-module would be:

```
turning (mach1).
drilling (mach1).
```

```
reaming (mach1).
face_milling (mach3).
```

This indicates that turning, drilling, and reaming could be processed on machine one, and face milling on machine three. A complete listing of the software is presented in Appendix A. Together, all the clauses and software sub-modules mentioned above comprehensively define and describe the facility's capacity and capability to the inference technique built in to the CAPP system, and help it in the process planning function.

5.5.3 Part Profile Input And Comprehension

The part profile input module, module four in Figure 23, interacts with the user of the prototype CAPP system, and initiates information input and part description. Completely and comprehensively interactive, this module presents the user with appropriate questions and choices. It requests information to help the user describe to software the actual construction of the part under consideration. The construction of the part description code and the significance of each digit used is explained in detail in Chapter Three. The input module also presents the user of Macro-CAPP information that may be required to use the system effectively. An example of the initial input screen used is presented in Figure 27. This screen introduces the user to Macro-CAPP, prior to actual use.

The input module requests information on the component class. The user needs to analyze the part under consideration, and enter the component class. A number ranging from one through six could be entered. This would help the reasoning strategy of the CAPP system software decide if the part were prismatic or rotational. It

would also give the part an approximate value of the L/D ratio if the component were rotational or an approximate value of A,B,C for a prismatic part. The notations L, D, A, B, C have been described in detail in Chapter Three.

The next piece of information requested by the input module considers external part shape. It is numerical in nature. The number used varies from one through six. It is used by the subsequent part profile comprehension module. It helps differentiate between the types of external surfaces considered. It must be remembered that this is a hybrid code that is used in part description. The second digit works in conjunction with the first in deciding part shape and structure.

The third digit is a numerical input that describes the component's internal shape. The description of a component's internal shape is identical for prismatic and rotational parts. This digit is independent of the first two. It is used by subsequent reasoning strategy to decide upon the alternatives that could be used to machine the internal surfaces required. This is followed by the fourth digit that describes in detail the external plane surface machining that is required for a part. This helps decide if the part requires any plane surface machining, and which CNC machine could be used.

The fifth interactive input that forms part of the main part description code is numerical in nature and describes the auxiliary holes that the part may contain. It could be radial or axial holes. This information is used by the inference mechanism to decide upon the actual methods that could be used in this facility to generate the holes required.

```
Welcome To MACRO CAPP
An Interactive CAPP System
That Interfaces CAPP With a CIM System
Designed For Implementation
In A Versatile FMS Cell

This Integrated System Uses GT Coding
For Interactive Part Description
Please Consult Accurate Part Drawing Prior
To Actual Part Description Input
Please Remember That The Part Input Supplied
Indicates To The Software The Information
Required On Part Profile
And Some Interactive Input

Have A Nice Session


The shop contains the following
mach1 - turning_center
mach2 - turning_center
mach3 - horiz_spindle_mach_center
mach4 - vert_spindle_mach_center
mach5 - turning_center
Are you ready to start
Start/Wait, S/W
Use Capital's Only
```

Figure 27. The Initial Input Screen

At each stage in the information input process, a delay function is used to ensure that the screen does not scroll very rapidly. This gives the user adequate response time. If the delay function were not used some of the input screens would scroll by at a very fast rate, inhibiting the user from comprehending the descriptions, instructions, and relevant information presented. The input screen used describes to the user the construction of the facility. It gives the user an idea of the machines, their capacities, specifications, and the overall shop structure. After the part description is completed, software waits for the user to return a character or variable initiate the CAPP information input process.

There are auxiliary inputs used to describe the part under consideration comprehensively. The material type is a character input that helps decide the speeds, feeds, depths of cut in the process planning procedure. The material types under consideration are described in detail in Chapter Four. The next two auxiliary inputs, the surface finish desired and the tolerance required in the final part, are numeric in nature. They help decide upon the actual finish machining processes available that could be used to arrive at the final, required surface finish and maintain the desired tolerance. These inputs help in deciding upon the actual sequence of operations required in part manufacture. Information on surfaces finishes and tolerances considered by the system are presented in Chapter Four. The dimensions of the part are interactively input. These are dependent upon the actual part under consideration. They are entirely numeric in nature. They are used in computing the actual machining times, and later in the computation of job flowtimes. The use of part dimensions in the computation of machining time and job flowtimes is presented in detail in Chapter Six.

At each stage a safety mechanism is used to ensure that if an erroneous input is used software recognizes an obvious error, and prompts the user to re-enter the same. If, for example, a character input is used where a numerical input is sought, then software would re-prompt the user to use the appropriate correct input. Besides, after all the inputs have been entered, they are displayed on the screen to help the user validate their accuracy. Software waits until the appropriate character variable is entered to verify the accuracy of the inputed information. The process planning function could be continued if the inputed information is correct or re-started if any change is required.

The interactive codes that are input into the software describe the following information:

1. Component class.
2. External shape.
3. Internal shape.
4. External plane surface machining.
5. Auxiliary holes.
6. Material.
7. Surface finish.
8. Tolerances.
9. Initial form processed.
10. Dimensions.

Figure 28 exhibits the construction of an interactive input screen. It is evident from the discussion presented above that none of the inputs used are superfluous in nature. Only those inputs that are absolutely required are used to avoid the storage of

excess, unused information. Each input entered is invaluable to the process planning task undertaken by this prototype CAPP system.

5.5.4 Process and Procedure Inference Techniques

The first module in the process and procedure inference technique consists of software routines that identify the processes, procedures, materials, etc that can be processed on each machine. The first block of software in the inference procedure identifies the processes that can be executed on each machine in the pseudo facility modelled. This sub-module considers each machine and its identifier in software. It then reasons through the facility capacity clauses that contain the processes that can be executed on that machine. This software sub-module is as follows:

```
openwrite (myfile, "machcap1.one),
writedevice (screen),
machtype (mach1,X1),
write ("The first machine is", X1), nl,
writedevice (myfile),
write ("The first machine is", X1), nl,
!, machine (mach1, A1),
writedevice (myfile),
write (mach1, A1), nl,
writedevice (screen),
write ("The processes possible of machine 1"), nl,
write (mach1, A1), nl,
fail.
```

The software presented above identifies the machine, its capabilities and writes the same on to the screen and to a output file. Appendix A presents the actual location and listing of the code described above. Figure 29 contains an output derived through the use of the machine capacity identification module.

```
The Input Consists Of Five Digit Codes
And Several Interactive Inputs

Digit One Inputs Component Class
Component Class ?
Component Class  3

Digit Two Explains the External Shape of
The Product To The System Software

External Shape ?
External Shape 5

Digit Three Explains The Internal Configuration of
The part to System Software

Internal Shape ?
Internal Shape  4

Digit Four Explains The Machining
Required On The Outer Surface

Outer Surface Plane Machining ?
Outer Plane Surface Machining 4

The Fifth Digit Describes The Position
And Number of Auxilliary Holes

Auxilliary Holes Required ?
Auxilliary Holes Required 3
```

Figure 28. The Interactive Input Screen

```
The fourth machine is vert_spindle_mach_center
mach4 drilling
mach4 tapping
mach4 reaming
mach4 boring
mach4 internal_threading
mach4 side_mill
mach4 face_mill
mach4 plain_mill
mach4 dovetail_mill
mach4 slitting
mach4 end_mill
mach4 shell_mill
mach4 pocket_mill
```

Figure 29. Output Derived Using The Machine Capacity Identification Module

The second sub-module of the machine capacity identification module contains software to identify the material that can be processed on a particular machine. This software module reasons through the facility capacity and capability to arrive at the material that each machine can process. The output is written into a file and on to the screen. The code used identifies the machine, reasons through the machine capability description software, and identifies the part material that a specific machine can process. The software used in this sub-module is as follows:

```
openwrite (myfile,'material.one'),
writedevice (myfile),
write ("The material that can be processed are'), nl,
writedevice (screen),
write ("The material that can be processed are'), nl,
!, mach_material (mach3, X31),
write (X31), nl,
writedevice (myfile),
write (X31), nl,
fail.
```

It is evident that the facility capability module is used by this routine to derive the materials that can be processed on a specific machine. The use of the cut and fail predicates help in promoting and controlling search through the system. Figure 30 contains an output derived from this software sub-module.

The third sub-module of the machine capacity identification module deduces the job types that can be processed on each machine in the pseudo facility modelled. Six different job types are considered. They can be broadly classified as rotational or prismatic jobs. Under each of these heads, there are three sub-types considered. This is obvious from the classifications used in the first digit in the part description code. The output derived is written into a file and on to the screen. This software sub-module, presented in detail in Appendix A, considers a particular machine, considers the job types that can be processed on a particular machine as defined in the facility capability module, and outputs the same into a file, and on the screen.

```
The materials that can be
 processed on Machine 1

copper
brass
aluminum
steel_lt3carbon
steel_lt6carbon
steel_lt1carbon
iron_soft
iron_medium
iron_hard
stainless_steel
```

Figure 30. Material Types Processed On A Machine - An Output

Another software module is used to determine the actual surfaces that can be machined by each machine. When a component is to be planned for, then the component class, outer surface and other machining requirements have to be considered in deciding on its processing sequence. If, for example, the outer surface of a component is smooth and the cross section of the part is circular, then the part can be machined on the any of the turning centers. This is subject to the dimensional capacities of the turning centers. To decide the machines that can be used, the surfaces that need to be machined are considered. The operations that can produce the surfaces are considered. Eventually, the machines that can process them are identified. This forms an important part of the reasoning mechanism. An output derived from the use of this software module is presented in Figure 31.

The machining surfaces considered correspond to the surfaces considered in this prototype. They include external surface machining, internal machining, external plane surface machining, and the generation of holes. The surface profiles considered are described in detail in Chapter Three. The outputs generated are written into a file and on to the screen.

The next modular segment of software deduces the machines that can perform each machining process. For example, it may be necessary to deduce the machining centers that can turn a rotational part. It is then possible to identify the machines through which this job can be routed. The software routine in question considers the surface that requires to be machined, the processes that can be used to generate that surface considering the part description and dimensions, and identifies the machines that could be used. The software rules search through the facility capacity and machining information provided to arrive at the machines that can be used to perform a

```
The following surfaces can be machined
on a turning center mach1.

job_type rotational

outer_smooth
outer_thread
outer_groove
outer_slot
outer_stepped_one_side
outer_stepped_both_sides
outer_smooth_chamfers
internal_none
internal_smooth
internal_stepped_one_side
internal_stepped_both_sides
internal_slot
internal_thread
internal_groove
internal_two_principal_bores
internal_multiple_parallel_bores
internal_non_parallel_bores
holes_none
holes_radial_related
holes_radial_unrelated
holes_axial_related
holes_axial_unrelated
holes_multidirection
holes_related_multidirection
```

Figure 31. Surfaces That Can Be Processed On A Machine - An Output

required operation, for example, drilling. The software routine is presented in detail in Appendix A. It's structure is as below:

```
openwrite ("drilling.one"),
writedevice (myfile),
write ("Drilling can be accomplished on the following machines"),
nl, writedevice (screen),
write ("Drilling can be accomplished on the following machines"),
drilling (Z10),
write (Z10), nl,
writedevice (myfile),
write (Z10), nl,
delay (10000), fail.
```

As in previous software modules, the output from this software module is written to a file and to the screen. A detailed description of the output file handling procedures used in this prototype system are presented later in this chapter. Figure 32 illustrates outputs derived through the use of this software module.

It is evident that the purpose of this software module is to identify the processes and procedures that can be performed on each machine modelled in the facility. Each piece of information derived through the use of this software module is used by the forward chaining reasoning process used to arrive at the actual process plan. The machines available, the operations that can be performed on them, the operation required to generate a specific part profile, the specifications and capabilities of each machine, etc. are used. The sections presented above give the user an idea of the working of the software mechanism in this module of the prototype system. By writing the relevant outputs to the screen and to files, the output derived can be used by other software modules if required, and can be viewed by the user immediately. The user can understand the processes that are possible on each machine, the machines modelled, the surfaces that can be attained on each machine, and the materials that can be processed on each machine. It would give the user a comprehensive view of the facility's structure and capacity.

```
 Drilling can be accomplished
 On the following machines
mach1
mach2
mach3
mach4
mach5


 Turning can be accomplished
 On the following machines
mach1
mach2
mach5
```

Figure 32. Processes And Machines They Can Be Performed Upon

5.5.5 Forward Chaining Process Planning Mechanism

The forward chaining process planning technique deduces the processes required to transform a raw material to the finished part. The term 'forward chaining' denotes that the planning process originates from the raw material and develops to the finished form. The code input to the system describes the raw material form and the surface profile of the finished part. The planning procedure works from the first machining operation to the final cutting process. It uses the GT inputs to discern the part profile to be machined. This intelligent inference engine uses available information to arrive at the final process plan. The following sections describe the forward chaining technique used. Figure 33 describes the manufacturing process plan deduction technique used.

The first stage involves understanding if the part is rotational or prismatic in construction. The GT code is read in, the numeric value understood, and the type of part under consideration is identified. If the part is rotational, then the Length/Diameter (L/D) ratio of the part is understood. The L/D ratio, as indicated before, is dependent upon the GT code. Software checks if the L/D ratio is greater than 3.0, less that 0.5, or it lies between the two values. This information helps understand the construction of the rotational part with respect to its length.

If the part is prismatic then the first digit of the GT code helps understand if the part is flat, cubical, or long. This is represented by the input code used in the first digit. These are the three broad classifications used in the description of prismatic parts. This helps software achieve a broad understanding of the dimensional details of the part under consideration. It helps the system decide upon the machines on

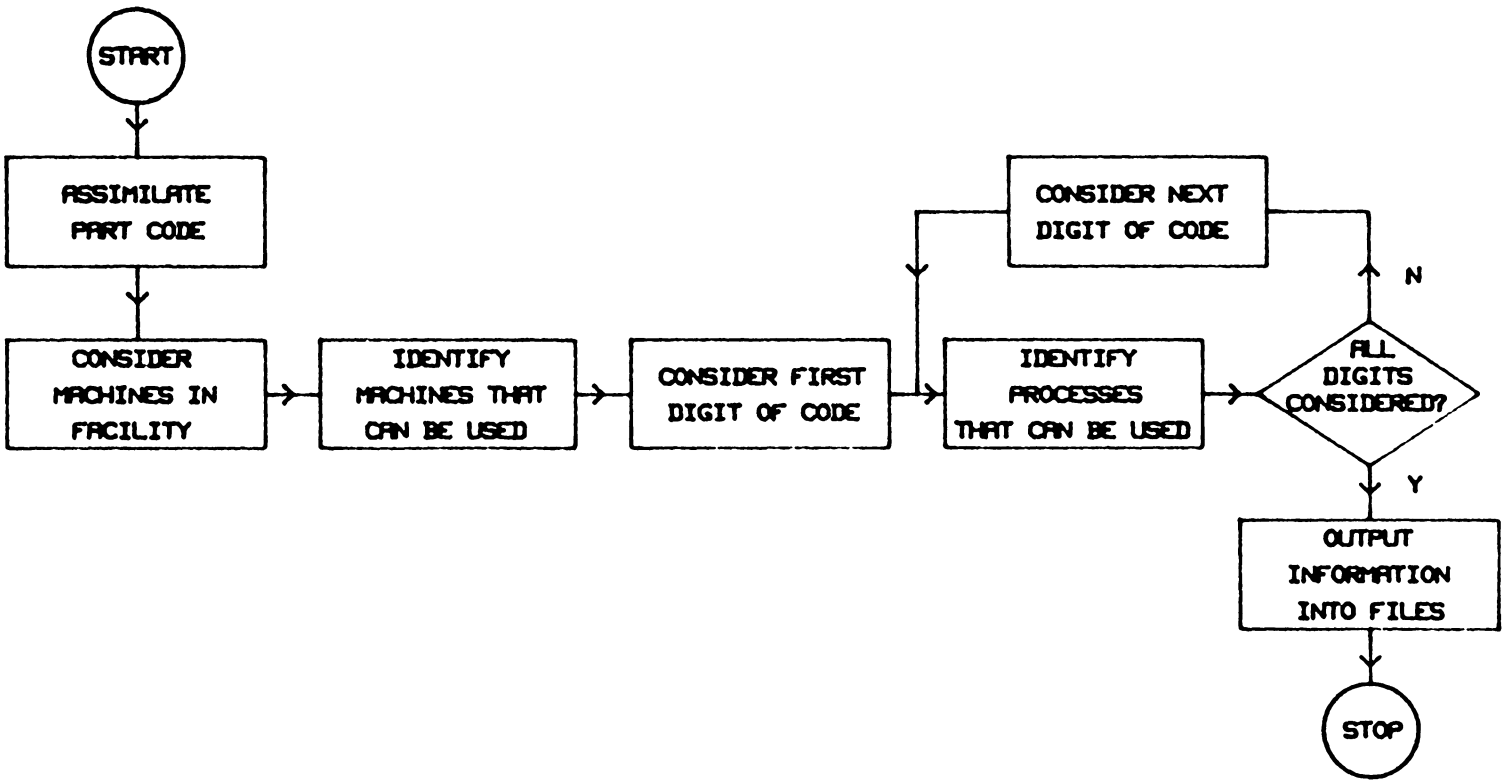


FIGURE 33 - PROCESS PLAN DEDUCTION TECHNIQUE - AN OUTLINE

which these parts can be machined. The operations that it may require will depend on milling machine capacity and job requirements. If the part is rotational, all rotational surface machining is to be done on turning centers. Subsequent operations may be done on turning or milling centers depending upon shop capacity and job requirements.

The second stage involves understanding the process required to transform the part. The second digit in the input code is used. The outer surface for a rotational part could be smooth, threaded, with a groove, an outer slot, stepped to one side, stepped to both sides, with or without chamfers. For a prismatic part, it could be smooth, require external plane surface machining, with or without chamfers. The software at this juncture checks to see what processes can be used to machine the required outer surface profile. Since this is a rule based system, there are stored in software, comprehensive rules that reason through the facility's capacity to identify the operations that could be used. Stored as simple clauses, as explained earlier, is the information that describes the operations that can be executed on each machine. The inference engine understands the part surface profile, identifies the operations that could be used, and then the machines on which the part could be loaded. The use of the cut and the fail predicate to provoke or prevent search along with the built in search strategy provided by Turbo Prolog helps identify all the alternate processes and procedures possible in the pseudo facility modelled to manufacture the part.

After the external surface machining requirements have been identified, the next feature considered is the internal machining required. The part could be prismatic or rotational. The internal features considered for both rotational and prismatic parts are: smooth through holes, holes stepped to both sides, holes stepped to one side,

internal slots, internal grooves, multiple internal principal holes that are parallel, and multiple principal bores that are parallel. The manufacturing processes used to generate these internal features are drilling, boring, reaming, gang drilling, multiple spindle drilling, or a combination of these. These processes are performed on machining centers only for prismatic parts, but on machining and turning centers for rotational parts. Prolog's inherent characteristics are used in this to identify all manufacturing processes, methods, and machines that can be used in to generate the required surfaces.

The third digit of the GT code is used to identify the actual internal surface machining required. This information is read in by the inference engine which then identifies the processes and machines that can be used. The machines that could be used and the processes suggested are checked considering part dimensions and specifications. Stored in software are the internal surface profiles considered, processes that can be used to generate a particular profile, and the machines that can be used for these processes. The depth first search strategy of Prolog coupled with the use of cut and fail are used to identify alternatives that can be used in the required internal surface machining.

The fourth digit in the code signifies the external surface construction of the prismatic or the rotational part. This software module reads in information regarding the external plane surface machining required, and uses the same information to arrive at the required manufacturing processes. As explained earlier in this chapter as also in Chapter Three, it is the fourth digit of the GT code that provides the prototype system's software with information on external plane surface profile. The processes used to generate the plane surfaces are mainly milling procedures. The external

surface types considered are: single and multiple plane surfaces, keyways, slots, and grooves. A very wide variety of milling procedures are considered. The inference engine uses rules in software combined with the appropriate use of the cut and fail features within Prolog. It helps in promoting or curtailing search through the information stored in software to arrive at the alternatives that can be used.

The material type is assimilated by software and used in the determination of the appropriate speeds, feeds, and depths of cut. This information is used by Prolog to deduce the appropriate machining parameters. The suggested machining parameters are presented in Chapter Four. This coupled with information about part dimensions are used to compute machining times. This is however done in the software module that keeps track of dynamic system status over time. The surface finish of the part is read in. The final machining process suggested is dependent upon the surface finish required. Information is stored regarding the surface finish that a part surface requires. The knowledge base stores information on the surface finish that a particular process generates. The surface finish considered varies from 16 to 128 micro-inches. The choice of the finishing process used takes into consideration the rotational or prismatic construction of the part, and the actual surface finish desired. The part's material type and the surface finish desired are both read into CAPP software from the auxiliary part description input.

The process plan uses a forward chaining technique to arrive at the process plan. The GT code used serves as a hybrid code in assisting the forward chaining process. A program listing is presented in Appendix A.

5.5.6 Process Plan Output Technique

The process plan is output into files stored on disk. While the actual process plan is output into a single file, individual pieces of information deduced through the use of the inference techniques described above are written into separate files. The following are the files output by the system:

- Machine capacity output.
- Machines in system under consideration.
- Job type.
- Material processed.
- Machine specifications.
- Surface types that can be machined.
- Processes on individual machines.
- Machines that can perform specific processes.
- Process plan with alternate routes and processes.

The file containing a process plan is stored in a file with an identifier that is dependent upon the GT part code. The file identifier is used in process plan retrieval if the same part is encountered again. The file identifier is dependent upon the GT code used in part profile description. The GT code of a part is checked with the file identifier of existing part plans. Each digit of the new part is compared with the name of existing plan files, and if there is a match, the plan is retrieved and not generated. A wide variety of file handling procedures provided by Turbo Prolog are used in this application. The files output by the system can be used to educate the user on the facility's capabilities. They are used as input to other software modules that may mesh with the CAPP system. Information contained in the files output by this prototype system includes:

1. Machine specifications.
2. Manufacturing processes possible on individual machines.
3. Facility construction.
4. Machines that can perform a given manufacturing process.
5. Materials processed in the facility.

Figure 34 presents a flowchart which details the output files generation schema. Figure 35 presents a sample output process plan.

The output files that contain information on the facility construction and capacity state the types of machines that are used and their dimensional and operating specifications. This information is presented in detail in Chapter Four. It helps the user know the possibilities that exist, and helps in verifying the accuracy of the system. The processes possible on each machine, materials that can be processed, and the surface profiles that can be machined on specific machines are output into files after the inference mechanism deduces the same through a search strategy. This information is used in the planning process, and it helps the user in validating the system's outputed plan.

The software structure identifies the machines that can be used in processing the part. These are output to files so that they can be read in and used by subsequent software modules. The processes and procedures form part of the detailed process plan that is output. This file helps in deriving hard copies of the system's output and proves to be the information link between the process planning software and the dynamic shop status module. This is the outputed file that contains the suggested process plan with the alternate routes and options built into it.

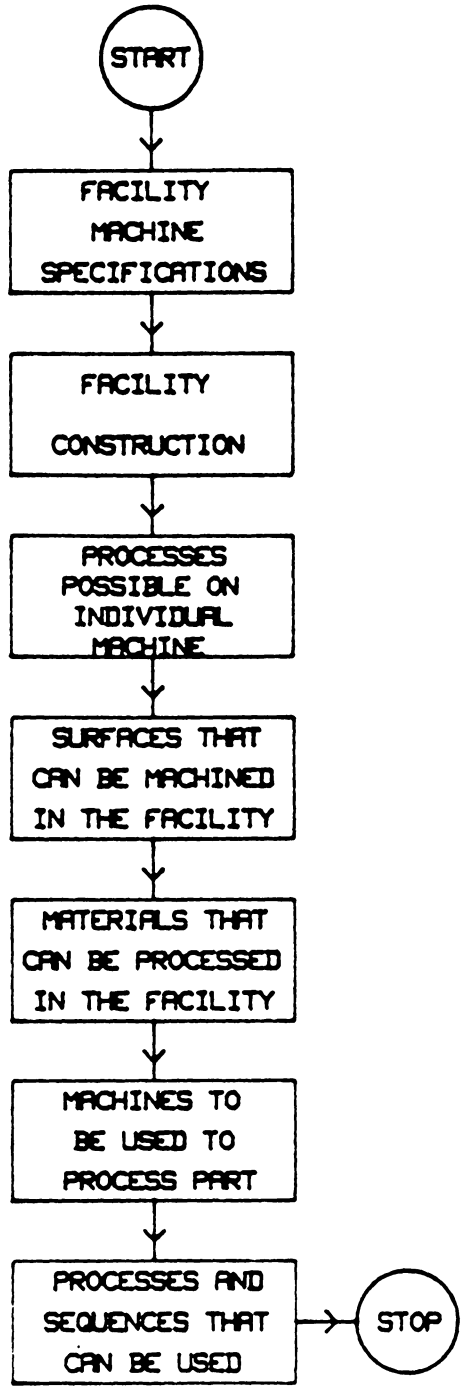


FIGURE 34 - OUTPUT FILES GENERATION SCHEMA

```
ROTATIONAL_SHORT
OUTER_SMOOTH
INTERNAL_SMOOTH
OUTER_PLANE_NONE
HOLES_NONE

THE MATERIAL OF THE PART IS ALUMINUM
QUANTITY 10000

SPEED 550      FEED 0.045

USE MACHINE ONE - TURNING CENTER
QUEUE TIME 4512.34

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  6.00     DIAMETER  2.00
SPEED = 1100 TIME = 1212.12

SMOOTH INTERNAL HOLES - DRILL AND REAM
LENGTH OF HOLE =  6.00  DIAMETER  0.05
SPEED = 4000 TIME =  333.33

NO PLANE SURFACE MACHINING REQUIRED

NO HOLES IN PART - NO DRILLING REQUIRED

USE MACHINE TWO - TURNING CENTER
QUEUE TIME 3124.34

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  6.00     DIAMETER  2.00
SPEED = 1100 TIME = 1212.12

SMOOTH INTERNAL HOLES - DRILL AND REAM
LENGTH OF HOLE =  6.00  DIAMETER  0.05
SPEED = 4000 TIME =  333.33

NO PLANE SURFACE MACHINING REQUIRED

NO HOLES IN PART - NO DRILLING REQUIRED

USE MACHINE FIVE - TURNING CENTER
QUEUE TIME 6723.67

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  6.00     DIAMETER  2.00
SPEED = 1100 TIME = 1212.12

SMOOTH INTERNAL HOLES - DRILL AND REAM
LENGTH OF HOLE =  6.00  DIAMETER  0.05
SPEED = 4000 TIME =  333.33

NO PLANE SURFACE MACHINING REQUIRED

NO HOLES IN PART - NO DRILLING REQUIRED

USE MACHINE TWO - TURNING CENTER
```

Figure 35. Sample Process Plan

5.5.7 Conclusion:

This chapter details the properties of Prolog that are used in this prototype CAPP system. The software modules used in the part comprehension, user interaction, plan deduction, and facility and plan output are discussed and described. Prolog with its depth first search strategy, elegant data structures, simple syntax, and the ability to force or prevent search helps in identifying all possible alternative manufacturing processes and machines that can be used to manufacture a part. Detailed discussion and description on the actual process planning technique, system inputs, their assimilation and use, the deduction of alternate routes and methods, and information output methods are presented. Some examples of outputs derived from the system are presented.

6.0 ROUTE CHOICE AND DYNAMIC SHOP STATUS

6.1 *Introduction*

The purpose of this software segment of the prototype CAPP system is to maintain the dynamic shop status of the shop over time. While the previous, qualitative Prolog portion of the system devises process plans and manufacturing instructions for part production, every alternate route and technique devised is now read, considered, and evaluated with respect to the objective function which is the minimization of flowtime. The queues at every machine modelled in the pseudo-facility are monitored in terms of time units. The actual machining operations required to transform each part are considered along with the operational sequence and hierarchy in order to compute the manufacturing times. The choice of the appropriate route is made considering the alternatives available with respect to the dynamic shop sta-

tus, and the flowtime that each alternative requires to be processed through the system.

6.2 Information Used By The Dynamic Shop Status Module

To consider each part, compute it's manufacturing time, understand the actual processes required, their machining time, and repeat this process for each and every route, and choose between available alternatives the following information is used by the system:

- Part type under consideration.
- Part quantity.
- Part material.
- Machines that can be used to machine the part.
- Machining processes needed for this part.
- Dimensional characteristics.
- Surface finish desired.

The previous, qualitative software segment written in Prolog outputs into individual files the information mentioned above except information on specific part dimensions and quantity for every part considered by system software. This information is read in from individual files and stored in specific variables for use in the dynamic shop status module. The part type information, when read in, describes part profile information to the software. This helps in interactively requesting the user to supply specific dimensional information. The material type of the job under consideration is

read in from an individual file. It helps software assign speeds, feeds, and other machining parameters that could be used. This information is used in the machining time computation routines. The machining processes suggested by the Prolog inference engine are also read in. Information on part quantity is read in interactively. The user is prompted for relevant dimensions considering the part's construction. The surface finish desired is output from the previous software segment, and read in by this module written in FORTRAN. Appendix B presents a detailed listing of the FORTRAN code used.

An example of the information required by the dynamic shop status module to process a part that is generated by the qualitative Prolog software module includes:

- Part material - copper.
- Part type - rotational with L/D > 0.5
- Machines to be used - turning centers only.
- Processes needed to machine each surface with alternatives.
- Information on the hierarchy of processes.
- Information on which processes can be used for which surface.
- Part quantity information.
- Part dimensions.

Information that is read in interactively includes specific part dimensions and the part quantity. The part dimension information requested is dependent upon the specific part that is under consideration. The part dimension information requested for each part type is given in detail later in this chapter. The information mentioned above including information on part material, part profile specifications, machines that can be used, alternatives available, process hierarchy, etc. are read in from files written out by the qualitative Prolog software module. The processes required to

machine the part are specified and written out into individual files for each digit of the five digit input code by the Prolog software module. Each of these files needs to be read in by the dynamic system status module.

Also stored in this segment of system software is information on speeds and feeds for each material type considered by the system. This is used by dynamic shop status software module to decide upon the speeds, feeds, and depths of cut that can be used. The suggested speeds, feeds, and other machining parameters are presented in detail in Chapter Four. Depending upon the surface to be machined, specific machining time computations formulae are used. These are presented in detail in Appendix C. The actual formulae used to compute machining time for each surface machined are stored in this module of the dynamic shop status module.

Figure 36 illustrates the software structure, logic flow, and file handling that is used in reading in data from files and interactively from the user. On completion of execution of the qualitative Prolog software, the REXX control program executes the quantitative FORTRAN dynamic shop status module. The first submodule of the dynamic shop status module opens all the data files outputed by the qualitative Prolog segment. The operations suggested for each surface that requires machining are read. This information is used along with the description of each digit of the inputed code and the part material information to compute the machining time that would be required. The Prolog module suggests all possible alternatives that could be used. This is repeated for each alternative that can be used to generate a surface. Information on possible alternatives outputed by the qualitative Prolog segment is read in by this software module. If, for example, an external thread has to be machined it

could be done using a single point tool, a tap, or a thread cutting die. This information is output by the qualitative Prolog module.

Once each one of the available alternatives have been considered and all possible, available alternatives have been exhausted, the procedure is repeated for every machine that can be used. It is possible, for example, if the part is rotational and an external thread needs to be cut, then the part can be processed on any of the turning centers. The process and machine chosen are stored. This is then repeated for every digit under consideration. Once all the digits of the input code have been considered, the final plan is output to a file. This plan uses the route that minimizes flowtime. It is apparent from the discussion presented above that this software segment reads in and writes out information from multiple files.

Figure 37 illustrates the information that is read in and stored within the dynamic shop status module. The files that are read in by this software segment detail the machines that can be used to process a part. This assists the dynamic shop status module in assessing each alternative in conjunction with alternative processes that can be used. The use of this information is discussed above. The part material file specifies the actual part material that is used. This is used to identify the part's machining parameters. The part surface profile is stored along with the processes that can be used to generate the surfaces. This is done for each surface that needs to be machined, that is, for each digit described in the GT code. The interactive information regarding the quantity under consideration is used to compute machining times and flowtime of a part through the facility. Dimensions of each surface to be machined are read in interactively. The dimensions requested are dependent upon the part profile under consideration.

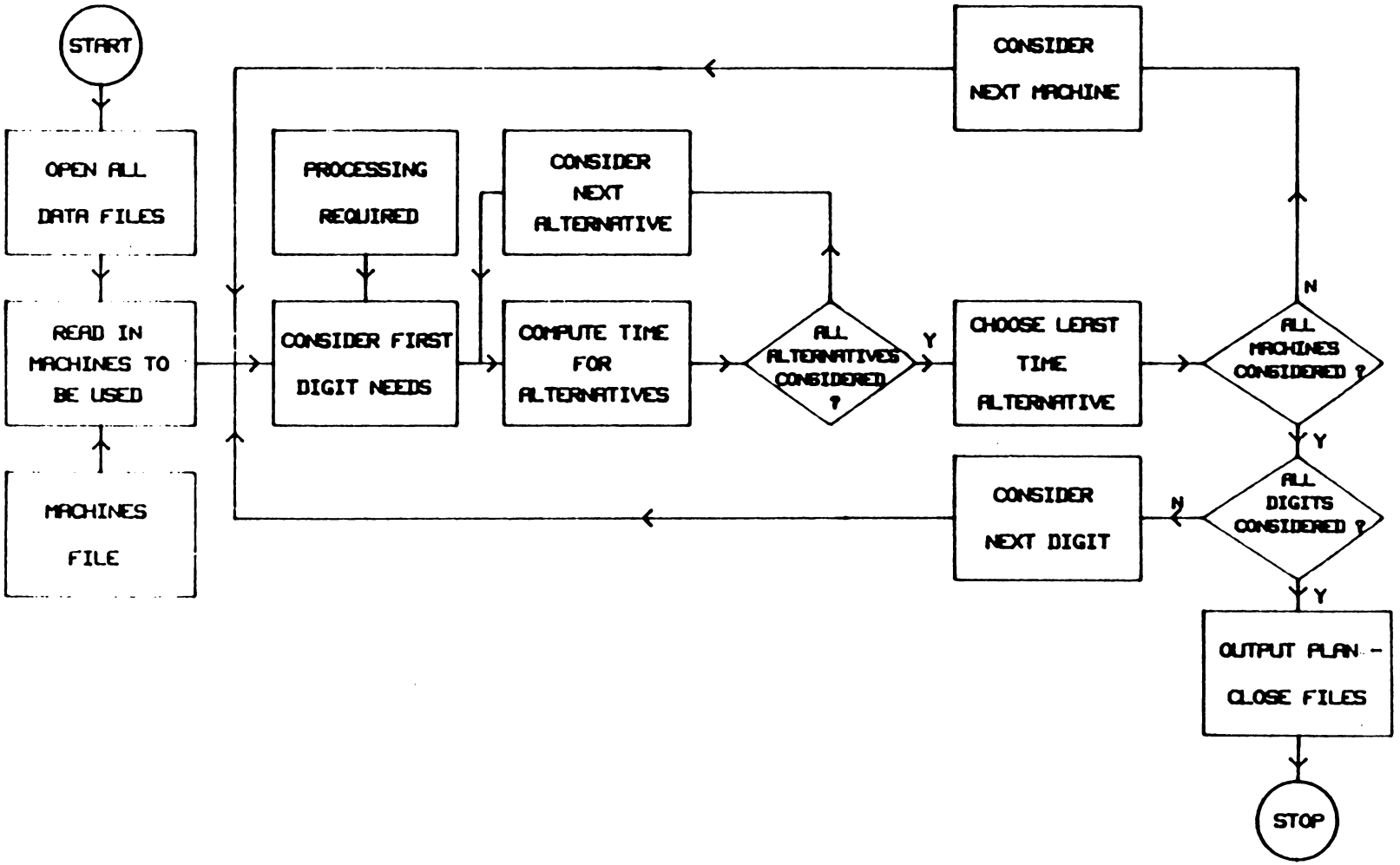


FIGURE 36 - SOFTWARE STRUCTURE, LOGIC FLOW AND FILE HANDLING

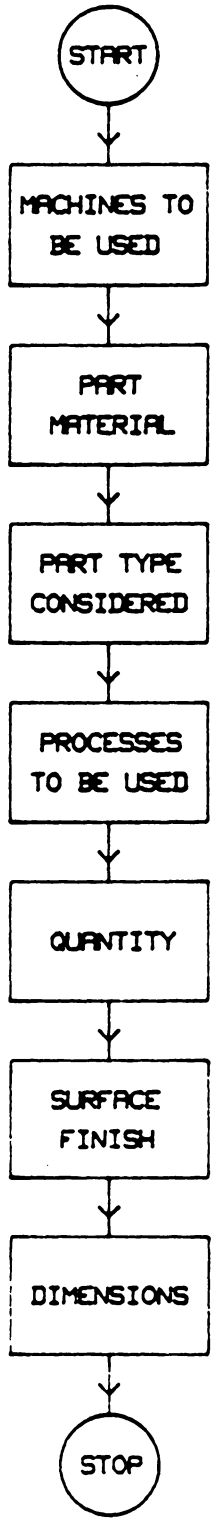


FIGURE 37 - INFORMATION USED BY DYNAMIC STATUS MODULE

6.3 Procedure And Logic Used

The first step that the dynamic shop status module performs is to open all the relevant files from the output generated by the qualitative Prolog module of system software. This is needed by the software module that maintains shop status to understand part characteristics that are relevant in deciding the actual processes to be used, in the computation of machining times, and in route decisions. The opening, utilization, and information output into files is discussed in detail in the section 6.2. All the files used are opened through software. Files used supply information on part machining processes, material, tolerances, etc. They are read and used whenever information is required.

The first piece of information read in is the material type. This information is used to assign speeds and feeds that could be used for all the machining processes considered. If, for example, the material information read input indicates that part material is copper, then this software segment searches for and retrieves the appropriate machining information on speeds, feeds, etc. These machining parameters are used in the actual computation of machining times in conjinction with part dimensions and the specifications of the machine used.

The next step is to understand part profile. The qualitative Prolog inference engine outputs individual files for each surface that needs to be machined. One file is output for each digit used in the GT code. These five files that contain surface profile machining information are read in. This gives software an idea of the part under consideration, and the machining processes that are required. The next step exe-

cuted is to interactively read in all the part dimensional information interactively. This is done considering the actual job type. If, for example, the job is rotational the user is prompted to enter job length and diameter. If the job is prismatic, then the user is prompted to enter job length, width, and height. Software considers input read in to decide upon the job type, and prompts the user for the appropriate dimensions. If, for example, the part is rotational and stepped to one side, software would ask for the depth and length of the step. These dimensional values are stored, and used to perform initial turning or surface machining time computations. The machining time computations used standard formulae. The machining time computation techniques stored in software are capable of comprehensively computing the machining time for every surface type that can be considered by Macro-CAPP. The details of the machining time computational techniques and formulae are presented in Appendix C.

Figure 38 describes the logic portrayed in software with respect to the input of information on part material type, job type, and the interactive input of machining dimensions. The logic of the interactive software module that reads in the initial dimensional values is described in Figure 39.

The next step is to identify the machines than can be used to process the part. This is output by the qualitative Prolog software module. The machines that can be used to produce the part are dependant upon the component class of a part. If, for example, a part is prismatic then it follows that the part can be machined on milling centers and not on turning centers. It can be argued that a four jaw chuck could be used be used to hold the part on a turning center. But this is not feasible in practice due to the long setup times that would be required. A rotational part is machined predominantly on a turning center. Each one of the machines that are suggested by

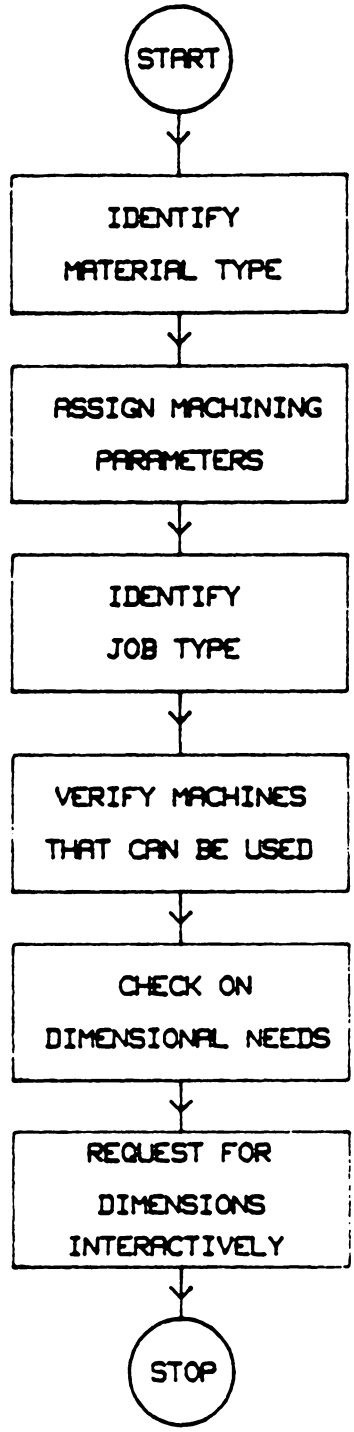


FIGURE 38 - INITIAL INFORMATION INPUT UTILIZATION

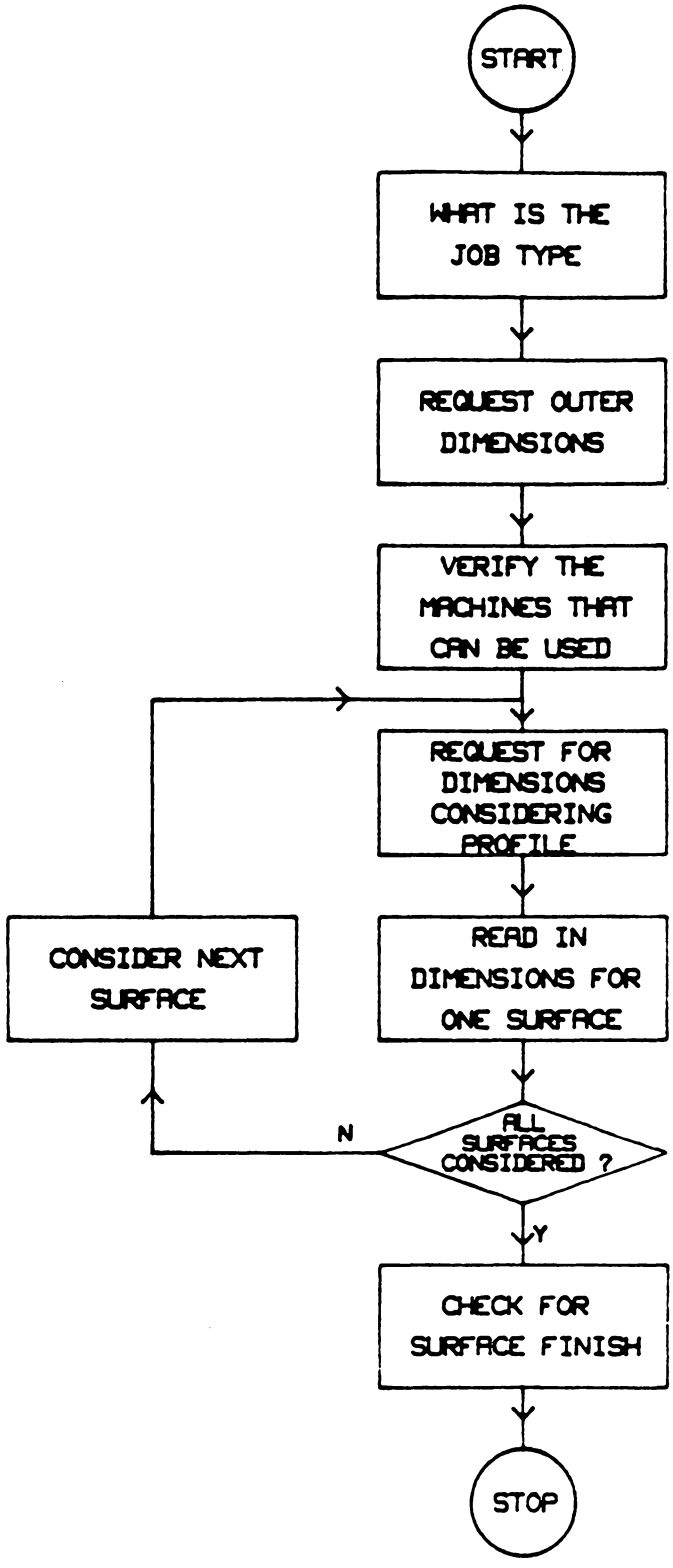


FIGURE 39 - LOGIC OF INITIAL DIMENSIONAL INPUT

the Prolog software module are evaluated for the processes suggested. If the part is prismatic, its length, width, and height are used to verify if it can be loaded on to all machines that are output by the Prolog software. If the part is rotational, then it's length and diameter are used to check if both the turning centers modelled in the facility can be used to machine the part. This step ensures that no job that exceeds the specifications and capacity of a machine is loaded on that machine.

If a part is loaded on to a specific machine, then all required operations that can be performed on this machine are executed before the part is moved to another machine. If, for example, a rotational part that is to be manufactured requires to be turned, threaded, knurled, and faced, it would not be practical to execute each of these operations on a different machine, loading and unloading the job on each occasion. It would be practical to identify the machine that would process the part subject to scheduling constraints, and then perform all the possible operations on that machine before moving the job to another machine. This practical approach would reduce set up costs and times, and avoid delays due to unloading jobs from one machine, and then loading them on to another. Figure 40 illustrates the overall logic flow used in this segment of software.

Every alternative process and route that can be used to machine a part is output by qualitative Prolog software module. This is inputed to the dynamic system status module written in FORTRAN. Every alternative inputed needs to be evaluated with respect to the performance measure under consideration, namely the minimization of flowtime that would be required if the part was machined using a specific process. The surfaces are considered in the actual order that they must be machined in. This order is identical to that suggested by the Prolog software to machine and produce

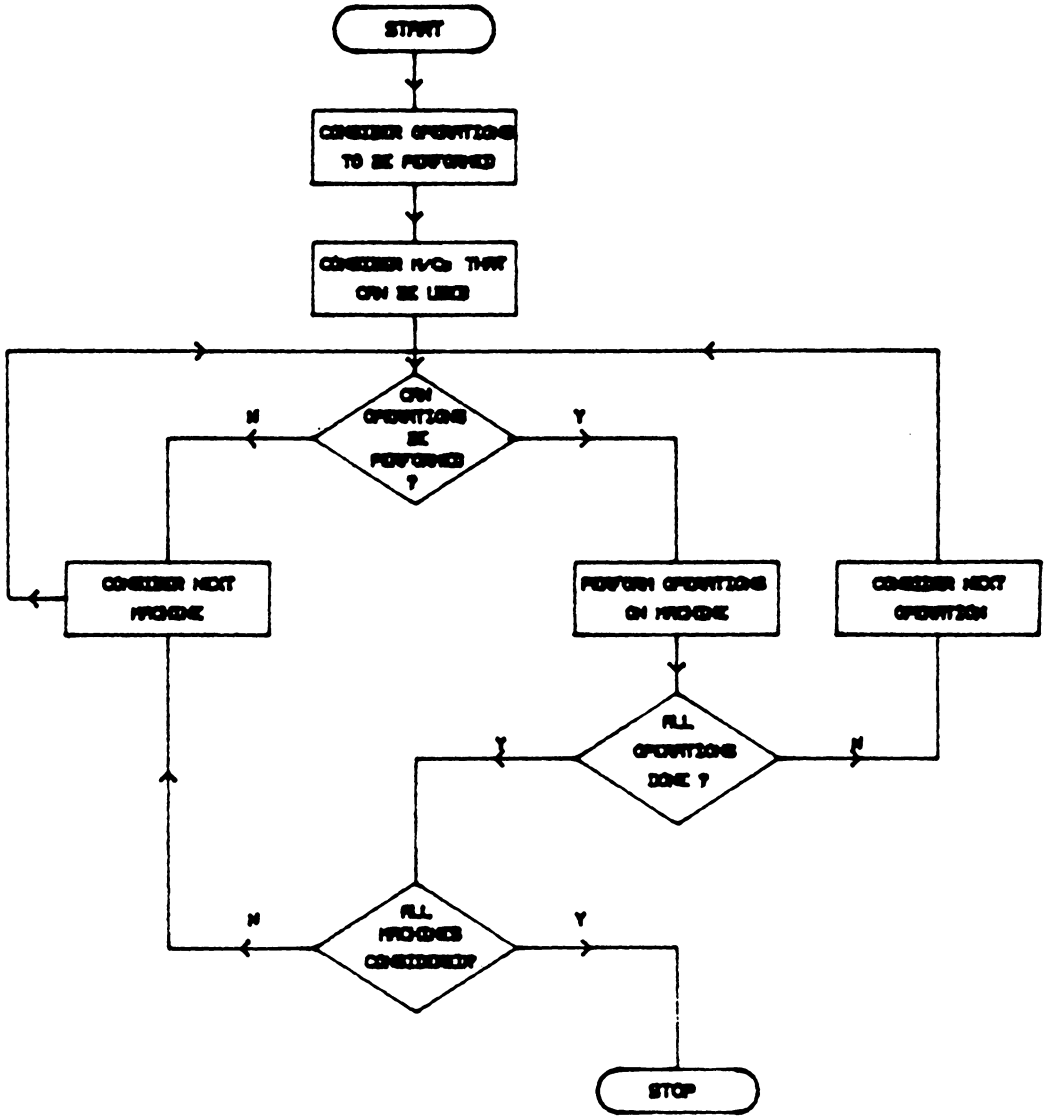


FIGURE 40 - OVERALL LOGIC FLOW

the part. By choosing the minimization of actual flowtime through the system as the performance measure, the shop load is captured in the decision on which route to follow. The queue length, for example, would represent the number of jobs within the facility or in queue waiting to be processed, but would not consider the possible variations in machining times between different part types. The flowtime that a job may require through a specific route is an accurate measure of shopload on that route at that instant.

In deciding upon the route to be followed through the system, the first machining process considered in the machining time evaluation is that used to machine the external part surface. All possible processes suggested are considered, and the process that uses the least amount of flowtime is chosen. If, for example, a plane external surface has to be generated, it could be done using different milling procedures. Each alternative process is evaluated with respect to a chosen machine. In this case either of the two milling centers could be used. The alternative that would require the least machining time is chosen considering every machine that can handle the processes required to machine a part. Since the flowtime of each queue at each machine at any is known in time units, the process that could ensure the least flowtime for the part under consideration is chosen. Figure 41 illustrates the working of this software segment.

This process is repeated for the second set of processes. These are those used in the machining of through bores for rotational or prismatic parts. It is known at this juncture what machines can be used to machine a part. For each machine the individual processes are evaluated and the process that requires the least machining time is chosen. This is repeated for all the machines in consideration. The machine

that would result in the least job flowtime is chosen along with the processes that would be used. Figure 42 illustrates the working of the software segment that chooses the processes and machines that are used to machine the internal through bores. Interactively, the user is prompted by the system to provide relevant dimensions such as the diameter and number of bores, the length of the bores, etc. Machining time computations are based on this information as also on any information that might have been entered earlier.

The next concern is the external plane surface machining that may be required. Chapters Three and Five illustrate the external plane surfaces considered for both prismatic and rotational parts. All plane surface machining is to be done on milling centers. Each available alternative is considered. As mentioned above, the choice of the alternative machines and processes is output by the qualitative Prolog software segment. The machining times required for each operation is considered, and the operation that ensures the least increase in machining time is chosen. This is repeated for the other milling center modelled in the pseudo-facility also. At this stage, the operation and the machine that causes the smallest increase in job flowtime through the facility are chosen. Figure 43 illustrates the working of the software segment that chooses the processes and machines that are used to machine external plane surfaces. This software module prompts the user to enter any relevant dimensional data that may be required to help compute machining times. The dimensions the user is prompted to enter is dependent upon the part profile. It could be, for example, the length, width, and height of the plane surface to be machined.

The last feature considered by this software segment is the holes that are drilled into the part. The actual hole classifications considered are presented in Chapter

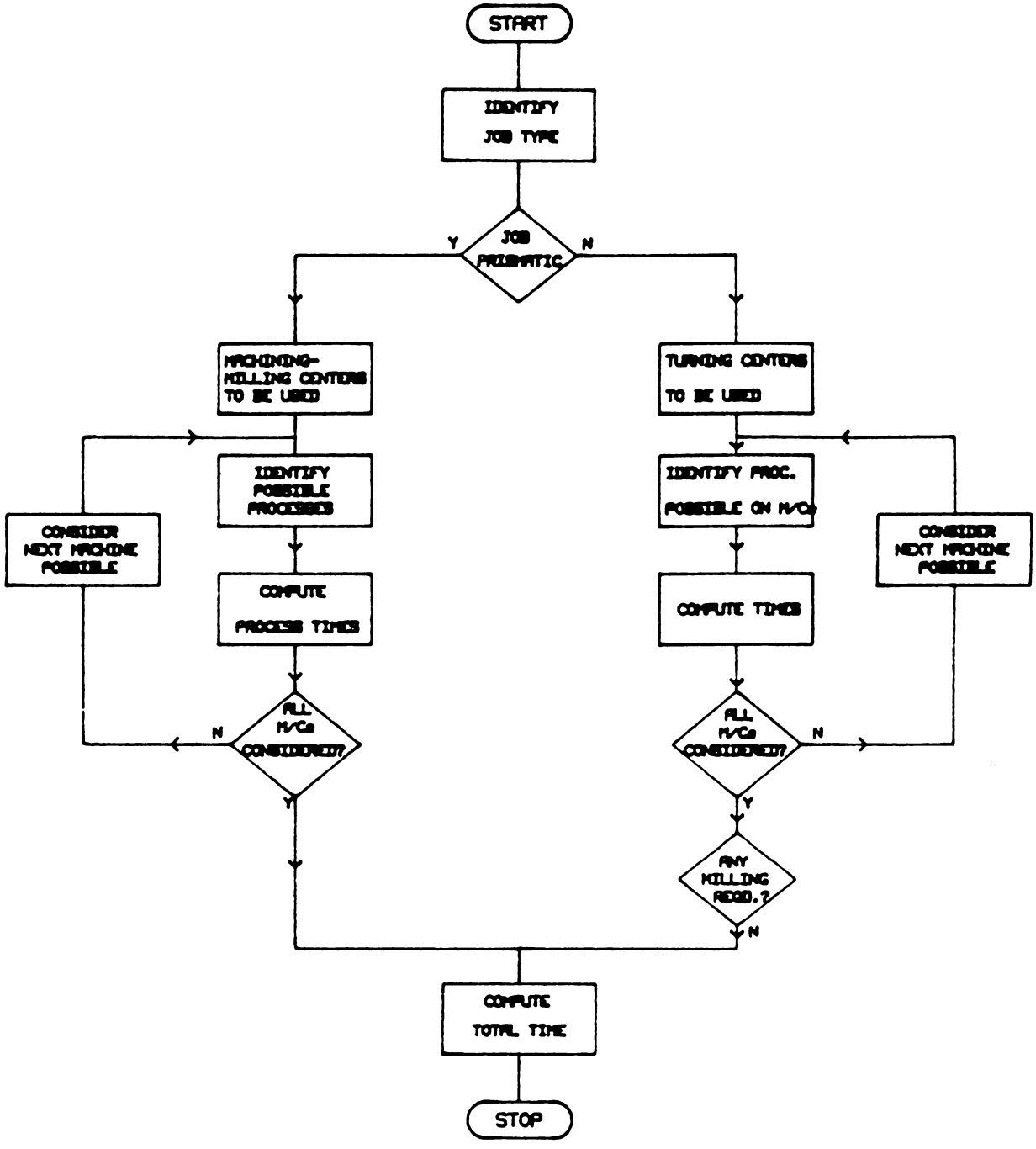


FIGURE 41 - LOGIC USED TO DECIDE INITIAL MACHINING REQUIREMENTS

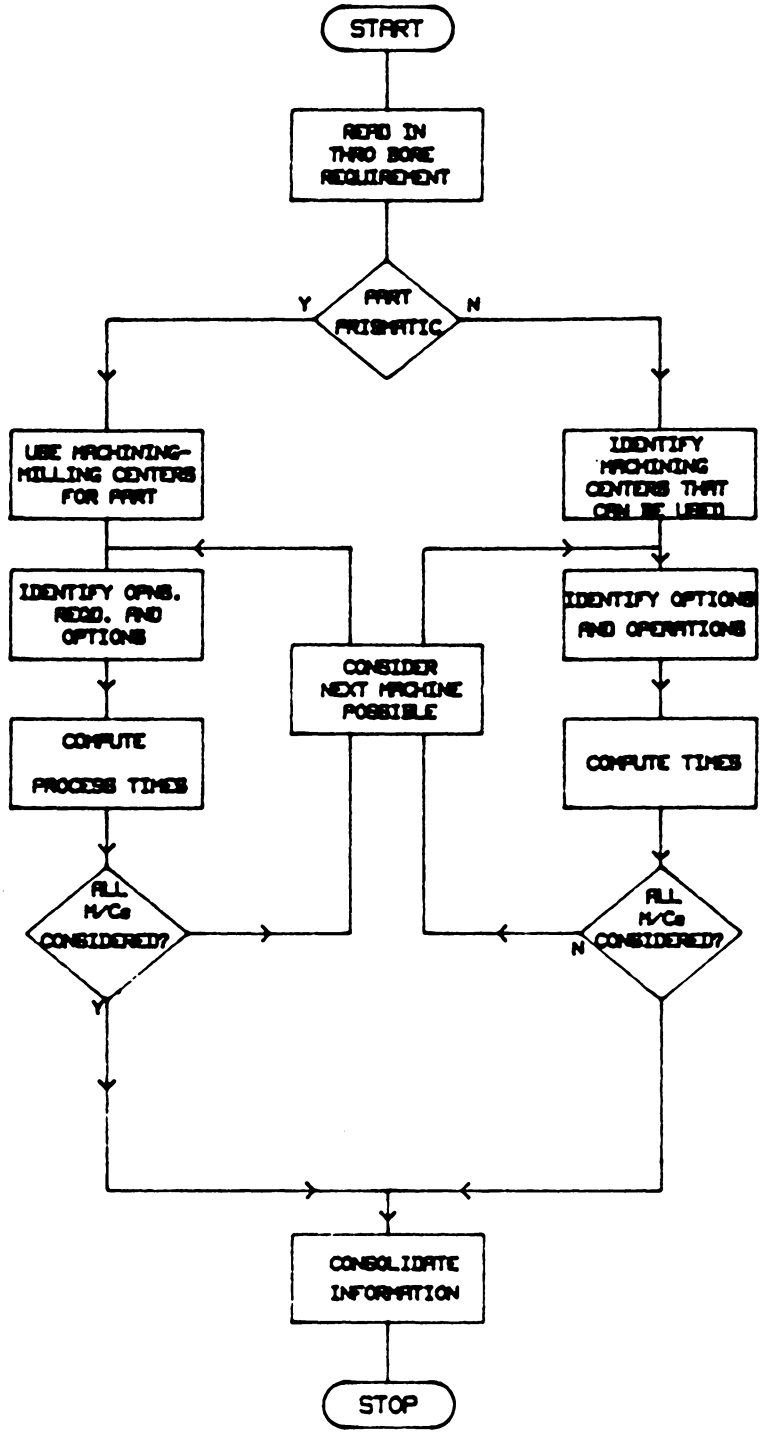


FIGURE 42 - SYSTEM LOGIC - PROCESSES AND MACHINES TO MACHINE THROUGH BORES

Three. The machining operations required are decided depending upon the actual requirements. The holes, in most cases, needs to be drilled and reamed. Multiple drilling or gang drilling may be used depending upon the location, direction, and orientation of the holes. The user is prompted to enter information on the actual number of blind holes, their length, depth, etc. These pieces of information are used to compute machining times for all machining operations used in the generation of holes in this prototype system. The information required at this stage is dependent upon the part profile information inputed by the part description code. The dimensions requested at this stage are very detailed in nature. They include information on the diameter, depth, and number of holes.

The machines that can be used to drill the holes have already been identified. They are output by the qualitative Prolog module of system software. Each one of the operations that can be used to to generate the holes are considered, and their actual machining times computed. It must be remembered that on turning centers, a hole can be drilled into a rotational part only along the axis of rotation. Holes not on the axis need to be drilled on milling centers. The appropriate operation that would result in the least machining time is identified. This is then repeated for the other machines that can be used to perform the same job, and the machine that would result in the least flowtime considering current shop status is chosen along with the appropriate manufacturing techniques. Figure 44 illustrates the working of the software segment that chooses the processes and machines that are used to generate both radial and axial holes in the job.

While software identifies all the available alternatives operations available with respect to the manufacture of a component, it is required at each stage to output

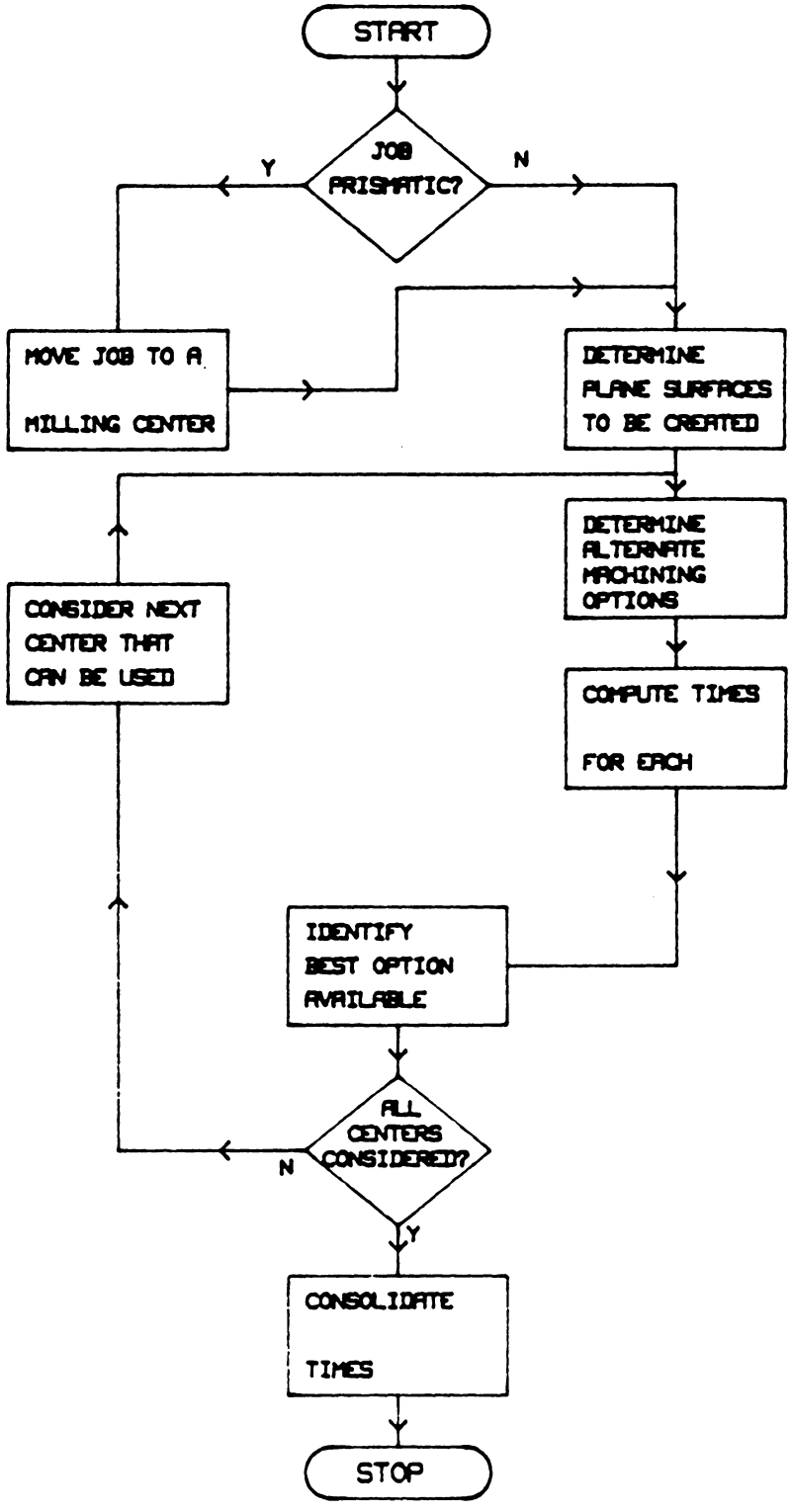


FIGURE 43 - SYSTEM LOGIC-EXTERNAL PLANE SURFACE MACHINING

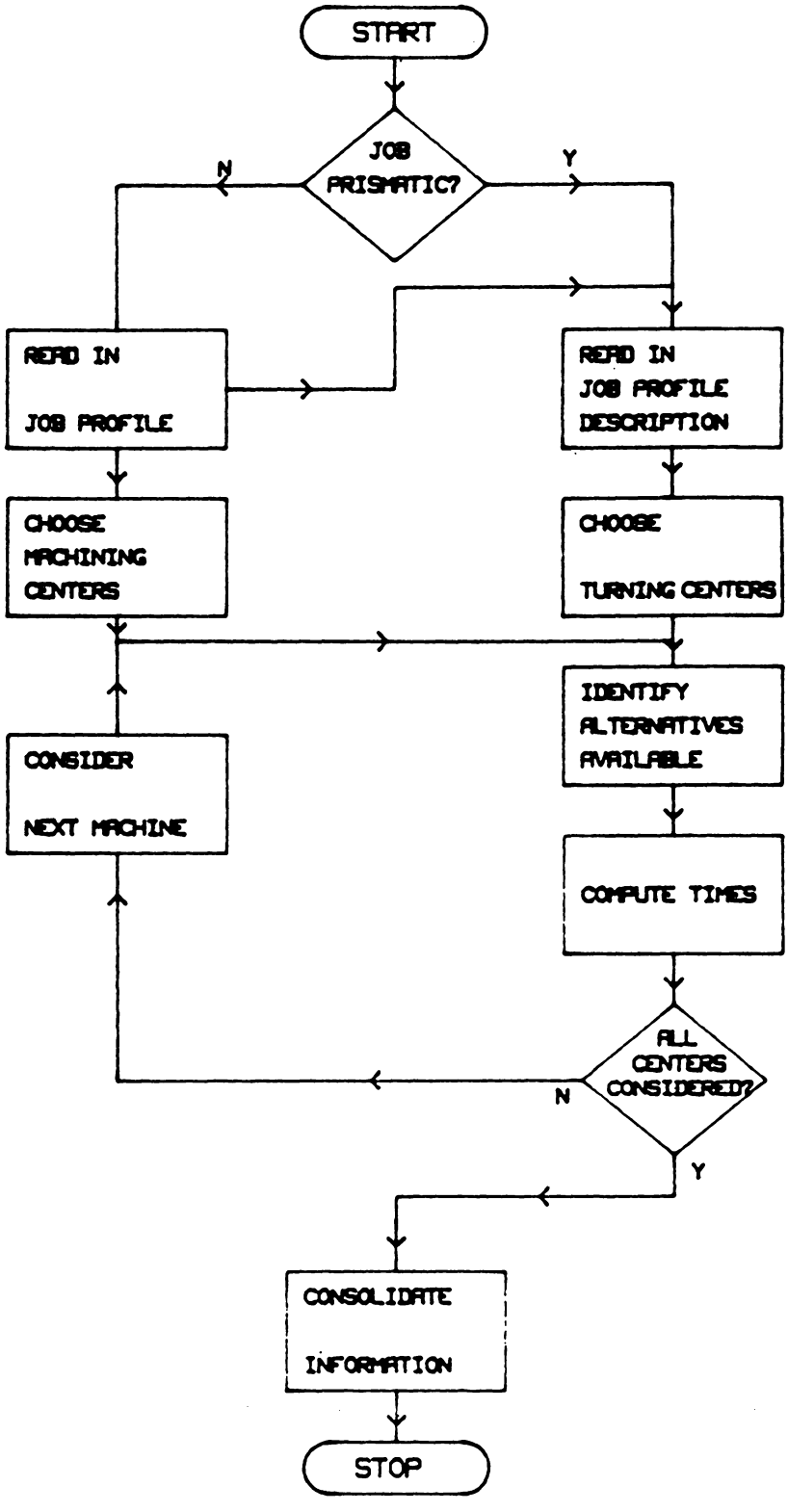


FIGURE 44 - SYSTEM LOGIC-RADIAL AND AXIAL HOLE DRILLING

process plans along with adequate computations of machining times, so that these can be used by the shopfloor personnel. They will use the alternatives generated along with the route, speeds, feeds, sequence of operations, and the machining processes suggested in planning for actual part production. Figure 45 contains a process plan output generated by the dynamic shop status module. It contains the material type to be used, cutting parameters suggested, route and operations to be used, along with outputs concerning the machining times and job flowtime estimated. Chapter Seven contains examples of parts considered by the prototype system along with outputs generated. Detailed analysis of the outputs along with the advantages and drawbacks are presented in this chapter.

At this point, it needs to be emphasized that this prototype has been developed to serve as a component part of a CIM system that works on a real time basis to help in process planning, scheduling, and routing. The prototype is continually active, keeping track of shop status and loads. This ensures that the shop status considered by the prototype CAPP system is realistic and accurate. A clock that keeps track of the passage of time was implemented. This ensures that the actual arrival time of a job, time between arrivals, and the impact of the time factor on shop status and queue lengths were considered. Presented in Figure 46 is the method implemented in software to keep track of the passage of time between job arrivals, and incorporate this factor in the actual prototype design and implementation. This requires software to acquire from the computer system the time at when a lot arrives to the system. Since the time of arrival of the previous lot to the facility is stored, the time difference between the arrival of lots to the system is computed. This value is subtracted from queue lengths stored in time units. The queue length values then reflect the passage

```
THE MATERIAL OF THE PART IS COPPER
QUANTITY 20000

SPEED 280     FEED 0.025

USE MACHINE ONE - TURNING CENTER
QUEUE TIME 4456.43

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  4.00     DIAMETER  2.00
SPEED =  560 TIME = 5714.29

SMOOTH INTERNAL HOLES - DRILL AND REAM
LENGTH OF HOLE =  4.00  DIAMETER  0.5
LENGTH OF STEP =  2.00  DIAMETER  0.25
SPEED = 2240 TIME =  2142.86

NO PLANE SURFACE MACHINING REQUIRED

NO HOLES IN PART - NO DRILLING REQUIRED

USE MACHINE TWO - TURNING CENTER
QUEUE TIME 7789.12

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  4.00     DIAMETER  2.00
SPEED =  560 TIME = 5714.29

SMOOTH INTERNAL HOLES - DRILL AND REAM
LENGTH OF HOLE =  4.00  DIAMETER  0.5
LENGTH OF STEP =  2.00  DIAMETER  0.25
SPEED = 2240 TIME =  2142.86

NO PLANE SURFACE MACHINING REQUIRED

NO HOLES IN PART - NO DRILLING REQUIRED

USE MACHINE FIVE - TURNING CENTER
QUEUE TIME 8856.63

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  4.00     DIAMETER  2.00
SPEED =  560 TIME = 5714.29

SMOOTH INTERNAL HOLES - DRILL AND REAM
LENGTH OF HOLE =  4.00  DIAMETER  0.5
LENGTH OF STEP =  2.00  DIAMETER  0.25
SPEED = 2240 TIME =  2142.86

NO PLANE SURFACE MACHINING REQUIRED

NO HOLES IN PART - NO DRILLING REQUIRED

USE MACHINE ONE - TURNING CENTER
```

Figure 45. A Sample Output Describing Processes, Route, And Machining Times

of time. These queue length values are used in determining the route to be used. The techique used is described in Figure 46.

6.4 Conclusion

This objective of this segment of the prototype CAPP system is to consider part quantity, part profile, the machining processes required in their actual hierarchy of operations, the part's dimensions, material type, and surface finish desired in deciding upon the job route, the operations to be performed on a specific machine, and the actual machining times estimated. The dynamic status of the shop modelled is monitored on a real time basis to ensure that the heuristic used is applied in an accurate manner. Details of the logic represented in system software is explained in a detailed manner through the use of a variety of flowcharts. Samples of outputs derived are also presented. This segment of the prototype system helps the CAPP portion of the software integrate with other CIM constituents such as loading, routing, scheduling, and capacity planning.

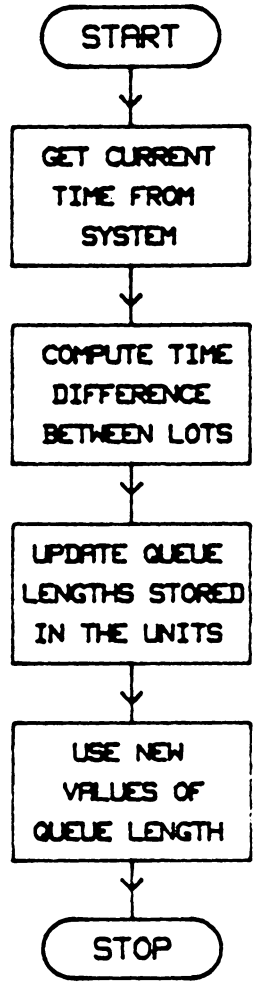


FIGURE 46 - TECHNIQUE USED TO KEEP TRACK OF TIME

7.0 RESULTS AND ANALYSIS

7.1 Introduction

The prototype CAPP system was tested with respect to a variety of parts that were both rotational and prismatic in construction. Presented in this chapter are outputs derived at every stage of the process planning procedure for one rotational and one prismatic part. The files generated and outputed by the qualitative Prolog segment of the process planning software were checked and validated. Manual process plans were developed for both sample parts assuming similar manufacturing facilities and conditions. The process plans, routes, and machining parameters suggested by the CAPP system developed were checked against the manual process plan for logic and technical accuracy. The operations suggested, their hierarchy, speeds, and machining parameters suggested were verified.

The input reading mechanism of the dynamic shop status module was verified. The inputs supplied to system software was written out and manually verified to en-

sure that the software was reading in the information supplied accurately and in the format desired. Machining time computations along with the queue tracking mechanism was checked. This was done by manually computing the times associated with each operation using the formulae and technical specifications suggested. The values computed manually were compared using those computed by Macro-CAPP. The implementation and utilization of the route determination heuristic were checked. An accurate mechanism that kept track of the passage of time was implemented to help the system work on a real-time basis.

Presented below are process plans developed through the use of Macro-CAPP for rotational and prismatic components. The outputs derived at each stage in the planning procedure are given. The processes suggested are presented to help in the system validation process. The advantages that dynamic systems provide over static systems is evident. This is also evident from comparing the depth and variety of manufacturing information and instructions provided on a global basis by Macro-CAPP with process planning systems such as Turbo-CAPP [83], APPAS [90], GENPLAN [74], TIPPS [9], etc.

7.2 Planning For Rotational Components

The rotational component considered in this example is described in Figure 47. It contains the following features:

- Stepped to one side.
- One through bore on the axis.
- Four auxiliary holes related by graduation around a circle.

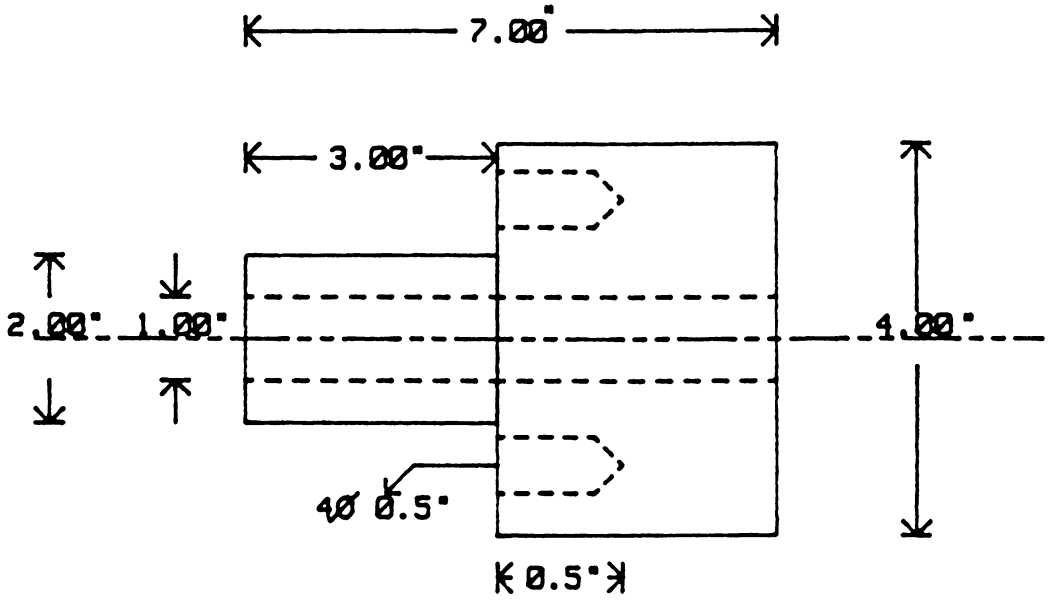
- Chamfers present on external surface on step.
- Part material is aluminum.
- Surface finish desired on all machined surfaces is 64 micro inches.
- Diameter of part is four inches.
- Diameter of step is two inches.
- Length of step is three inches.
- Overall part length is seven inches.
- Diameter of through bore one inch.
- Diameter of auxiliary holes is half an inch.
- Depth of auxiliary holes is half an inch.
- Initial material type is bar stock.
- Quantity to be produced is ten thousand.

Since the part is made of aluminum, appropriate machining parameters as described in Chapter Four are used. The Length/Diameter (L/D) ratio is 7/4 which is equal to 1.75. This ratio, that classifies the part as rotational, is required in entering the appropriate GT code. All the part description information presented above is required in the process plan deduction strategy used by Macro-CAPP.

The part description code was entered after activating the system. It must describe to system software the actual part construction in a comprehensive manner. The group technology code used was:

- rotational - 2.
- outer_stepped_one_side - 2.
- internal_smooth - 2.
- surface_plane_none - 1.
- holes_axial_related - 3.

This was then followed by the auxiliary inputs to the system. This included:



QUANTITY = 10000

MATERIAL = ALUMINIUM

FIGURE 47 - EXPERIMENTAL ROTATIONAL COMPONENT CONSIDERED

- Surface finish - 64 micro inches.
- Initial material - bar stock.

These inputs were followed by the actual execution of the system's process planning software that would derive the alternate routes and plans to process the part. Presented in Figure 48 is a comprehensive picture of the output derived from the CAPP module written in Prolog. This includes information on part surface profile, operations and machines suggested for each surface to be machined, machining parameters, part material, and tolerances. The final process plans are written out by the dynamic shop status module.

It is important to note that this output is not the final process plan. It is a collection of the processes and alternative methods that can be used to generate each surface. It is apparent from a perusal of Figure 48 that the hierarchy of operations required is maintained. Since this is not the final process plan used, but only contains inputs to the dynamic process selection module, it does not contain information on the actual processes to be used or on the suggested machining parameters.

The final process plan developed is presented in Figure 49. It contains information on the processes suggested, machines that can be used, alternatives, potential flowtimes for each alternative, information on queues, processes that Macro-CAPP suggests for use, and the appropriate machining parameters. It provides the user with a comprehensive picture of the shop's status, and the reason for choosing a specific route. For each surface that needs to be machined possible alternatives are suggested. The alternative that would require the least flowtime is chosen.

```
ROTATIONAL

MACH1
MACH2
MACH5

OUTER_SINGLE_STEP
MULTIPLE TURNING REQUIRED

INTERNAL SMOOTH
DRILL
REAM

EXTERNAL_PLANE_NONE

HOLES_AXIAL_RELATED
MULTIPLE DRILL
MULTIPLE REAM

MATERIAL STAINLESS STEEL

INIT MATERIAL  BAR STOCK

SURFACE FINISH 125
```

Figure 48. Prolog Output For Rotational Component

```
ROTATIONAL
OUTER_SMOOTH
INTERNAL_SMOOTH
OUTER_PLANE_MACHINING_NONE
HOLES_AXIAL_RELATED

THE MATERIAL OF THE PART IS STAINLESS STEEL
QUANTITY 10000
OVERALL LENGTH 7.00
DIAMETER OF HOLE 1.00
SURFACE FINISH  125

SPEED  90   FEED 0.020

USE MACHINE ONE - TURNING CENTER
QUEUE TIME 3456.80

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  7.00    DIAMETER = 4.00
SPEED =   90 TIME = 38888.89

INTERNAL_SMOOTH - DRILL AND REAM
LENGTH OF HOLE =  7.00  DIAMETER = 1.0
SPEED =  360 TIME =  9722.22

NO PLANE SURFACE MACHINING REQUIRED

AXIAL HOLES_RELATED
LENGTH OF HOLE = 0.5  DIAMETER = 0.5
NO. OF HOLES = 4
SPEED = 720  TIME = 347.22


USE MACHINE TWO - TURNING CENTER
QUEUE TIME 5543.52

THE PART IS ROTATIONAL - TURN PART
THE PART LENGTH =  7.00    DIAMETER = 4.00
SPEED =   90 TIME = 38888.89

INTERNAL_SMOOTH - DRILL AND REAM
LENGTH OF HOLE =  7.00  DIAMETER = 1.0
SPEED =  360 TIME =  9722.22

NO PLANE SURFACE MACHINING REQUIRED

AXIAL HOLES_RELATED
LENGTH OF HOLE = 0.5  DIAMETER = 0.5
NO. OF HOLES = 4
SPEED = 720  TIME = 347.22


USE MACHINE FIVE - TURNING CENTER
QUEUE TIME 8799.93

THE PART IS ROTATIONAL - TURN PART
```

Figure 49. Final Process Plan For Rotational Component

```
THE PART LENGTH =  7.00    DIAMETER = 4.00
SPEED =   90 TIME = 38888.89

INTERNAL_SMOOTH - DRILL AND REAM
LENGTH OF HOLE =  7.00  DIAMETER = 1.0
SPEED =  360 TIME =  9722.22

NO PLANE SURFACE MACHINING REQUIRED

AXIAL HOLES_RELATED
LENGTH OF HOLE = 0.5  DIAMETER = 0.5
NO. OF HOLES = 4
SPEED = 720  TIME = 347.22

USE MACHINING CENTER ONE - TURNING CENTER
```

Figure 50. Final Process Plan For Rotational Component - Continued

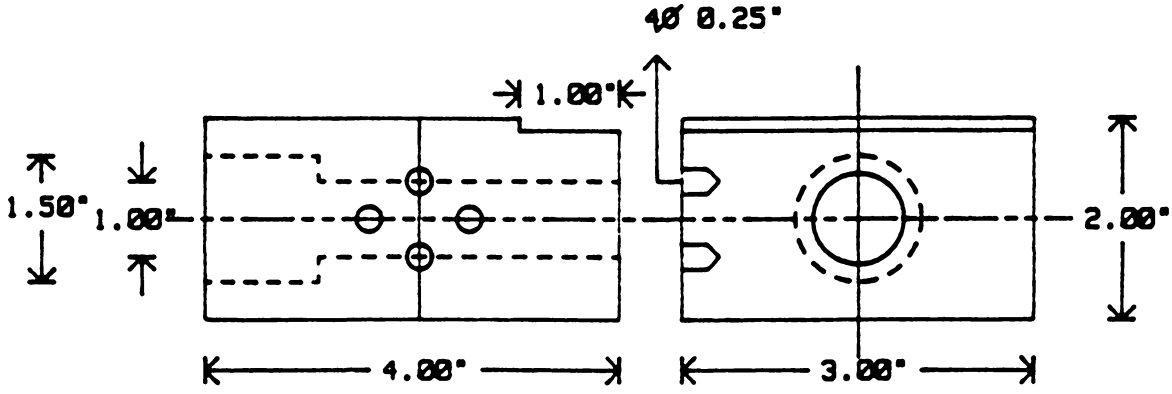
The alternatives presented in the figure help choose the machine that the part can be processed upon. Any of the turning centers can be used. The part is rotational. Its dimensions when checked allow for it to be mounted on any of the turning centers modelled in the facility. The multispindle machines used are capable of performing all the operations required to produce the part. The outer surface must be turned to a diameter of four inches with a part length of seven inches. The next operation would be turning down the external diameter to form a step with a diameter of two inches and a length of three inches. More than one cut may be used to arrive at this step. The through bore required may be drilled and then reamed. The auxiliary holes may be generated using a multispindle arrangement on the machine.

It is possible to compute the potential machining times to generate each cut and complete each process for every surface generated. The outputs generated that specify the machining times are presented in Figure 49. This can be then computed for each alternate route that could be used. The heuristic is then applied through software and the route with the least flowtime is chosen. This is evident from a perusal of the process plan presented in Figures 49 and 50.

7.3 Planning For Prismatic Components

The prismatic component considered by this example is as described in Figure 51.

- Length of edges is four, three, and two inches.
- One principal through bore stepped to one side.



QUANTITY = 20000
MATERIAL = ALUMINUM

FIGURE 51 - EXPERIMENTAL PRISMATIC COMPONENT CONSIDERED

- Chamfers present on external surface on step.
- Radial auxiliary holes, related by graduations around a circle.
- Part material is aluminum.
- Surface finish desired on all machined surfaces is 64 micro inches.
- Depth of external step is one tenths of an inch.
- Diameter of through bore is one and one half inches.
- Diameter of auxiliary holes is one quarter of an inch.
- Length of step is one inch.
- Depth of auxiliary holes is half an inch.
- Initial material type is prismatic cast aluminum blanks.
- Quantity to be produced is twenty thousand.

Since the part is prismatic in nature the ratio of A/B is 1.33, and the A/C ratio is 2. Analysis of the A, B, C ratios would denote the part as cubic. It is now required to enter the GT code to describe to system software the part's construction. The code would be as follows:

- cubic - 6.
- outer_plane_machining - 1.
- internal_stepped_to_one_side - 3.
- surface_plane_step - 3.
- holes_axial_related - 3.

This would then be followed by the auxiliary inputs to the system. This would include:

- Material - aluminum.
- Quantity - twenty thousand.
- Surface finish - 64 micro inches.
- Initial material - cast aluminum blanks.

This information was input to Macro-CAPP. It was then followed by the actual execution of the reasoning mechanism embedded in the inference engine. Prolog output

was written into files which contained information on part surface profile, machining processes and procedures, machines suggested, part material, and tolerances. This information is presented in Figure 52, and was read in by the dynamic system status module.

Although this does not constitute the final process plan, it gives the user a collection of the processes and alternative methods that can be used to generate each surface. This output also exhibits the fact that the hierarchy of operations required is maintained. No information is presented on the actual processes to be used or on the suggested machining parameters.

The final process plan developed for the cubic component is presented in Figure 53 and 54. As in the previous example, it contains information on the processes suggested, machines that can be used, alternatives, potential flowtimes for each alternative, information on queues, processes that Macro-CAPP suggests, and the appropriate machining parameters. The user is provided with a comprehensive picture of the shop's status, and the reason for choosing a specific route. The alternatives presented in the figure help choose the machine that the part can be processed upon. Any of the milling centers can be used. The part is cubic, and it's dimensions when checked allow for it to be mounted on any of the milling centers modelled in the facility.

Milling centers modelled in the facility are capable of performing all the operations required to produce the part. The outer surface must be milled to a a length of four inches, breadth of three inches, and a height of two inches. The internal bore with the step is created using multiple drilling, reaming, and boring operations. The auxiliary holes are then drilled in an axial direction using multiple spindle arrange-

```
CUBIC

MACH3
MACH4

OUTER_PLANE-MACHINING
SIDE MILL
FACE MILL
PLAIN MILL

INTERNAL STEPPED ONE SIDE
MULTIPLE DRILLING
MULTIPLE REAMING

EXTERNAL_PLANE_SINGLE_STEP
FACE MILLING
SIDE MILLING
PLAIN MILLING

HOLES_RADIAL_RELATED
MULTIPLE DRILL
MULTIPLE REAM

MATERIAL STAINLESS STEEL

INIT MATERIAL  CAST BLANKS

SURFACE FINISH  64
```

Figure 52. Prolog Output For Prismatic Component

```
CUBIC
OUTER_PLANE_MACHINING
INTERNAL_SINGLE_STEP
OUTER_SINGLE_STEP
HOLES_RADIAL_RELATED

THE MATERIAL OF THE PART IS ALUMINUM
QUANTITY 20000
OVERALL LENGTH 4.00
OVERALL WIDTH 3.00
OVERALL HEIGHT 2.00
SURFACE FINISH  64

SPEED  550   FEED 0.045  MILL FEED 0.003

USE MACHINE THREE - HORIZONTAL MACHINING CENTER
QUEUE TIME 872.98

THE PART IS PRISMATIC - FACE MILLING REQUIRED
THE PART LENGTH =  4.00    PART WIDTH = 3.00
SPEED =   550 TIME = 3367.00

THE PART IS PRISMATIC - SIDE MILLING REQUIRED
THE PART LENGTH =  4.00    PART WIDTH = 3.00
SPEED =   550 TIME = 5050.50

THE PART IS PRISMATIC - PLAIN MILLING REQUIRED
THE PART LENGTH =  4.00    PART WIDTH = 3.00
SPEED =   550 TIME = 3367.00

USE FACE MILLING

INTERNAL_SINGLE_STEP - MULTIPLE DRILL AND REAM
LENGTH OF HOLE =  4.00  DIAMETER = 1.50
LENGTH OF STEP = 1.00   DIAMETER = 1.00
SPEED =  3300 TIME =  9722.22

PLAIN_EXTERNAL_SINGLE_STEP - FACE MILLING REQUIRED
LENGTH OF PART = 3.00 WIDTH OF STEP = 1.00
SPEED = 550  TIME = 2693.00

PLAIN_EXTERNAL_SINGLE_STEP - PLAIN MILLING REQUIRED
LENGTH OF PART = 3.00 WIDTH OF STEP = 1.00
```

Figure 53. Final Process Plan For Prismatic Component

```
SPEED = 550  TIME = 2693.00

RADIAL HOLES_RELATED - MULTIPLE DRILLING AND REAMING
LENGTH OF HOLE = 0.25  DIAMETER = 0.5
NO. OF HOLES = 4
SPEED = 1000  TIME = 55.55

USE MACHINE FOUR - VERTICAL MACHINING CENTER
QUEUE TIME 6798.34

THE PART IS PRISMATIC - FACE MILLING REQUIRED
THE PART LENGTH =  4.00    PART WIDTH = 3.00
SPEED =   550 TIME = 3367.00

THE PART IS PRISMATIC - SIDE MILLING REQUIRED
THE PART LENGTH =  4.00    PART WIDTH = 3.00
SPEED =   550 TIME = 5050.50

THE PART IS PRISMATIC - PLAIN MILLING REQUIRED
THE PART LENGTH =  4.00    PART WIDTH = 3.00
SPEED =   550 TIME = 3367.00

USE FACE MILLING

INTERNAL_SINGLE_STEP - MULTIPLE DRILL AND REAM
LENGTH OF HOLE =  4.00  DIAMETER = 1.50
LENGTH OF STEP = 1.00   DIAMETER = 1.00
SPEED =  4300 TIME =  9722.22

PLAIN_EXTERNAL_SINGLE_STEP - FACE MILLING REQUIRED
LENGTH OF PART = 3.00 WIDTH OF STEP = 1.00
SPEED = 550  TIME = 2693.00

PLAIN_EXTERNAL_SINGLE_STEP - PLAIN MILLING REQUIRED
LENGTH OF PART = 3.00 WIDTH OF STEP = 1.00
SPEED = 550  TIME = 2693.00

RADIAL HOLES_RELATED - MULTIPLE DRILLING AND REAMING
LENGTH OF HOLE = 0.25  DIAMETER = 0.5
NO. OF HOLES = 4
SPEED = 1000  TIME = 55.55


USE MACHINE 3 - HORIZONTAL MACHINING CENTER
```

Figure 54. Final Process Plan For Prismatic Component - Continued

ments. These holes then need to be reamed and finished. More than one cut may be used to arrive at individual surfaces.

The potential machining times required to generate each cut, and complete each process is computed. This is then computed for each alternate route that could be used. The heuristic is then applied through software and the route with the least flowtime is chosen. This is evident from a perusal of the process plan presented in Figure 52.

7.4 Auxiliary Outputs

It is possible through the use of Macro-CAPP to derive auxiliary outputs that could help the user understand the system and the facility better. These outputs are derived from the system's knowledge base through the use of the inference engine. Outputs that can be derived are:

- Facility construction and machine types.
- Specifications of each machine.
- Machining processes possible on each machine.
- Job types that can be processed on any machine.
- Machines on which a particular operation can be performed.
- Material types that can be processed on a particular machine.

The inference mechanism that is used to identify the above mentioned outputs is described in detail in Chapter Five along with sample outputs. These outputs help

the user validate process plan outputs, since it helps derive information that is used in process plan deduction.

7.5 Advantages And Limitations of Macro-CAPP

When the plans, processes, and methods suggested by Macro-CAPP are considered, it's advantages are apparent. This system provides the user with a picture of facility status that other systems do not. It informs the user of the queues in time units at each and every machine. This helps the user get a comprehensive picture of facility load and status. It takes into consideration the queues, congestion, and WIP at each machine modelled in terms of time units. The plan is chosen after evaluating available alternatives. Alternative processes and machines are considered prior to making a decision on which route and sequence of operations can be used. Although minimization of flowtime is the objective function used in this prototype, any desired objective function could be programmed into the dynamic shop status module.

There are a multitude of static systems described in literature. Static systems reviewed in the literature do not consider the dynamic nature of the facility. The plans suggested by them include processes and methods that would be used in the facility irrespective of current facility status. This could result in excessive WIP, flowtimes, and congestion. Macro-CAPP works in conjunction with current facility status. It avoids the drawbacks associated with a static system. Very often a static system that is oblivious of facility status may suggest a route which is heavily congested, has a machine break down, etc. This would reduce the productivity, increase

WIP, etc. This system outputs all the routes and alternatives considered to process a part along with the flowtimes associated for each route. This helps the user understand why a specific route was chosen.

This system is robust in nature. It is a very capable system with general purpose machines that can perform a variety of operations. As mentioned in Chapter Four, the system modelled contains a very wide variety of machines that can perform a very wide variety of machining operations. The machines considered can process both prismatic and rotational parts, perform multiple milling operations, turning, drilling, reaming, boring, etc. Ten different material types are considered. The machining centers considered can process small to large quantities without any appreciable effect on the economics of manufacture. This is because the part setup and change is controlled through software. The machines modelled are representative of state-of-the-art machine tools available.

The software used in this system is a combination of Prolog and FORTRAN. It is modular in nature. It can be easily adapted to any multiple machining center FMS cell. Any change in shop construction can be captured in software with changes to the knowledge base. Minor changes to other software segments may also be needed. These changes can be easily incorporated into software.

Most CAPP systems described in the literature have a very limited manufacturing capability. They either consider a single process such as turning or a single machine such as an engine lathe. They do not consider multiple machines, multiple processes, a wide part family, or a variety of material types as is considered by Macro-CAPP. No CAPP systems mentioned in the literature consider modern machining centers individually or multiple machines working together in a facility. They

do not provide the user with several alternative processes and routes to choose from. They usually consider very small part families.

Mentioned above are the advantages of Macro-CAPP over other existing CAPP systems. It is evident that this system does have a few shortcomings also. Although this system does not provide machining instructions at the micro level, it does provide sufficient information to help with part manufacture. It does not provide tool paths or CNC machining instructions that some other systems provide, but provides the user with job routes, processes, machining parameters, job flowtimes, and possible alternatives. This system does not provide the user with jig or fixture information, detailed machine cutting information, cutting tool specifications, tool holding device information, etc. These are weaknesses of Macro-CAPP.

It is possible to incorporate some of these features in Macro-CAPP, involving additional software effort. For example, detailed cutting parameters, cutting tool specifications, and tool holding device details can be introduced. This would involve substantial software generation. The effort involved would be also be related to facility construction, system considered, etc. It would be very difficult to suggest jigs and fixtures that could be used. Adequate technology for interfacing jig and fixture information with CAPP systems has not been developed and needs to be researched.

Learning, if incorporated into Macro-CAPP, would help the system receive feedback from the facility floor. It would make the system adapt to the actual working of a manufacturing facility. However, the concepts of introducing learning into rule based systems are in their nascent stage. A great deal of research is needed before this can be introduced in practical CAPP systems.

This system uses a custom designed GT code that is used to describe in sufficient detail the part family under consideration. As mentioned earlier, it is a concise and comprehensive code. It would have been ideal however to provide Macro-CAPP with a graphics interface to describe the part to system software. System software should be able to understand the part profile from the graphics input provided. This would help in reducing the time from design to actual manufacture, and help bring about better CAPP CIM Integration. The use of CAD input devices would have definitely enhanced the efficiency and effectiveness of Macro-CAPP.

The information provided by a CAPP system is dependent upon it's software quality, information stored and knowledge base, and inference mechanism. It is as efficient and effective as it is designed to be. Although Macro-CAPP is a prototype system it could be converted into a CAPP system that works on a real time basis in an industrial situation. The skeleton developed and demonstrated in Macro-CAPP can be built upon and adapted to develop CAPP systems that function in industrial FMS cells. The development of Macro-CAPP made it evident that substantial effort is required to design, develop, program, and build a CAPP system. The period of time required to develop a CAPP system is dependent upon the complexity of the system considered and the level of detail required in the plans generated. It is also apparent that the choice of languages and software should depend entirely on the application. No fixed or preconcieved notion should influence the designers thought process in this regard. Process planning systems must be developed in a modular manner to allow for easy modification to the system's capabilities.

7.6 Conclusion

A prismatic and a rotational part were planned for using Macro-CAPP. The purpose of this exercise was to analyze, understand, and validate the working and outputs derived from this prototype system. The plans and outputs derived for both parts considered at each stage in the CAPP procedure are presented. Each part is described, it's input code and auxiliary input presented. The outputs generated at each stage in planning process are presented, discussed, and analyzed. The advantages and weaknesses of Macro-CAPP are discussed.

8.0 CONCLUSION AND FURTHER RESEARCH IDEAS

8.1 Summary of Research

The objective of this research was to design and develop a prototype process planning system for a FMS facility. The pseudo facility modelled consisted of modern machining centers that could perform a variety of machining operations. A link was also constructed between CAPP and other CIM functions such as facility loading, scheduling, route generation and choice. It is possible to further develop this prototype to construct interfaces with other CIM components.

This research highlights the fact that CAPP is not only a CAD-CAM interface, but also needs to interact with other CIM constituents if it is to be used widely in an effective manner. CAPP needs to function in conjunction with other functions such as production planning and control, inventory and procurement, JIT techniques, quality

engineering, cost control and analysis, and industrial automation. A software based CAPP system has been created to illustrate the methods and concepts that can be used as a tool that illustrates how the above mentioned concepts would work. This research also serves as a stepping stone to help in future research and development in CAPP and related areas within CIM.

8.2 Accomplishments Of This Research

A concise, yet comprehensive GT code was designed. This code considers the shape, material, geometry, and dimensions of the part family for which this prototype is developed. This required the design of a code that described the part considered adequately to system software. Description of the part family, part characteristics, materials, GT code specifics, and other relevant details of the part profile input system are presented in Chapters Three and Four. Detailed figures and flowcharts help illustrate the research, design, and development that was done with respect to part profile input in this prototype system.

The CAPP system developed provides manufacturing instructions at the macro-level. This is perhaps a pioneer in the field of CAPP systems developed for multiple machines within a FMS scenario. A very robust FMS is modelled. A very wide variety of possible machining processes are considered. This system moves away from traditional CAPP systems that consider simple, single machines or individual machining processes.

The system developed interfaces a generative CAPP system with the dynamic nature of the shop, and implements it within a CIM framework. Traditional CAPP systems, both variant and generative, have tended to be static in nature. They suggest plans irrespective of current shop status. This has resulted in the development of plans that are not feasible in actual production, and has hampered the applicability of these systems in real life situations. This system considers shop status, the availability of alternate methods to produce a part, and alternate route generation. A heuristic is applied to arrive at the route and the sequence of machining operations subject to hierarchical constraints.

A software system, that uses three languages, has been designed, tested, and implemented in a modular fashion. At each stage of the system's development, the logic and the output generated was checked and verified for possible logical errors. It is possible through changing the knowledge stored in software to change and alter the construction of the pseudo facility modelled. No major software changes would be required if the system construction were to be changed.

At each stage, through adequate examples, the intricacies and details of the system have been described through illustrative figures, flowcharts, and outputs. The logic and reasoning used in each software segment is described through the use of multiple flowcharts and figures. The outputs derived from the software at each stage are presented. Examples of parts and profiles considered, inputs provided, and the outputted process plans are presented.

8.3 Further Research Ideas

Considering the current level of research in CAPP, it is obvious that much needs to be done until widespread CAPP system use becomes possible. Although CAPP is an important CIM constituent, research in CAPP is still in its nascent stage. There are several possible areas of research in CAPP. There are numerous ideas and thoughts that can be researched and developed. Ideas for further, future research are presented below under two heads: ideas that were generated by this research that can be considered as possible off-shoots, and as ideas for research within the overall CAPP spectrum.

8.3.1 Thoughts For Further Study Generated By This Research

Based upon the knowledge gained by this research experience, it obvious that several extensions are possible. It is possible to consider a wider variety of part features in this inputed part profile. This would involve choosing a more complex and complicated part family for CAPP system design. Restrictions placed upon external surface profiles for both rotational and prismatic parts can be removed. It is possible to consider a wider variety of geometrical features.

Part profile input currently is through the use of a GT code. It should be possible to consider interfacing the CAPP system with a CAD system to help define and describe to the system the features of the part being considered. This would help in the process of truly interfacing CAD and CAM through a completely automated CAPP

system. Human interaction and input in this system would be much less than in a CAPP system using GT codes.

The use of a CAD input device would also help in devising CAPP systems that would be automated to the extent of developing a process plan for a part just after the initial part design has been created on a CAD system such as CADAM. This would be invaluable in a CIM atmosphere especially with recent trends to enhance communication between CAD, CAM, and other CIM constituents as enunciated by concurrent engineering principles. However, there are numerous problems in part surface profile description and comprehension that hinders the widespread use of CAD input.

The CAPP system devised gives the user a variety of outputs that would suffice for this application. However, through appropriate software it is possible to device postprocessors that could convert machining instructions to appropriate CNC machining code. This would require for the CAPP system to identify the machine to be used. This could reduce the time lag between the design and actual part manufacture.

It should be possible to interface CAPP systems with graphical simulators to view the movement of products, the occurrence of queues, the build up of WIP, machine utilization, system congestion, etc. This could be perhaps done through the development of interfaces between CAPP systems and languages such as CINEMA. The graphics capabilities of these existing software systems could be integrated with complex CAPP software through software links.

This prototype system uses a heuristic to determine the route and the actual sequence of operations to be used. It should be possible, however, to use more com-

plex heuristics or optimization routines. It should be possible, for example, to develop mixed integer programming formulations to model a facility and optimize the objective function under consideration. The solution of the mixed integer program could be effected through the use of existing, available software. Exact optimal routes could be derived.

It should be possible to devise interfaces between CAPP systems and various CIM functions. This research developed an interface between CAPP, scheduling, FMS, alternate route generation, loading, etc. It should be possible to interface CAPP systems with other functions such as quality control, cost analysis, inventory and procurement, etc. Each one of these would help in the developing of concepts that would increase the feasibility and applicability of CAPP systems. They would help in promoting the widespread use of CAPP within CIM.

CAPP systems that deal with the dynamic situation in a facility seem to be good application areas for the use of parallel processing techniques in software development. This would require designing code that would be processed on parallel processors. The development of these techniques is still in it's nascent stage. They are being developed increasingly for applications that involve extensive computation such as in the solution of large linear programming systems, and in complex mathematical applications. The same techniques can be researched, developed, adapted, and applied to serve computer intensive CAPP systems that currently require extensive software developmental time and computer execution time.

These are some ideas for future research that would follow directly from this research. Each one of these ideas would require a great deal of research and devel-

opment. They would indeed constitute substantial contributions to existing literature and research on CAPP.

8.3.2 Thoughts For Further Research Within CAPP

There are numerous problems to be addressed in CAPP even at the macro-CAPP level. The problems will certainly multiply when studied at the micro-level, when each of the problems are actually solved.

8.3.2.1 Part Profile Input Techniques

Most systems developed to date have cumbersome GT codes that have to be input [67,68,85]. Some systems have developed CAD boundary representation techniques that take the profile drawn on a screen and convert it into data that can be used by the CAPP system to develop a process plan [9,35,40,67,68,85].

The problem with using GT in CAPP is that it requires planners to remember and use complex codes. In designing the code, factors to be considered include the population of the components to be machined, the detail the code must represent, the structure of the code such as a monocode, a hierarchical code, or a hybrid code, and the digital representation. The code needs to be concise since even a 10 digit code has numerous combinations. Personnel using the system on the shopfloor do not appreciate using long, tedious, complicated codes that are error prone. This problem is further accentuated in complex systems that use complex coding techniques. This is a real problem faced in the expansion of CAPP use, especially in batch manufac-

turing scenarios with a variety of parts being manufactured, and a variety of codes being used to describe part surfaces.

Methods that would bring about quick, easy coding of parts without losing any accuracy need to be identified. Any new coding mechanism will need to input all the relevant information. The basic concepts of GT have to be researched, developed, and evolved such that the tediousness of part coding can be reduced.

It would be ideal if the part profile is inputed to the CAPP system using a graphics system and a CAD link [71]. Software could extract relevant parameters from this drawing in a format compatible to the CAPP system. The inputed part profile must be decomposed systematically into 'shape' elements, analyzed, and it's characteristics identified. The CAPP system needs to consider the shape of the part prior to actual processing, and the shape that it must be processed into, and then develop a method to process the part. In the ideal case, the parameters generated by software from the inputed drawing would generate the process plan without any human intervention. This will require extensive software design, especially if a wide variety of parts are to be considered.

If the input to the CAPP system is a two or three dimensional display or a hard-copy prepared by a CAD system or a geometric model in CAD data base, it is not efficient to have a planner interpret this because it is time consuming and error prone. Component models in the CAD system must be postprocessed into a format compatible to a specific CAPP system. This will reduce the time spent, and reduce possible errors. This is a problem critical to future CAPP development that can be addressed today. It is definitely possible that as better CAD systems are designed

and integrated with CAPP systems, they will bring about faster and more accurate inputing of information into the system.

8.3.2.2 Skeleton CAPP Systems To Create CAPP Systems 'Quick and Easy'

A major problem in CAPP system design is that it is always shop specific. A CAPP is currently developed to serve a specific shop, with software being written exclusive for that particular shop or system. This requires that the specific shop's capacity is comprehensively described in the software. Decision logic used to decide which machine is used to process a part has to be designed, specific to that system. This requires even small shops with batch manufacturing process to develop individual CAPP systems to suit their shop. This may not be economically viable.

The software requirements of a CAPP must not be underestimated. The CAPP must comprehensively describe in the knowledge base facts such as the number of machines available, machine capacity with respect to depth of cut, feeds, speeds, maximum forces, maximum and minimum job sizes that can be machined, etc. Software must describe the tools that can be used, their capacities, holding device data, machinability of materials that may be processed, computed machining times, and standard costs.

A skeleton process planning system would provide the basic infrastructure and assist in the quick development of a CAPP system. It would provide the software to link the databases mentioned above. The individual user would then write the software in a manner compatible to the skeleton. Shop specific factors that the system would require would be inputed to the system. This system would reduce the pro-

gramming and developmental effort. The skeleton system could provide the adequate decision structure for the problem.

8.3.2.3 CAPP System Integration With CAD Databases

Ideally, a CAPP system that is implemented in a manufacturing facility with multiple machines would store initial part drawings in a CAD database and use these drawings to generate comprehensive process plans automatically. These outputs can then be used as a feedback to effect design modifications as illustrated by concurrent engineering principles. There have been several attempts to achieve this, but widespread success has not yet been reported [67,68,85,71]. Such a system requires the inputed profile to be stored in the CAD system in a format that is compatible to the CAPP system software. The profile would be retrieved when required to generate a process plan.

The difference between this system and systems that produce plans after inputing a part profile is that this system requires part drawings to be stored in the system for possible future use. It would also permit remote retrieval of part drawings to generate process plans. This could be best described as a scenario in which the designer on completion of a part design stores it in a CAD database. This drawing would be retrieved using software by the manufacturing facility which would produce a comprehensive plan through generative process planning. Such a system would interface CAD and CAM, reducing human interference. The problems in developing this system include software generation, development of CAPP systems that take into consideration all the machines of the shopfloor, part profile recognition techniques, and generation of appropriate machining instructions. Although it seems that such a

system should be possible, there are tremendous problems that need to be overcome [12].

8.3.2.4 Profile Transformation

Profile transformation is a critical problem that involves the transformation and assimilation of part profiles and surface shapes by software from input which could be in the form of a GT code or a CAD drawing. Current research has only dealt with profile transformation of turned surfaces, and holes [67,68].

Research is needed in the conversion/decomposition module that will break up an inputed drawing into elements for surfaces such as threaded surfaces, complex contours, ellipses, tapers, chamfers, fillets, punched holes, etc. These have to be represented in a format that can be recognized by CAPP systems. The fact that a wide variety of surfaces cannot be converted into a format that can be used by the CAPP system software is a problem that reduces the effectiveness of a CAPP system, and reduces the spectrum for which it can be applied. It is important that the problem of effectively transforming input parameters that can recognize profiles such as curves, contours, helixes, etc be addressed. It is necessary that parameters for any surface can be transformed into a format that the CAPP system software is able to understand.

8.3.2.5 Downloading Instructions From CAPP to Machines On Shop Floor

There are prototype CAPP systems under development that attempt to download machining instructions directly from the system to machines on the floor. It would be

ideal that once the CAPP system develops a comprehensive process plan, machining instructions are directly produced. This then needs to be converted into detailed manufacturing instructions, in a format that can be directly loaded into the controller of a numerically controlled machine. This would help in the development of a totally automated factory, and help close the link between CAD and CAM.

Downloading instructions will require that the machine on which the job is to be manufactured is decided upon through software. Using the speed, feed, depth of cut instructions that the system developed, the detailed program for the CNC machine has to be developed. This can then be loaded to the controller if it is known which machine will be used to perform a specific process. Extensive software development will be required since the actual process plan has to be developed, the shop load checked, its dynamic condition reviewed, and the machine that should perform a particular process identified. When this is done, the CNC instructions as developed from the CAPP system instructions must be downloaded into the controller of the relevant machine.

8.3.2.6 CAPP In Cost Estimation

If CAPP systems are to be used to help prepare bids for customers, they must be equipped with cost estimating routines that will take into consideration processing time and other factors to help determine costs that will be incurred in processing. This is an area that has not been researched in depth. This area will help management decide upon the correct price that they should quote in a bid taking into consideration factors such as shop capacity, expected shop load, possible manufacturing

routes, etc. CAPP can also be used to generate manufacturing costs that will help industry make decisions such as the make-buy decision.

The problems associated with cost estimating is that a CAPP system has to be custom-made for the facility. Software has to be built into the system to take into consideration factors such as machining time, tool cost, jigs and fixtures used and their cost, etc. It will be easier for the system to compute machining times and their costs and use this to develop an estimate of the actual cost. An application such as this would make a CAPP system more cost effective and attractive to a potential user.

8.3.2.7 Application Of CAPP In A Wider Industry Spectrum

A review of the literature indicates that the areas of CAPP use are very restricted. It is evident that the techniques to solve the problem in most situations are currently known. When new applications are sought, it is almost certain that new problems will arise that will result in the development of new techniques and methods for CAPP application.

The following areas are systems in which CAPP use can be introduced, or expanded from current levels:

- In continuous process industries.
- In metal cutting.
- In joining processes.
- Jig/fixture design.
- Non-traditional machining processes.
- Planning for assembly.
- Metal fabricating.

- Wood working.

All this indicates that the current areas of possible application of CAPP are just the tip of the iceberg. There could be many more possible application areas of CAPP. As each new application area is researched, many more new concepts, techniques, and methodologies will be added.

Bibliography

1. Allen, D.K., 'Generative Process Planning Using The DCLASS Information System', Monograph #4, Computer Aided Manufacturing Laboratory, Brigham Young University, Provo, Utah, 1979.

2. Allen, D.K. & Smith, P.R., 'Computer Aided Process Planning', Computer Aided Manufacturing Laboratory, Brigham Young University, Provo, Utah, 1980.

3. Berenji, H.R., & Khoshnevis, B., 'Use Of Artificial Intelligence In Automated Process Planning', Computers In Mechanical Engineering, September, 1986, pp.47-55.

4. Berra, P.B., 'Investigation Of Automated Planning And Optimaization Of Metal Working Processes', Doctoral Dissertation, Purdue University, West Lafayette, Indiana, 1968.

5. Caldeira, E., 'Integration Of CAPP And Computerized Standard Data', Proceedings - Fall IIE Conference, December, 1985, pp.106-109.

6. Chang, T.C., 'Interfacing CAD And CAM - A Study In Hole Design', Masters Thesis, Virginia Tech, Blacksburg, Virginia, 1980.

7. Chang,T.C., Wysk, R.A., & Davis, R.P., 'Interfacing CAD And CAM - A Study In Hole Design', Computing And Industrial Engineering, Vol.6, No.2, 1982, pp.91-102.

8. Chang,T.C., & Wysk, R.A., 'An Integrated CAD/Automated Process Planning System', AIIE Transactions, Vol.13, No.3, 1981, pp.223-233.

9. Chang, T.C., ' TIPPS - A Totally Integrated Process Planning System', Doctoral Dissertation, Virginia Tech, Blacksburg, Virginia, 1982.

10. Chang, T.C., 'Computer Aided Process Planning Today And In The Future', Proceedings Of The Fall IIE Conference, November, 1983, pp.349-355.

11. Chang, T.C., & Wysk, R.A., 'Integrating CAD And CAM Through Automated Process Planning', International Journal Of Production Research, Vol.22, No.5, 1984, pp.877-894.

12. Chang, T.C., & Wysk, R.A., 'An Introduction To Automated Process Planning Systems', Prentice Hall, New Jersey, 1985.

13. Chryssolouris, G., & Gruenig, I., 'Process Planning Interface For Intelligent Manufacturing Systems', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.181-188.

14. Davies, J.B., & Darbyshire, I.L., 'The Use Of Expert Systems In Process Planning', Annals Of The CIRP, Vol.33, No.1, 1984.

15. Descotte, Y. & Latombe, J.C.,'GARI: A Problem Solver That Plans How To Machine Mechanical Parts', Proceedings - IJCAI 7, Vancouver, Canada, pp.766-772, August, 1981.

16. El-Midany, T.T., & Davies, B.J., 'AUTOCAP - A Dialog System For Planning The Sequence Of Operations For Turning Components', International Journal Of Machine Tool Research, Vol.21, No.3/4, 1981, pp.175-191.

17. Emerson, C., & Ham, I., 'An Automated Coding And Process Planning System Using A DEC PDP-10', Computing And Industrial Engineering, Vol.6, No.2, 1982, pp.159-168.

18. Eshel, G., & Barash, M., & Chang, T.C., 'A Rule Based System For Automatic Generation Of Process Outlines For Deep Drawing Processes', Proceedings - ASME Winter Annual Meeting, Miami Beach, Florida, November, 1985, pp.1-18.

19. Eskicioglu, H. & Davies, B.J., 'An Interactive Process Planning System For Prismatic Parts', International Journal Of Machine Tool Research, Vol.21, No.3/4, 1981, pp.193-206.

20. Eversheim, W., & Diels, A., ' Changing Requirements For CAP Systems Lead To A New CAP-Data Model ', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.9-15.

21. Ferreira, P., Puls, F., Chang, T.C., & Liu, C.R., 'A CAD/CAM System For Complex Surfaces', Automated Process Planning Systems, ASME Winter Annual Meeting, Miami Beach, Florida, November, 1985, pp.189-199.

22. Halevi, G., & Weill, R., 'An Algorithm To Determine Number Of Cuts In Machining Operations', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.63-70.

23. Ham, I., 'Group Technology', Handbook Of Industrial Engineering, Edited By G.Salvendy, John Wiley & Sons, New York, 1982.

24. Hannam, R.G., & Plummer, J.C.S., 'Capturing Production Engineering Practice Within A CADCAM System', International Journal Of Production Research, Vol.22, No.2, 1984, pp.267-280.

25. Hegland, D.E., 'Out In Front With CAD/CAM at Lockheed-Georgia', Production Engineering, Vol.28, No.11, 1981, pp.45-48.

26. Houtzel, A., 'Integrating CAD/CAM Through Group Technology', Proceedings Of The Annual Meeting And Technical Conference Of The Numerical Control Society, Dallas, Texas, May, 1981, pp.430-444.

27. Hsu-Pin Wang & Wysk, R.A., 'Microcomputer Based Process Planning Systems', Proceedings - Fall IIE Conference, December 1985, pp.145-154.

28. Hummel, K.E., 'An Expert Machine Tool Planner', Proceedings - ASME International Computers In Engineering Conference, New York City, New York, Vol.2, 1985, pp.367-373.

29. Inui, M., Suzuki, H., Kimura, F., & Sata, T., 'Extending Process Planning Capabilities With Dynamic Manipulation Of Product Models', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.273-280.

30. Iwata, K., & Sugimara, N., 'A Knowledge Based Computer Aided Process Planning System For Machine Parts', Proceedings - 16th Annual Seminar On Manufacturing Systems, Vol.14, No.2, Tokyo, 1984.

31. Iwata, K., & Fukuda, Y., 'KAPPS: Know-How and Knowledge Assisted Production Planning System In The Machining Shop', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.287-294.

32. Jiang, W., Xu, H., 'CAPP Systems And Applications In China', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.99-104.

33. Joshi, S., Chang, T.C., & Liu, C.R., 'Process Planning Formalization In An AI Framework', International Journal Of AI In Engineering, Vol.1, No.1, 1986, pp.45-53.

34. Joshi, S., & Chang, T.C., 'Feasible Tool Approach Directions For Machining Holes In Automated Press Planning Systems', Proceedings - Symposium On Integrated and Intelligent Manufacturing, ASME Winter Annual Meeting, Anaheim, California, December, 1986, pp.157-169.

35. Joshi, S., & Chang, T.C., 'CAD Interface For Automated Process Planning', Proceedings - 19th .us CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.39-45.

36. Joshi, S., Vissa, N., & Chang, T.C., 'Expert Process Planning System With Solid Model Interface', International Journal Of Production Research, Vol.26, No.5, 1988, pp.863-885.

37. Kimemia, J., & Gershwin, S.B., 'Flow Optimization In Flexible Manufacturing Systems', International Journal Of Production Research, Vol. 23, No.1, 1985, pp.81-96.

38. Li, J., Han, C., & Ham, I., 'CORE-CAPP - A Company Oriented Semi-Generative Computer Aided Process Planning System', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.219-225.

39. Link, C.H., 'CAPP - CAM-I Automated Process Planning System', Proceedings Of The Annual Meeting And Technical Conference Of The Numerical Control Society, Cincinnati, Ohio, 1976, pp.401-408.

40. Matsushima, K., Onada, N., & Sata, T., 'The Integration Of CAD And CAM By The Application Of Artificial Intelligence Techniques', Annals Of The CIRP, Vol.34, No.1, 1985, pp.329-332.

41. Meenakshi Sundaram, R., & Ta Jen Cheng, 'Microcomputer Based Process Planning Using Geometric Programming', International Journal Of Production Research, Vol.24, No.1, 1986, pp.119-127.

42. Milner, D.A., 'The Use Of Decision Table Logic In a Manufacturing System', International Journal Of Production Research, Vol.15, No.1, 1977, pp.17-26.

43. Milacic, V.R., Urosevic, M., Veljovic, A., Miler, A., & Race, I., 'SAPT - Expert System Based On Hybrid Concept Of Group Technology', Proceedings - 19th.us CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.189-195.

44. Nau, D.S., & Chang, T.C., 'Prospects For Process Selection Using Artificial Intelligence', Computers In Industry, Vol.4, No.3, 1983, pp.253-263.

45. Nau, D.S., & Chang, T.C., 'A Knowledge-Based Approach To Generative Process Planning', Automated Process Planning Systems, Proceedings - ASME Winter Annual Meeting, Miami Beach, Florida, November, 1985, pp.65-72.

46. Nau, D.S., & Gray, M., 'SIPS: An Application Of Hierarchical Knowledge Clustering To Process Planning', Integrated And Intelligent Manufacturing, Edited By Liu, C.R., & Chang, T.C., ASME, 1986, pp.219-226.

47. Nau, D.S., & Luce, M., 'Knowledge Representation And Reasoning Technique For Process Planning', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.91-98.

48. Nof, S.Y., & Barash, M.M., 'Dynamic Process Selection Procedures And Their Effect On Machine Configuration', International Journal Of Machine Tool Research, Vol.20, pp.137-146.

49. Ohashi, K., & Hitomi, K., 'A New Approach To Process Planning And Scheduling For Flexible Machining Cells', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.169-174.

50. Opitz, H., 'A Classification Scheme To Describe Workpieces', Pergamon Press, Elmsford, New York, 1970.

51. Opitz, H., & Weindahl, H.P., 'Group Technology And Manufacturing Systems For Small And Medium Quantity Production', International Journal Of Production Research, Vol.9, No.1, 1971, pp.181-203.

52. Park, M.W., & Davies, B.J., 'Integration Of Process Planning Into CAD Using IGES', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.315-319.

53. Phillips, R.H., & ElGomayel, J., 'Group Technology Applied To Product Design', MAPEC Module, Purdue University, West Lafayette, Indiana, August 1977.

54. Phillips, R.H., 'A Computerized Process Planning System Based Upon Component Classification & Coding', Doctoral Dissertation, Purdue University, West Lafayette, Indiana, 1978.

55. Phillips, R.H., & Mouleeswaran, C.B., 'A Knowledge Based Approach To Generative Process Planning', Proceedings - AUTOFACT 1985, Detroit, June, 1985, pp.10-1 - 10-15.

56. Phillips, R., Arunthavanathan, V., Doug Zhou, X., 'An Intelligent Design and Process Planning System', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.17-22.

57. Rasch, F.O., 'IPROS - A Variant Process Planning System', Proceedings -19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.157-160.

58. Rebholz, M., 'Computer Aided Planning In Cold Forging', CIRP Annals, Vol.29, No.1, 1980, pp.173-177.

59. Rolstadas, A., 'Flexible Design Of Production Planning Systems', International Journal of Production Research, Vol.26, No.3, 1988, pp.507-520.

60. Santochi, M., & Guisti, F., 'Proposal of a Product Modeller Oriented To Manufacturing Problems', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.27-31.

61. Sakomoto, C., Kado, S., Miyamatsu, M., Miyamoto, Y., & Kojima, N., 'POPULAR - An Automatic Process Planning System', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.105-110.

62. Schaffer, G., 'GT Via Automated Process Planning', American Machinist, May, 1980, pp.119-122.

63. Schaffer, G., 'Implementing CIM', American Machinist, Special Report No. 736, August, 1981, pp.151-174.

64. Shankar, K., & Tzen, Y.J., 'A Loading And Dispatching Problem In A Random Flexible Manufacturing System', International Journal Of Production Research, Vol.23, No.3, 1985, pp.579-595.

65. Smith, R.M., 'Computer Aided Fully Generative Process Planning', Manufacturing Engineering, May, 1981, pp.98-99.

66. Spadoni, M., & Mutel, B., 'A New Approach To Automatic Generation Of Process Planning', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.121-125.

67. Srihari, K., & Greene, T.J., 'An Evaluation And Review Of Computer Aided Process Planning Systems', Proceedings - World Productivity Forum & 1987 International Industrial Engineering Conference, Washington D.C., May, 1987, pp.438-444.

68. Steudel, H.J., 'Computer Aided Process Planning: Past, Present, and Future', International Journal Of Production Research, Vol.22, No.2, 1984, pp.253-266.

69. Subramanyam, S., Lu, C-Y.S., Zdeblick, W., 'A Characterization Of The Process Planning Task From An Artificial Intelligence Perspective', Proceedings - 19th .us CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.197-206.

70. Tonshoff, H.K., Beckendorff, U., & Schaele, M., 'Some Approaches To Represent The Interdependence Of Process Planning And Process Control', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.257-271.

71. Tipnis, V.A., 'Computer Aided Process Planning', A Critique Of Research And Implementation', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.295-300.

72. Tirupatikumara, S., Joshi, S., Moodie, C.L., Kashyap, R.L., & Chang, T.C., 'Expert Systems In Industrial Engineering', International Journal Of Production Research, Vol.24, No.5, 1986, pp.1107-1125.

73. Tsang, J., 'The Propel Process Planner', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.71-77.

74. Tulkoff, J., 'Lockheed's GENPLAN', Proceedings Of The Annual Meeting And Technical Conference Of The Numerical Control Society, Dallas, Texas, May, 1981, pp.417-421.

75. Van Houten, F.J.A.M., & Kals, H.J.J., 'ROUND: A Flexible Technology Based Process And Operations Planning For NC Lathes', Proceedings - CIRP 16th Annual Seminar On Manufacturing Systems, Vol.14, No.1, Tokyo, 1984, pp.168-174.

76. Van Houten, F.J.A.M., & Tiemersma, J.J., 'An Adaptive Control Module For ROUND', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.233-240.

77. Van't Erve, A.H., Kals, H.J.J., 'XPLANE: A Knowledge Base Driven Process Planning Expert System', Annals Of The CIRP, Vol.35, No.1, 1986, pp.325-329.

78. Van't Erve, A.H., & Kals, H.J.J., 'The Selection Of Optimum Machining Operations In Automated Process Planning', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.47-54.

79. Vogel, S.A., & Adlard, E.J., 'The Autoplan Process Planning System', Proceedings Of The Annual Meeting And Technical Conference Of The Numerical Control Society, 1981, pp.422-429.

80. Wang, H.P., & Wysk, R.A., ' Micro-GEPPS: A Microcomputer-Based Process Planning System', Computer Aided Intelligent Process Planning, ASME, PED-Vol.19, 1985, pp.139-150.

81. Wang, H.P., & Wysk, R.A., 'Intelligent Reasoning For Process Planning', Journal Of Manufacturing Systems, Vol.5, No.2, 1986, pp.103-111.

82. Wang, W., 'Application Of Solids Modelling To Automate Machining Parameters For Complex Parts', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.33-37.

83. Wang, H., & Wysk, R., 'Turbo-CAPP: A Knowledge Based Computer Aided Process Planning System', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.161-167.

84. Weber, I.H., 'Development Of Technical Expert Systems By Means Of Prolog', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.161-167.

85. Weill, R., Spur, G., & Eversheim, W., 'Survey Of Computer Aided Process Planning Systems', CIRP Annals, Vol.31, No.2, 1982, pp.539-551.

86. Wilhelm, W.E., & Hyun-Myung Shin, 'Effectiveness Of Alternate Operations In A Flexible Manufacturing System, International Journal Of Production Research, Vol.23, No.1,1985, pp.65-79.

87. Wolfe, P.M., 'Automatic Process Planning Using Artificial Intelligence', Proceedings Of The Annual IIE Conference, May, 1984, pp.387-395.

88. Wright, A.J., Darbyshire, I.L., Park, M.W., Davies, B.J., 'EXCAPP and ICAPP:Integrated Knowledge Based Systems For Process Planning Components', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.309-313.

89. Wysk, R.A., Barash, N.M., & Moodie, C.M., 'Unit Machining Operations: An Automated Process Planning And Selection Program', Journal Of Engineering For Industry, Vol.102, November, 1980, pp.297-302.

90. Wysk, R.A., 'An Automated Process Planning And Selection Program: APPAS', Doctoral Dissertation, Purdue University, West Lafayette, Indiana, 1977.

91. Yao, D.D., 'Material And Information Flows In Flexible Manufacturing Systems', Material Flow, Vol.2, No.2, 1985, pp.143-149.

92. Zdeblick, W., 'Process Planning Evolution: The Impact Of Artificial Intelligence', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.175-179.

93. Zhang, S., Gao, W.D., 'TOJICAP - A System Of Computer Aided Process Planning System', CIRP Annals, Vol.33, No.1, 1984, pp.299-301.

94. Zuckerman, M.I., 'A Knowledge Base Development For Producibility Analysis In Mechanical Design', Proceedings - 19th CIRP International Seminar On Manufacturing Systems, State College, Pennsylvania, June, 1987, pp.301-307.

Appendix A. CAPP INFERENCE SOFTWARE IN PROLOG

```
/*  MACRO-CAPP IS A COMPUTER AIDED PROCESS  PLANNING  */
/*  CODE DESIGNED TO HELP IN DEVELOPING COMPUTER      */
/*  PLANS FOR A VARIETY OF PARTS TO BE MANUFACTURED   */
/*  IN A PSEUDO FACILITY. IT IS USER FRIENDLY AND     */
/*  INTERCATIVE IN NATURE.                            */

      code = 1920

DOMAINS

      Comp_Class = Integer
      Ext_Shape = Integer
      Int_Shape = Integer
      Plane_Surface_Mach = Integer
      Aux_Holes = Integer
      Dimension = Integer
      Tolerance = Integer
      Surface_Finish = Integer
      Mach_Identifier = Integer
      N = Integer
      file = myfile
      mach_id = Symbol
      operation = Symbol
      Init_Form = Symbol
      Material_Type = Symbol
      maxheight = Symbol
      maxdia = Symbol
      mindia = Symbol
      material = Symbol
      maxlength = Symbol
      minspeed = Symbol
      maxspeed = Symbol
      speedincr = Symbol
      type = Symbol
      name = Symbol
      width = Symbol
      jtype = Symbol
      jobdes = Symbol
      number = Real
      maxturrets = Symbol
      maxhp = Symbol
      maxtools = Symbol
      maxatc = Symbol
```

Predicates

```
delay (integer)
read_comp
readaux
run
mach_operations
go
start
surface_finish (integer)
mach_cap (mach_id,jtype)
mach_use(mach_id, name)
machtype (mach_id, name)
machine (mach_id, operation)
mach (maxdia, number)
mach (maxlength, number)
mach (minspeed, number)
mach (minspeed, number)
mach (maxhp, number)
mach (maxtools, number)
mach (maxturrets, number)
mach (speedincr, number)
mach_material (material, type)
jtype (jobdes)
thread (symbol)
outer (symbol)
internal (symbol)
holes (symbol)
plane_surface (symbol)
material (symbol)
operations_reqd (symbol, symbol)
turning (symbol)
facing (symbol)
drilling (symbol)
boring (symbol)
reaming (symbol)
knurling (symbol)
external_threading (symbol)
internal_threading (symbol)
taperturning (symbol)
partoff (symbol)
side_mill (symbol)
face_mill (symbol)
plain_mill (symbol)
dovetail (symbol)
slitting (symbol)
end_mill (symbol)
shell_mill (symbol)
pocket_mill (symbol)
write_file_machcap1
write_file_machcap2
write_file_machcap3
write_file_machcap4
write_file_machcap5
write_file_machmat1
write_file_machmat2
write_file_machmat3
write_file_machmat4
write_file_machmat5
write_file_jtype_mach1
write_file_jtype_mach2
write_file_jtype_mach3
write_file_jtype_mach4
```

```
write_file_jtype_mach5
write_file_outer
write_file_inner
write_file_external_plane
write_file_holes
write_file_turning
write_file_facing
write_file_drilling
write_file_boring
write_file_reaming
write_file_knurling
write_file_external_threading
write_file_internal_threading
write_file_taperturning
write_file_partoff
write_file_side_mill
write_file_face_mill
write_file_plain_mill
write_file_dovetail
write_file_slitting
write_file_end_mill
write_file_shell_mill
write_file_pocket_mill
rotational_short
rotational
rotational_long
cubical_flat
cubical
cubical_long
outer_smooth
outer_thread
outer_groove
outer_slot
outer_stepped_one_side
outer_stepped_both_sides
/* outer_smooth_chamfers
*/ outer_no_plane_machining
outer_plane_machining
outer_plane_machining_chamfers
internal_none
internal_smooth
internal_stepped_one_side
internal_stepped_both_side
internal_slot
internal_thread
internal_groove
internal_two_principal_bores
internal_multiple_parallel_bores
internal_non_parallel_bores
surface_plane_none
surface_plane_external_single_curve
surface_plane_external_multi_curve
surface_plane_external_keyway
surface_plane_external_groove
surface_plane_slot
holes_none
holes_axial_related
holes_axial_unrelated
holes_radial_related
holes_radial_unrelated
holes_multidirection
holes_related_multidirection
material_copper
```

```
        material_brass
        material_aluminum
        material_steel_LT3Carbon
        material_steel_LT6Carbon
        material_steel_LT1Carbon
        material_iron_soft
        material_iron_medium
        material_iron_hard
        material_stainless_steel
        cubic_surface_finish_32
        cubic_surface_finish_64
        cubic_surface_finish_128
        rot_surface_finish_32
        rot_surface_finish_64
        rot_surface_finish_128

goal

    internal_stepped_both_side.

Clauses

/*    THIS SEGMENT CONTAINS SIMPLE CLAUSES THAT   */
/*    STORE FACILITY CAPACITY AND CAPABILITY.     */

/*    CLAUSES THAT CONTAIN JOB TYPES THAT CAN BE  */
/*    PROCESSED ON A SPECIFIC MACHINE             */

        mach_cap (mach1, rot).
        mach_cap (mach2, rot).
        mach_cap (mach3, cubic).
        mach_cap (mach3, rot).
        mach_cap (mach4, cubic).
        mach_cap (mach4, rot).
        mach_cap (mach5, rot).

/*    TYPES OF JOBS - OUTER DESCRIPTION OF JOB    */
/*    TYPES THAT CAN BE CONSIDERED                */

        jtype (rot).
        jtype (cubic).
        outer (smooth).
        outer (thread).
        outer (groove).
        outer (slot).
        outer (stepped_one_side).
        outer (stepped_both_sides).
        outer (no_plane_mach).
        outer (plane_mach).
        outer (plane_mach_chamfers).

/*  INTERNAL SURFACE DESCRIPTION FOR PRISMATIC &  */
/*  ROTATIONAL JOBS THAT CAN BE MACHINED          */

        internal (none).
        internal (smooth).
```

```
internal (stepped_one_side).
internal (stepped_both_side).
internal (slot).
internal (thread).
internal (groove).
internal (two_principal_bores).
internal (multiple_parallel_bores).
internal (non_parallel_bores).

/* EXTERNAL PLANE SURFACE MACHINABLE FOR */
/* BOTH CUBIC AND ROTATIONAL JOBS.      */

plane_surface (none).
plane_surface (external_single_curve).
plane_surface (external_multi_curve).
plane_surface (keyway).
plane_surface (groove).
plane_surface (slot).

/* TYPES OF HOLES MACHINABLE FOR   */
/* BOTH CUBIC AND ROTATIONAL JOBS. */

holes (none).
holes (axial_related).
holes (axial_unrelated).
holes (radial_related).
holes (radial_unrelated).
holes (multidirection).
holes (related_multidirection).

/*  THIS SEGMENT STORES THE MATERIAL TYPES */
/*  THAT CAN BE MACHINED IN THIS FACILITY  */

material (copper).
material (brass).
material (aluminum).
material (steel_LT3_Carbon).
material (steel_LT6_Carbon).
material (steel_LT1_Carbon).
material (stainless_steel_).
material (iron_soft).
material (iron_medium).
material (iron_hard).

/*  THIS SEGMENT STORES THE SURFACE FINISH */
/*  THAT IS CONSIDERED IN THIS FACILITY    */

surface_finish (32).
surface_finish (64).
surface_finish (128).
thread (yes).

/*  OPERATIONS REQUIRED TO MACHINE SURFACES     */
/*  CONSIDERING THE SECOND DIGIT OF INPUT CODE  */
```

```
operations_reqd (turning, smooth).
operations_reqd (external_threading, thread).
operations_reqd (tapping, thread).
operations_reqd (die_threads, thread).
operations_reqd (turning, groove).
operations_reqd (slot_mill, slot).
operations_reqd (end_mill, slot).
operations_reqd (face_mill, flat).
operations_reqd (side_mill, flat).
operations_reqd (plain_mill, flat).
operations_reqd (side_mill, cubic).
operations_reqd (face_mill, cubic).
operations_reqd (plain_mill, cubic).
operations_reqd (face_mill, long).
operations_reqd (side_mill, long).
operations_reqd (plain_mill, long).
operations_reqd (face_mill, chamfers).
operations_reqd (side_mill, chamfers).
operations_reqd (plain_mill, chamfers).
operations_reqd (turning, stepped_one_side).
operations_reqd (turning, stepped_both_sides).
operations_reqd (no_machining, no_plane_mach).
operations_reqd (side_mill, plane_mach).
operations_reqd (face_mill, plane_mach).
operations_reqd (plain_mill, plane_mach).
operations_reqd (face_mill, plane_mach_chamfers).
operations_reqd (side_mill, plane_mach_chamfers).
operations_reqd (plain_mill, plane_mach_chamfers).

/*   MACHINING OPERATIONS REQUIRED TO MACHINE  */
/*   SURFACES CONSIDERING THE THIRD DIGIT      */

operations_reqd (no_machining, int_none).
operations_reqd (drill_and_ream, int_smooth).
operations_reqd (drill_and_ream, int_stepped_one_side).
operations_reqd (drill_and_ream, int_stepped_both_sides).
operations_reqd (drill_and_bore, int_slot).
operations_reqd (drill_and_tap, int_thread).
operations_reqd (drill_and_bore, int_groove).
operations_reqd (multiple_drill_and_ream, int_two_principal_bores).
operations_reqd (multiple_drill_and_ream, int_multiple_parallel_bores).
operations_reqd (gang_drill_and_ream, int_non_parallel_bores).

/*   MACHINING OPERATIONS REQUIRED TO MACHINE    */
/*   SURFACES CONSIDERING THE FOURTH DIGIT.      */

operations_reqd (no_machining, plane_none).
operations_reqd (plain_mill, plane_external_single_curve).
operations_reqd (face_mill, plane_external_single_curve).
operations_reqd (side_mill, plane_external_single_curve).
operations_reqd (plain_mill, plane_external_multi_curve).
operations_reqd (face_mill, plane_external_multi_curve).
operations_reqd (side_mill, plane_external_multi_curve).
operations_reqd (end_mill, plane_keyway).
operations_reqd (pocket_mill, plane_keyway).
operations_reqd (end_mill, plane_surface_groove).
operations_reqd (pocket_mill, plane_surface_groove).
operations_reqd (slitting_mill, plane_surface_groove).
```

```
operations_reqd (shell_mill, plane_surface_groove).
operations_reqd (pocket_mill, plane_surface_slot).
operations_reqd (slitting_mill, plane_surface_slot).
operations_reqd (shell_mill, plane_surface_slot).


/*   MACHINING OPERATIONS REQUIRED TO MACHINE     */
/*   SURFACES CONSIDERING FIFTH DIGIT.            */

operations_reqd (no_machining, holes_none).
operations_reqd (multiple_drilling_and_reaming, holes_axial_related).
operations_reqd (gang_drilling_and_reaming, holes_axial_unrelated).
operations_reqd (multiple_drilling_and_reaming, holes_radial_related).
operations_reqd (gang_drilling_and_reaming, holes_radial_unrelated).
operations_reqd (gang_drilling_and_reaming, holes_multidirection).
operations_reqd (multiple_drilling_and_reaming, holes_related_multidirection).


/*   MACHINE TYPE IS ASSOCIATED WITH AN   */
/*   IDENTIFIER AND IS STORED             */


mach_use (mach1, turning_center).
mach_use (mach2, turning_center).
mach_use (mach3, horiz_spindle_mach_center).
mach_use (mach4, vert_spindle_mach_center).
mach_use (mach5, turning_center).

machtype (mach1, turning_center).
machtype (mach2, turning_center).
machtype (mach3, horiz_spindle_mach_center).
machtype (mach4, vert_spindle_mach_center).
machtype (mach5, turning_center).


/*   THE PROCESSES THAT CAN BE EXECUTED ON EACH          */
/*   MACHINE IN THE FACILITY IS STORED AS SIMPLE CLAUSES  */


machine (mach1, turning).
machine (mach1, facing).
machine (mach1, drilling).
machine (mach1, boring).
machine (mach1, reaming).
machine (mach1, knurling).
machine (mach1, external_threading).
machine (mach1, internal_threading).
machine (mach1, taperturning).
machine (mach1, partoff).


machine (mach2, turning).
machine (mach2, facing).
machine (mach2, drilling).
machine (mach2, boring).
machine (mach2, reaming).
machine (mach2, knurling).
machine (mach2, external_threading).
machine (mach2, internal_threading).
machine (mach2, taperturning).
machine (mach2, partoff).
```

```
machine (mach3, drilling).
machine (mach3, tapping).
machine (mach3, reaming).
machine (mach3, boring).
machine (mach3, internal_threading).
machine (mach3, side_mill).
machine (mach3, face_mill).


machine (mach2, plain_mill).
machine (mach2, dovetail_mill).
machine (mach2, slitting).
machine (mach3, end_mill).
machine (mach3, shell_mill).
machine (mach3, pocket_mill).


machine (mach4, drilling).
machine (mach4, tapping).
machine (mach4, reaming).
machine (mach4, boring).
machine (mach4, internal_threading).
machine (mach4, side_mill).
machine (mach4, face_mill).
machine (mach4, plain_mill).
machine (mach4, dovetail_mill).
machine (mach4, slitting).
machine (mach4, end_mill).
machine (mach4, shell_mill).
machine (mach4, pocket_mill).


machine (mach5, turning).
machine (mach5, facing).
machine (mach5, drilling).
machine (mach5, boring).
machine (mach5, reaming).
machine (mach5, knurling).
machine (mach5, external_threading).
machine (mach5, internal_threading).
machine (mach5, taperturning).
machine (mach5, partoff).

/*    SPECIFICATIONS AND CAPACITY OF EACH   */
/*    EACH MACHINE IN FACILITY              */


mach (maxdia1, 8.0).
mach (maxlength1, 24.0).
mach (minspeed1, 40.0).
mach (maxspeed1, 4000.0).
mach (speedincr1, 1.0).
mach (maxhp1, 30.0).
mach (maxturrets1, 2).
mach (maxtools1, 16).


mach (maxdia2, 8.0).
mach (maxlength2, 20.0).
mach (minspeed2, 32.0).
mach (maxspeed2, 4000.0).
mach (speedincr2, 1.0).
mach (maxhp2, 20.0).
```

```
mach (maxturrets2, 2).
mach (maxtools2, 12).


mach (maxwidth3, 26.0).
mach (maxheight3, 26.0).
mach (maxlength3, 32.0).
mach (minspeed3, 32.0).
mach (maxspeed3, 4000.0).
mach (speedincr3, 1.0).
mach (maxhp3, 20.0).
mach (maxtools3, 4).
mach (maxatc3, 45).


mach (maxwidth4, 40.0).
mach (maxheight4, 40.0).
mach (maxlength4, 40.0).
mach (minspeed4, 40.0).
mach (maxspeed4, 4000.0).
mach (speedincr4, 1.0).
mach (maxhp4, 20.0).
mach (maxtools4, 4).
mach (maxatc4, 45).


mach (maxdia5, 12.0).
mach (maxlength5, 20.0).
mach (minspeed5, 32.0).
mach (maxspeed5, 3000.0).
mach (speedincr5, 1.0).
mach (maxhp5, 30.0).
mach (maxturrets5, 1).
mach (maxtools5, 12).

/*    MATERIALS THAT CAN BE PROCESSED          */
/*    ON EACH MACHINE CONSIDERED IN THE FACILITY */


mach_material (mach1, copper).
mach_material (mach1, brass).
mach_material (mach1, aluminum).
mach_material (mach1, steel_LT3Carbon).
mach_material (mach1, steel_LT6Carbon).
mach_material (mach1, steel_LT1Carbon).
mach_material (mach1, iron_soft).
mach_material (mach1, iron_medium).
mach_material (mach1, iron_hard).
mach_material (mach1, stainless_steel).


mach_material (mach2, copper).
mach_material (mach2, brass).
mach_material (mach2, aluminum).
mach_material (mach2, steel_LT3Carbon).
mach_material (mach2, steel_LT6Carbon).
mach_material (mach2, steel_LT1Carbon).
mach_material (mach2, iron_soft).
mach_material (mach2, iron_medium).
mach_material (mach2, iron_hard).
mach_material (mach2, stainless_steel).
```

```
mach_material (mach3, copper).
mach_material (mach3, brass).
mach_material (mach3, aluminum).
mach_material (mach3, steel_LT3Carbon).
mach_material (mach3, steel_LT6Carbon).
mach_material (mach3, steel_LT1Carbon).
mach_material (mach3, iron_soft).
mach_material (mach3, iron_hard).
mach_material (mach3, iron_hard).
mach_material (mach3, stainless_steel).


mach_material (mach4, copper).
mach_material (mach4, brass).
mach_material (mach4, aluminum).
mach_material (mach4, steel_LT3Carbon).
mach_material (mach4, steel_LT6Carbon).
mach_material (mach4, steel_LT1Carbon).
mach_material (mach4, iron_soft).
mach_material (mach4, iron_medium).
mach_material (mach4, iron_hard).
mach_material (mach4, stainless_steel).

mach_material (mach5, copper).
mach_material (mach5, brass).
mach_material (mach5, aluminum).
mach_material (mach5, steel_LT3Carbon).
mach_material (mach5, steel_LT6Carbon).
mach_material (mach5, steel_LT1Carbon).
mach_material (mach5, iron_soft).
mach_material (mach5, iron_medium).
mach_material (mach5, iron_hard).
mach_material (mach5, stainless_steel).

/*   OPERATIONS AND THE MACHINES THEY  */
/*   CAN BE PERFORMED ON               */

turning (mach1).
turning (mach2).
turning (mach5).

facing (mach1).
facing (mach2).
facing (mach5).

drilling (mach1).
drilling (mach2).
drilling (mach3).
drilling (mach4).
drilling (mach5).

boring (mach1).
boring (mach2).
boring (mach3).
boring (mach4).
boring (mach5).

reaming (mach1).
reaming (mach2).
reaming (mach3).
reaming (mach4).
reaming (mach5).
```

```
knurling (mach1).
knurling (mach2).
knurling (mach5).

external_threading (mach1).
external_threading (mach2).
external_threading (mach3).

internal_threading (mach1).
internal_threading (mach2).
internal_threading (mach3).
internal_threading (mach4).
internal_threading (mach5).

taperturning (mach1).
taperturning (mach2).
taperturning (mach5).

partoff (mach1).
partoff (mach2).
partoff (mach5).

side_mill (mach3).
side_mill (mach4).

face_mill (mach3).
face_mill (mach4).

plain_mill (mach3).
plain_mill (mach4).

dovetail (mach3).
dovetail (mach4).

slitting (mach3).
slitting (mach4).

end_mill (mach3).
end_mill (mach4).

shell_mill (mach3).
shell_mill (mach4).

pocket_mill (mach3).
pocket_mill (mach4).

/*   INITIAL CONVERSATION WITH THE USER      */
/*   ON STARTING THE USE OF MACRO-CAPP       */

start:-

        writedevice(screen),
        write ("Welcome To MACRO CAPP"),nl,
        write ("An Interactive CAPP System"), nl,
        write ("That Interfaces CAPP With a CIM System"),nl,
        write ("Designed For Implementation"), nl,
       write ("In A Versatile FMS Cell"),
       nl, nl, delay (32000),
        write ("This Integrated System Uses GT Coding"), nl,
        write ("For Interactive Part Description"),
        nl, delay (32000),
```

```
        write ("Please Consult Accurate Part Drawing Prior"), nl,
        write("To Actual Part Description Input"),
        nl, delay (32000),
        write("Please Remember That The Part Input Supplied"),nl,
        write("Indicates To The Software The Information"), nl,
        write("Required On Part Profile"), nl,
        write("And Some Interactive Input"),
        nl, !, nl,
        write("Have A Nice Session"),
        nl, nl, delay (32000), nl,
        write ("The shop contains the following"), nl,
       write ("mach1 - turning_center"), nl,
       write ("mach2 - turning_center"), nl,
       write ("mach3 - horiz_spindle_mach_center"), nl,
       write ("mach4 - vert_spindle_mach_center "), nl,
       write ("mach5 - turning_center"), nl,
        write("Are you ready to start"),nl,
        write ("Start/Wait, S/W"), nl,
       write ("Use Capital's Only"), nl,
       readchar (Ch), Ch = 'S'.


start :-

        write ("The Process Will Be Inititiated"), nl,
        write ("All Over Again"),
        nl, delay (32000),
        write ("Please Prepare To Start"),
        delay (32000),
        start.


/*   DELAY FUNCTION USED TO PREVENT      */
/*   SCREEN FROM SCROLLING RAPIDLY       */


delay (N):-

        N>0 ,!,
        N1 = N-1,
        delay (N1).
       delay (0).


/*   THE PROCESSES THAT CAN BE PERFORMED ON EACH MACHINE */
/*   OUTPUTS WRITTEN TO THE SCREEN AND TO A FILE         */

write_file_machcap1:-

        openwrite (myfile, "machcap1.one"),
        writedevice (screen),
        machtype (mach1, X1),
        write (X1), nl, !,
        write ("The first machine is", X1), nl,
        writedevice (myfile),
        write ("The first machine is", X1),
        nl, !,
        machine (mach1,A1),
        writedevice (myfile),
        write (mach1,A1), nl,
        writedevice (screen), nl,
        write (mach1, A1), nl,
        delay (10000),
```

```
        fail.

write_file_machcap2:-

        openwrite (myfile, "machcap2.one"),
        writedevice (screen),
        machtype (mach2, X2),
        write (X2), nl, !,
        write ("The second machine is", X2), nl,
        writedevice (myfile),
        write ("The second machine is", X2),
        !, machine (mach2,A2),
        writedevice (myfile),
        write (mach2,A2), nl,
        writedevice (screen), nl,
        write (mach2, A2), nl,
        delay (10000), fail.

write_file_machcap3:-

        openwrite (myfile, "machcap3.one"),
        writedevice (screen),
        machtype (mach3, X3),
        write (X3), nl, !,
        write ("The third machine is", X3), nl,
        writedevice (myfile),
        write ("The third machine is", X3),
        !, machine (mach3,A3),
        writedevice (myfile),
        write (mach3,A3), nl,
        writedevice (screen), nl,
        write (mach3, A3), nl,
        delay (10000), fail.

write_file_machcap4:-

        openwrite (myfile, "machcap4.one"),
        writedevice (screen),
        machtype (mach4, X4),
        write (X4), nl, !,
        write ("The fourth machine is", X4), nl,
        writedevice (myfile),
        write ("The fourth machine is", X4),
        !, machine (mach4,A4),
        writedevice (myfile),
        write (mach4,A4), nl,
        writedevice (screen), nl,
        write (mach4, A4), nl,
        delay (10000),
        fail.

write_file_machcap5:-

        openwrite (myfile, "machcap5.one"),
        writedevice (screen),
        machtype (mach5, X5),
```

```
        write (X5), nl, !,
        write ("The fifth machine is", X5), nl,
        writedevice (myfile),
        write ("The fifth machine is", X5),
        !, machine (mach5,A5),
        writedevice (myfile),
        write (mach5,A5), nl,
        writedevice (screen), nl,
        write (mach5, A5), nl,
        delay (10000),
        fail.

/*  IDENTIFICATION OF MATERIALS THAT CAN BE     */
/*  PROCESSED ON EACH MACHINE WRITTEN TO THE    */
/*  SCREEN AND TO A FILE.                       */

    write_file_machmat1:-

        openwrite (myfile, "material1.one"),
        writedevice (myfile),
        write ("The materials that can be"), nl,
        write (" processed on Machine 1 "), nl,
        writedevice (screen),
        write ("The materials that can be"), nl,
        write (" processed on Machine 1 "), nl,
        !, mach_material (mach1, X11),
        writedevice (myfile),
        write (X11), nl,
        writedevice (screen),
        write (X11), nl,
        fail.

    write_file_machmat2:-

        openwrite (myfile, "material2.one"),
        writedevice (myfile),
        write ("The materials that can be"), nl,
        write (" processed on Machine 2 "), nl,
        writedevice (screen),
        write ("The materials that can be"), nl,
        write (" processed on Machine 2 "), nl,
        !, mach_material (mach2, X21),
        writedevice (myfile),
        write (X21), nl,
        writedevice (screen),
        write (X21), nl,
        delay (10000),
        fail.

    write_file_machmat3:-

        openwrite (myfile, "material3.one"),
        writedevice (myfile),
        write ("The materials that can be"), nl,
        write (" processed on Machine 3 "), nl,
        writedevice (screen),
        write ("The materials that can be"), nl,
```

```
        write (" processed on Machine 3 "), nl,
        !, mach_material (mach3, X31),
        writedevice (myfile),
        write (X31), nl,
        writedevice (screen),
        write (X31), nl,
        delay (10000),
        fail.


write_file_machmat4:-


        openwrite (myfile, "material4.one"),
        writedevice (myfile),
        write ("The materials that can be"), nl,
        write (" processed on Machine 4 "), nl,
        writedevice (screen),
        write ("The materials that can be"), nl,
        write (" processed on Machine 4 "), nl,
        !, mach_material (mach4, X41),
        writedevice (myfile),
        write (X41), nl,
        writedevice (screen),
        write (X41), nl,
        fail.


write_file_machmat5:-

        openwrite (myfile, "material5.one"),
        writedevice (myfile),
        write ("The materials that can be"), nl,
        write (" processed on Machine 5 "), nl,
        writedevice (screen),
        write ("The materials that can be"), nl,
        write (" processed on Machine 5 "), nl,
        !, mach_material (mach5, X51),
        writedevice (myfile),
        write (X51), nl,
        writedevice (screen),
        write (X51),
        nl, fail.


/*  THIS SOFTWARE MODULE IDENTIFIES THE     */
/*  JOB TYPES THAT CAN BE PROCESSED ON EACH  */
/*  MACHINE IN THE FACILITY MODELLED        */


write_file_jtype_mach1:-


        openwrite (myfile, "jtype_mach1.one"),
        writedevice (myfile),
        write ("The capacity of Machine 1 includes "),
        nl, writedevice (screen),
        write ("Machine 1's capacity"), nl,
        mach_cap (mach1, Y1),
        writedevice (myfile),
        write (Y1),
        writedevice (screen),
        write (Y1),
```

```
        delay (10000),
        fail.


write_file_jtype_mach2:-

        openwrite (myfile, "jtype_mach2.one"),
        writedevice (myfile),
        write ("The capacity of Machine 2 includes "),
        nl, writedevice (screen),
        write ("Machine 2's capacity"),
        nl, mach_cap (mach2, Y2),
        writedevice (myfile),
        write (Y2),
        writedevice (screen),
        write (Y2),
        fail.


write_file_jtype_mach3:-

        openwrite (myfile, "jtype_mach3.one"),
        writedevice (myfile),
        write ("The capacity of Machine 3 includes "),
        nl, writedevice (screen),
        write ("Machine 3's capacity"),         nl,
        mach_cap (mach3, Y3),
        writedevice (myfile),
        write (Y3),
        writedevice (screen),
        write (Y3),
        fail.


write_file_jtype_mach4:-

        openwrite (myfile, "jtype_mach4.one"),
        writedevice (myfile),
        write ("The capacity of Machine 4 includes "),
        nl, writedevice (screen),
        write ("Machine 4's capacity"), nl,
        mach_cap (mach4, Y4),
        writedevice (myfile),
        write (Y4),
        writedevice (screen),
        write (Y4),
        fail.


write_file_jtype_mach5:-

        openwrite (myfile, "jtype_mach5.one"),
        writedevice (myfile),
        write ("The capacity of Machine 5 includes "),
        nl, writedevice (screen),
        write ("Machine 5's capacity"),
        nl, mach_cap (mach5, Y5),
        writedevice (myfile),
        write (Y5),
        writedevice (screen),
        write (Y5),
```

```
        fail.

/*  THIS SOFTWARE SEGMENT IDENTIFIES THE TYPES OF      */
/*  OPERATIONS THAT CAN BE PERFORMED ON OUTER SURFACES */

    write_file_outer:-

        openwrite (myfile, "outer.one"),
        writedevice (myfile),
        write ("Operations that can be"), nl,
        write ("Performed on the outer surface"), nl,
        writedevice (screen),
        write ("Operations that can be"), nl,
        write ("Performed on the outer surface"),
        nl, outer (Z1),
        writedevice (myfile),
        write (Z1), nl,
        writedevice (screen),
        write (Z1),
        nl, fail.


/*  THIS SEGMENT OF SOFTWARE IDENTIFIES THE          */
/*  OPERATIONS THAT CAN BE PERFORMED ON INNER SURFACES */

    write_file_inner:-

        openwrite (myfile, "inner.one"),
        writedevice (myfile),
        write ("Operations that can be"), nl,
        write ("Performed on the inner surface"),
        nl, writedevice (screen),
        write ("Operations that can be"), nl,
        write ("Performed on the inner surface"), nl,
        internal (Z2),
        writedevice (myfile),
        write (Z2), nl,
        writedevice (screen),
        write (Z2), nl,
        fail.

/*  OPERATIONS THAT CAN BE PERFORMED ON  */
/*  EXTERNAL PLANE SURFACES              */

    write_file_external_plane:-

        openwrite (myfile, "external.one"),
        writedevice (myfile),
        write ("Operations that can be"), nl,
        write ("Performed on the external surface"), nl,
        writedevice (screen),
        write ("Operations that can be"), nl,
        write ("Performed on the external surface"), nl,
        plane_surface (Z3),
        writedevice (myfile),
        write (Z3), nl,
```

```
        writedevice (screen),
        write (Z3),
        nl, fail.

/*  TYPES OF HOLES THAT CAN BE MACHINED     */

    write_file_holes:-

        openwrite (myfile, "holes.one"),
        writedevice (myfile),
        write ("Types of Holes that can be"), nl,
        write ("Drilled On Jobs"), nl,
        writedevice (screen),
        write ("Types Of Holes that can be"), nl,
        write ("Drilled On Jobs"), nl,
        holes (Z4),
        writedevice (myfile),
        write (Z4), nl,
        writedevice (screen),
        write (Z4),
        nl, fail.

/*  MACHINES THAT CAN PERFORM TURNING       */
/*  ON ROTATIONAL JOBS IN FACILITY          */

    write_file_turning:-

        openwrite (myfile, "turning.one"),
        writedevice (myfile),
        write (" Turning can be accomplished "), nl,
        write (" On the following machines"), nl,
        writedevice (screen),
        write (" Turning can be accomplished "), nl,
        write (" On the following machines"), nl,
        turning (Z5),
        writedevice (myfile),
        write (Z5), nl,
        writedevice (screen),
        write (Z5), nl,
        fail.

/*  MACHINES THAT CAN PERFORM FACING.       */

    write_file_facing:-

        openwrite (myfile, "facing.one"),
        writedevice (myfile),
        write (" Facing can be accomplished "), nl,
        write (" On the following machines"), nl,
        writedevice (screen),
        write (" Facing can be accomplished "), nl,
        write (" On the following machines"), nl,
        facing (Z6),
        writedevice (myfile),
        write (Z6), nl,
        writedevice (screen),
        write (Z6), nl,
```

```
        fail.

/*  MACHINES THAT CAN PERFORM DRILLING.     */

    write_file_drilling:-

        openwrite (myfile, "drilling.one"),
        writedevice (myfile),
        write (" Drilling can be accomplished "), nl,
        write (" On the following machines"), nl,
        writedevice (screen),
        write (" Drilling can be accomplished "), nl,
        write (" On the following machines"), nl,
        drilling (Z7),
        writedevice (myfile),
        write (Z7), nl,
        writedevice (screen),
        write (Z7),
        nl, fail.

/*  MACHINES THAT CAN PERFORM BORING.       */

    write_file_boring:-

        openwrite (myfile, "boring.one"),
        writedevice (myfile),
        write (" Boring can be accomplished "), nl,
        write (" On the following machines"), nl,
        writedevice (screen),
        write (" Boring can be accomplished "), nl,
        write (" On the following machines"), nl,
        boring (Z8),
        writedevice (myfile),
        write (Z8), nl,
        writedevice (screen),
        write (Z8), nl,
        delay (10000),
        fail.

/*  MACHINES THAT CAN PERFORM REAMING.      */

    write_file_reaming :-

        openwrite (myfile, "reaming.one"),
        writedevice (myfile),
        write (" Reaming can be accomplished "), nl,
        write (" On the following machines"), nl,
        writedevice (screen),
        write (" Reaming can be accomplished "), nl,
        write (" On the following machines"), nl,
        reaming (Z9),
        writedevice (myfile),
        write (Z9), nl,
        writedevice (screen),
        write (Z9),
        nl, fail.
```

```
/*   MACHINES THAT CAN PERFORM KNURLING.      */

     write_file_knurling :-

          openwrite (myfile, "knurling.one"),
          writedevice (myfile),
          write (" Knurling can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Knurling can be accomplished "), nl,
          write (" On the following machines"), nl,
          knurling (Z10),
          writedevice (myfile),
          write (Z10), nl,
          writedevice (screen),
          write (Z10),
          nl, fail.

/*   MACHINES THAT CAN PERFORM EXTERNAL THREADING */

     write_file_external_threading :-

          openwrite (myfile, "extthread.one"),
          writedevice (myfile),
          write (" External Threading can be accomplished "), nl,
          write (" On the following machines"),
          nl, writedevice (screen),
          write (" External Threading can be accomplished "), nl,
          write (" On the following machines"), nl,
          external_threading (Z11),
          writedevice (myfile),
          write (Z11), nl,
          writedevice (screen),
          write (Z11), nl,
          delay (10000), fail.

/*   MACHINES THAT CAN PERFORM INTERNAL THREADING */

     write_file_internal_threading :-

          openwrite (myfile, "intthread.one"),
          writedevice (myfile),
          write (" Internal Threading can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Internal Threading can be accomplished "), nl,
          write (" On the following machines"), nl,
          internal_threading (Z12),
          writedevice (myfile),
          write (Z12), nl,
          writedevice (screen),
          write (Z12), nl,
          delay (10000),
          fail.
```

```
/*   MACHINES THAT CAN PERFORM TAPER TURNING          */

     write_file_taperturning :-

          openwrite (myfile, "tapturn.one"),
          writedevice (myfile),
          write (" Taper Turning can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Taper Turning can be accomplished "), nl,
          write (" On the following machines"), nl,
          taperturning (Z13),
          writedevice (myfile),
          write (Z13), nl,
          writedevice (screen),
          write (Z13), nl,
          delay (10000),
          fail.

/*   MACHINES THAT CAN PERFORM PARTOFF.               */

     write_file_partoff :-

          openwrite (myfile, "partoff.one"),
          writedevice (myfile),
          write (" Parting Off can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Part off can be accomplished "), nl,
          write (" On the following machines"), nl,
          partoff (Z14),
          writedevice (myfile),
          write (Z14), nl,
          writedevice (screen),
          write (Z14), nl,
          delay (10000),
          fail.

/*   MACHINES THAT CAN PERFORM SIDE MILLING.          */

     write_file_side_mill :-

          openwrite (myfile, "sidemill.one"),
          writedevice (myfile),
          write (" Side Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Side Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          side_mill (Z15),
          writedevice (myfile),
          write (Z15), nl,
          writedevice (screen),
          write (Z15), nl,
          delay (10000),
          fail.
```

```
/*   MACHINES THAT CAN PERFORM FACE MILLING.                    */

     write_file_face_mill :-

          openwrite (myfile, "facemill.one"),
          writedevice (myfile),
          write (" Face Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Face Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          face_mill (Z16),
          writedevice (myfile),
          write (Z16), nl,
          writedevice (screen),
          write (Z16), nl,
          delay (10000),
          fail.

/*   MACHINES THAT CAN PERFORM PLAIN MILLING.                   */

     write_file_plain_mill :-

          openwrite (myfile, "plainmill.one"),
          writedevice (myfile),
          write (" Plain Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Plain Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          plain_mill (Z17),
          writedevice (myfile),
          write (Z17), nl,
          writedevice (screen),
          write (Z17), nl,
          delay (10000),
          fail.

/*   MACHINES THAT CAN PERFORM DOVETAIL MILLING.                */

     write_file_dovetail :-

          openwrite (myfile, "dovetail.one"),
          writedevice (myfile),
          write (" Dovetail Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Dovetail Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          dovetail (Z18),
          writedevice (myfile),
          write (Z18), nl,
          writedevice (screen),
          write (Z18), nl,
          delay (10000),
          fail.
```

```
/*   MACHINES THAT CAN PERFORM SLITTING.                */

     write_file_slitting :-

          openwrite (myfile, "slitting.one"),
          writedevice (myfile),
          write (" Slitting can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Slitting can be accomplished "), nl,
          write (" On the following machines"), nl,
          slitting (Z19),
          writedevice (myfile),
          write (Z19), nl,
          writedevice (screen),
          write (Z19), nl,
          delay (10000),
          fail.

/*   MACHINES THAT CAN PERFORM END MILLING.             */

     write_file_end_mill :-

          openwrite (myfile, "endmill.one"),
          writedevice (myfile),
          write (" End Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" End Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          end_mill (Z20),
          writedevice (myfile),
          write (Z20), nl,
          writedevice (screen),
          write (Z20), nl,
          delay (10000),
          fail.

/*   MACHINES THAT CAN PERFORM SHELL MILLING.           */

     write_file_shell_mill :-

          openwrite (myfile, "shellmill.one"),
          writedevice (myfile),
          write (" Shell Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          writedevice (screen),
          write (" Shell Milling can be accomplished "), nl,
          write (" On the following machines"), nl,
          shell_mill (Z21),
          writedevice (myfile),
          write (Z21), nl,
          writedevice (screen),
          write (Z21), nl,
          delay (10000),
          fail.
```

```
/*  MACHINES THAT CAN PERFORM POCKET MILLING.                */

    write_file_pocket_mill :-

        openwrite (myfile, "pocketmill.one"),
        writedevice (myfile),
        write (" Pocket Milling can be accomplished "), nl,
        write (" On the following machines"), nl,
        writedevice (screen),
        write (" Pocket Milling can be accomplished "), nl,
        write (" On the following machines"), nl,
        pocket_mill (Z22),
        writedevice (myfile),
        write (Z22), nl,
        writedevice (screen),
        write (Z22), nl,
        delay (10000),
        fail.

/*  READ IN DATA DESCRIBING THE PART TO BE MACHINED */

    read_comp :-

        openwrite (myfile, "try1.one"),
        nl, writedevice (screen), nl,
        write (" The Input Consists Of Five Digit Codes"),
        nl, write ("And Several Interactive Inputs "),
        nl, nl,        write ("Digit One Inputs Component Class"),
        nl, write ("Component Class ?"),
        readint (Comp_Class), nl,
        writedevice (myfile),
        write ("Component Class", Comp_Class),
        nl, writedevice (screen),

/*  DIGIT TWO INPUTS INFORMATION ON EXTERNAL SHAPE  */

        write (" Digit Two Explains the External Shape of "), nl,
        write (" The Product To The System Software "), nl,
        write (" External Shape ? "),
        readint (Ext_Shape), nl, nl,
        write ("External Shape", Ext_Shape), nl, nl,
        writedevice (myfile),
        write ("External Shape", Ext_Shape), nl,
        writedevice (screen),

/*  DIGIT THREE CONSIDERS THE INTERNAL CONFIGURATION  */

        write (" Digit Three Explains The Internal Configuration of "), nl,
        write (" The part to System Software "), nl,
        write (" Internal Shape ? "),
        readint (Int_Shape), nl,
        write ("Internal Shape", Int_Shape), nl,
        writedevice (myfile),
        write ("Internal Shape", Int_Shape), nl,
        writedevice (screen),
```

```
/*    THE FOURTH DIGIT CONSIDERS MACHINING REQUIRED      */
/*    ON THE OUTER SURFACE                               */

        write (" Digit Four Explains The Machining "), nl,
        write (" Required On The Outer Surface "), nl,
        write ("Outer Surface Plane Machining ? "),
        readint (Plane_Surface_Mach), nl,
        write ("Plane Surface Machining", Plane_Surface_Mach), nl,
        writedevice (myfile),
        write ("Plane Surface Machining", Plane_Surface_Mach), nl,
        writedevice (screen),


/*    OF THE AUXILLIARY HOLES                            */

        write (" The Fifth Digit Describes The Position "), nl,
        write (" And Number of Auxilliary Holes "), nl,
        write (" Digit Five ? "), readint (Aux_Holes), nl,
        write ("Auxilliary Holes Required", Aux_Holes), nl,
        writedevice (myfile),
        write ("Auxilliary Holes Required", Aux_Holes), nl,
        writedevice (screen).


/*    INTERACTIVE DATA ON AUXILLIARY FEATURES            */


    readaux :-

        write ( "These inputs are with respect to "), nl,
        write ( "Auxilliary Information Required  "), nl,
        write ( "What is the Initial Dimension ?  "), nl,
        readint (Dimension), nl,
        write ("Dimension Of Job", Dimension), nl,
        writedevice (myfile),
        write ("Dimension Of Job", Dimension), nl,
        writedevice (screen),
        write ( "What is the material Type ?"), nl,
        readln (Material_Type), nl,
        write ("Type Of Material", Material_Type), nl,
        writedevice (myfile),
        write ("Type Of Material", Material_Type), nl,
        writedevice (screen),
        write ( "What is the Initial Form Processed ?"), nl,
        readln (Init_Form),
        write ("Initial Form Processed"),
        write (Init_Form),
        writedevice (myfile),
        write ("Initial Form Processed", Init_Form), nl,
        writedevice (screen),
        write (" What is the Tolerance Desired ? "), nl,
        readint (Tolerance), nl,
        write ("Tolerance", Tolerance), nl,
        writedevice (myfile),
        write ("Tolerance", Tolerance), nl,
        writedevice (screen),
        write ("What is the Surface_Finish Desired ? "), nl,
        readint (Surface_Finish), nl,
        write ("Surface Finish Of Job", Surface_Finish),
        nl, writedevice (myfile),
        write ("Surface Finish", Surface_Finish), nl,
```

```
                closefile (myfile),
                writedevice (screen).

/*      CHECK IF ALL INPUT IS CORRECT          */

        run:-

                writedevice (screen),
                delay (32000),
                start,
                delay (32000),
                read_comp,
                delay (32000),
                readaux, nl,
                write ("Is The Input Okay ? Yes/No "), nl,
                readchar (Ch),
                Ch = 'Y', nl,
                delay (32000),
                write ("The Input Has Been Transferred"), nl,
                write ("To A File"), nl, !,
/*              go, mach_operations,
  */            rot_surface_finish_128.


/*      IF INPUT IS NOT CORRECT RE-ENTER       */

        run:-

                write ("Please Check Your Inputs"), nl,
                write ("And Re-enter inputs once again"),
                nl, run.


        go :-

                machine (V,turning),
                nl, write(V), nl,
                delay (10000),
/*              machine (F, turning),
                write ("The Capacity of  ", F), nl,
                write ("Includes ", G), nl,
                fail, nl,
             */ mach_use(A,B),
                machtype(A,B),
                mach_cap(A,cubic),
                jtype(cubic),
                write (" Machines that could be used"), nl,
                write (" for a cubic job are  ", A, B),
                nl, machine (mach1, Y1),
                nl,delay (32000),
                write (Y1),
                delay (32000), nl,
                writedevice(screen),
                jtype(cubic),
                outer (smooth),
                thread (yes),
                operations_reqd (D, smooth),
                write ("To secure a smooth"), nl,
                write ("Cylindrical outer surface"), nl,
                write ("Use ", D), nl,
```

```
            operations_reqd (E, thread),
            write ("To produce threads process"), nl,
            write (" used includes ", E),
            nl, machine (X, turning),
            write (X)./*
            machine (F, turning),
            write ("The Capacity of  ", F), nl,
           write ("Includes ", G), nl,
*/
      mach_operations:-

            delay (32000),
            machine(mach1, Z1),
            write (Z1),
            nl, fail.

/* OPERATIONS REQUIRED TO PROCESS THE PART CONSIDERING */
/* THE FIRST DIGIT OF THE INPUT CODE.                  */


      rotational_short :-

                  openwrite (myfile, "digit1.one"),
                  jtype (rot),
                  mach_use (mach1,A1),
                  write ("A1", A1), nl,
                  writedevice (myfile),
                  write (A1), nl,
                  mach_use (mach2,A2),
                  write (A2), nl,
                  writedevice (screen),
                  write ("A2", A2), nl,
                  mach_use (mach5,A5),
                  write ("A5", A5),
                  nl, writedevice (myfile),
                  write (A5), nl,
                  closefile (myfile),
                  openwrite (myfile, "digit11.one"),
                  writedevice (myfile),
                  write ("1"), nl,
                  closefile (myfile),
                  openwrite (myfile, "digit12.one"),
                  writedevice (myfile),
                  write ("1"), nl,
                  write ("2"), nl,
                  write ("5"), nl,
                  closefile (myfile).


      rotational :-
                  openwrite (myfile, "digit1.one"),
                  jtype (rot),
                  mach_use (mach1,A1),
                  write ("A1", A1), nl,
                  writedevice (myfile),
                  write (A1), nl,
                  mach_use (mach2,A2),
                  write (A2), nl,
                  writedevice (screen),
                  write ("A2", A2), nl,
                  mach_use (mach5,A5),
                  write ("A5", A5), nl,
                  writedevice (myfile),
```

```
        write (A5), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit11.one"),
        writedevice (myfile),
        write ("2"), nl,
        closefile (myfile),
        openwrite (myfile, "digit12.one"),
        writedevice (myfile),
        write ("1"), nl,
        write ("2"), nl,
        write ("5"), nl,
        closefile (myfile).

rotational_long :-

        openwrite (myfile, "digit1.one"),
        jtype (rot),
        mach_use (mach1,A1),
        write ("A1", A1), nl,
        writedevice (myfile),
        write (A1), nl,
        mach_use (mach2,A2),
        write (A2), nl,
        writedevice (screen),
        write ("A2", A2), nl,
        mach_use (mach5,A5),
        write ("A5", A5), nl,
        writedevice (myfile),
        write (A5), nl,
        closefile (myfile),
        openwrite (myfile, "digit11.one"),
        writedevice (myfile),
        write ("3"), nl,
        closefile (myfile),
        openwrite (myfile, "digit12.one"),
        writedevice (myfile),
        write ("1"), nl,
        write ("2"), nl,
        write ("5"), nl,
        closefile (myfile).

cubical_long :-

        openwrite (myfile, "digit1.one"),
    jtype (cubic),
    writedevice (myfile),
        mach_use (mach3,A3),
        write (A3), nl,
        writedevice (screen),
        write ("A3", A3), nl,
        mach_use (mach4,A4),
        write ("A4", A4), nl,
        writedevice (myfile),
        write (A4), nl,
        closefile (myfile),
        openwrite (myfile, "digit11.one"),
        writedevice (myfile),
        write ("3"), nl,
        closefile (myfile),
        openwrite (myfile, "digit12.one"),
```

```
                writedevice (myfile),
                write ("4"), nl,
                write ("5"),
                closefile (myfile).


        cubical :-


                 openwrite (myfile, "digit1.one"),
              jtype (cubic),
              writedevice (myfile),
                 mach_use (mach3,A3),
                 write (A3), nl,
                writedevice (screen),
                write ("A3", A3), nl,
                 mach_use (mach4,A4),
                 write ("A4", A4), nl,
                writedevice (myfile),
                write (A4), nl,
                closefile (myfile),
                 openwrite (myfile, "digit11.one"),
                 writedevice (myfile),
                 write ("5"), nl,
                 closefile (myfile),
                 openwrite (myfile, "digit12.one"),
                 writedevice (myfile),
                 write ("3"), nl,
                 write ("4"),
                 closefile (myfile).


        cubical_flat :-

                 openwrite (myfile, "digit1.one"),
                 jtype (cubic),
                 writedevice (myfile),
                 mach_use (mach3,A3),
                 write (A3), nl,
                writedevice (screen),
                write ("A3", A3), nl,
                 mach_use (mach4,A4),
                 write ("A4", A4), nl,
                writedevice (myfile),
                write (A4), nl,
                closefile (myfile),
                 openwrite (myfile, "digit11.one"),
                 writedevice (myfile),
                 write ("6"), nl,
                 closefile (myfile),
                 openwrite (myfile, "digit12.one"),
                 writedevice (myfile),
                 write ("3"), nl,
                 write ("4"), nl,
                 closefile.


/* OPERATIONS REQUIRED TO PROCESS THE PART      */
/* CONSIDERING THE SECOND DIGIT OF THE INPUT CODE. */


        outer_smooth :-
```

```
        openwrite (myfile, "digit2.one"),
        jtype (rot),
        outer (smooth),
        operations_reqd (B21, smooth),
        writedevice (screen),
        write ("B21", B21), nl,
        writedevice (myfile),
        write (B21), nl,
        closefile (myfile),
        openwrite (myfile, "digit21.one"),
        writedevice (myfile),
        write ("21"),
        closefile (myfile).

outer_thread :-

        openwrite (myfile, "digit2.one"),
        jtype (rot),
        outer (thread),
        operations_reqd (B22, thread),
        writedevice (screen),
        write ("B22", B22), nl,
        write ("B22", "tapping"), nl,
        write ("B22", "die_threads"), nl,
        writedevice (myfile),
        write (B22), nl,
        write ("tapping"), nl,
        write ("die_threads"), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit21.one"),
        writedevice (myfile),
        write ("22"), nl,
        write ("23"), nl,
        write ("24"), nl,
        closefile (myfile).


  outer_groove :-

        openwrite (myfile, "digit2.one"),
        jtype (rot),
        outer (groove),
        operations_reqd (B23, groove),
        writedevice (myfile), nl,
        write (B23),
        nl, writedevice (screen),
        write ("B23", B23), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit21.one"),
        writedevice (myfile),
        write ("25"), nl,
        closefile (myfile).

  outer_slot :-
```

```
        openwrite (myfile, "digit2.one"),
        jtype (rot),
        outer (slot),
        operations_reqd (B24, slot),
        writedevice (myfile),
        write (B24), nl,
        write (end_mill),
        writedevice (screen),
        write ("B24", B24), nl,
        write ("B24", endmill), nl,
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit21.one"),
        writedevice (myfile),
        write ("26"), nl,
        write ("27"), nl,
        closefile (myfile).


outer_stepped_one_side :-


        openwrite (myfile, "digit2.one"),
        jtype (rot),
        outer (stepped_one_side),
        operations_reqd (B25, stepped_one_side), nl,
        writedevice (myfile),
        write (B25), nl,
        writedevice (screen),
        write ("B25", B25), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit21.one"),
        writedevice (myfile),
        write ("28"), nl,
        write ("28"), nl,
        closefile (myfile).


outer_stepped_both_sides :-


        openwrite (myfile, "digit2.one"),
        jtype (rot),
        outer (stepped_both_sides),
        operations_reqd (B26, stepped_both_sides),
        writedevice (myfile),
        write (B26), nl,
        writedevice (screen),
        write ("B26", B26),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit21.one"),
        writedevice (myfile),
        write ("29"), nl,
        write ("29"), nl,
        closefile (myfile).


/*      outer_smooth_chamfers:-

        openwrite (myfile, "digit2.one"),
```

```
        jtype (rot),
        outer (smooth_chamfers),
        operations_reqd (B27, chamfers),
        write ("B27",B27), nl,
        writedevice (myfile),
        write (B27), !,
        nl, closefile (myfile).
*/

    outer_no_plane_machining:-


        openwrite (myfile, "digit2.one"),
        writedevice (screen),
        jtype (cubic),
        outer (no_plane_mach),
        writedevice (screen),
        operations_reqd (B210, no_plane_mach),
        write ("B210", B210), nl,
        writedevice (myfile),
        write (B210), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit21.one"),
        writedevice (myfile),
        write ("30"), nl,
        write ("30"), nl,
        closefile (myfile).


    outer_plane_machining :-


        openwrite (myfile, "digit2.one"),
        jtype (cubic),
        outer (plane_mach),
        writedevice (screen),
        operations_reqd (B211, plane_mach),
        write ("B211", B211), nl,
        write ("B211", "face_mill"), nl,
        write ("B211", "plain_mill"), nl,
        writedevice (myfile),
        write (B211),
        write ("face_mill"), nl,
        write ("plain_mill"), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit21.one"),
        writedevice (myfile),
        write ("31"), nl,
        write ("32"), nl,
        write ("33"), nl,
        closefile (myfile).


    outer_plane_machining_chamfers:-


        openwrite (myfile, "digit2.one"),
        jtype (cubic),
        outer (plane_mach_chamfers),
        writedevice (screen),
        operations_reqd (B212, plane_mach_chamfers),
```

```
        write ("B212", B212), nl,
        writedevice (myfile),
        write (B212),
        write ("side_milling"), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit21.one"),
        writedevice (myfile),
        write ("31"), nl,
        write ("32"), nl,
        write ("33"), nl,
        closefile (myfile).

/* OPERATION SREQUIRED TO PROCESS THE PART        */
/* CONSIDERING THE THIRD DIGIT OF THE INPUT CODE. */


    internal_none:-

        openwrite (myfile, "digit3.one"),
        writedevice (screen),
        jtype (cubic),
        internal (none),
        writedevice (screen),
        operations_reqd (B31, int_none),
        write ("B31", B31),
        nl, writedevice (myfile),
        write (B31), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile,"digit31.one"),
        write ("40"),
        nl,
        closefile (myfile).


    internal_smooth:-


        openwrite (myfile, "digit3.one"),
        jtype (cubic),
        internal (smooth),
        writedevice (screen),
        operations_reqd (B32, int_smooth),
        write ("B32",B32), nl,
        writedevice (myfile),
        write (B32), nl,
        closefile (myfile),
        openwrite (myfile,"digit31.one"),
        writedevice (myfile),
        write ("41"), nl,
        delay (10000),
        closefile (myfile).


    internal_stepped_one_side :-


        openwrite (myfile, "digit3.one"),
        jtype (cubic),
        internal (stepped_one_side),
        writedevice (screen),
        operations_reqd (B33, int_stepped_one_side),
```

```
        write ("B33",B33), nl,
        writedevice (myfile),
        write (B33), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
        writedevice (myfile),
        write ("42"), nl,
        closefile (myfile).


internal_stepped_both_side:-

        openwrite (myfile, "digit3.one"),
       jtype (cubic),
        internal (stepped_both_side),
        writedevice (screen),
        operations_reqd (B34, int_stepped_both_sides),
        write ("B34",B34), nl,
        writedevice (myfile),
        write (B34), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
        writedevice (myfile),
        write ("43"), nl,
        closefile (myfile).


internal_slot :-

        openwrite (myfile, "digit3.one"),
        jtype (cubic),
        internal (slot),
        writedevice (screen),
        operations_reqd (B35, int_slot),
        write ("B35",B35), nl,
        writedevice (myfile),
        write (B35), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
        writedevice (myfile),
        write ("44"), nl,
        closefile (myfile).


internal_thread:-


        openwrite (myfile, "digit3.one"),
        jtype (cubic),
        internal (thread),
        writedevice (screen),
        operations_reqd (B36, int_thread),
        write ("B36",B36), nl,
        writedevice (myfile),
        write (B36),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
       writedevice (myfile),
       write ("45"), nl,
```

```
        closefile (myfile).

internal_groove:-

        openwrite (myfile, "digit3.one"),
        jtype (cubic),
        internal (groove),
        writedevice (screen),
        operations_reqd (B37, int_groove),
        write ("B37", B37), nl,
        writedevice (myfile),
        write (B37),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
        writedevice (myfile),
        write ("46"), nl,
        closefile (myfile).

internal_two_principal_bores:-

        openwrite (myfile, "digit3.one"),
       jtype (cubic),
       internal (two_principal_bores),
        writedevice (screen),
        operations_reqd (B38, int_two_principal_bores),
        write ("B38", B38), nl,
        writedevice (myfile),
        write (B38),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
        writedevice (myfile),
        write ("47"), nl,
        closefile (myfile).

internal_multiple_parallel_bores :-

        openwrite (myfile, "digit3.one"),
       jtype (cubic),
       internal (multiple_parallel_bores),
        writedevice (screen),
        operations_reqd (B39, int_multiple_parallel_bores),
        write ("B39",B39), nl,
        writedevice (myfile),
        write (B39),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
        writedevice (myfile),
        write ("48"), nl,
        closefile (myfile).

internal_non_parallel_bores:-
```

```
        openwrite (myfile, "digit3.one"),
        jtype (cubic),
        internal (non_parallel_bores),
        writedevice (screen),
        operations_reqd (B391, int_non_paralled_bores),
        write ("B391", B391), nl,
        writedevice (myfile),
        write (B391),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit31.one"),
        writedevice (myfile),
        write ("49"),nl,
        closefile (myfile).

/* OPERATIONS REQUIRED TO MACHINE THE PART      */
/* CONSIDERING THE FOURTH DIGIT OF INPUT CODE   */


    surface_plane_none:-

        openwrite (myfile, "digit4.one"),
        writedevice (screen),
        jtype (cubic),
        plane_surface (none),
        writedevice (screen),
        operations_reqd (B41, plane_none),
        write ("B41", B41), nl,
        writedevice (myfile),
        write (B41), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit41.one"),
        writedevice (myfile),
        write ("61"), nl,
        closefile (myfile).


    surface_plane_external_single_curve:-


        openwrite (myfile, "digit4.one"),
        jtype (cubic),
        plane_surface (external_single_curve),
        writedevice (screen),
        operations_reqd (B42, plane_external_single_curve),
        writedevice (screen),
        write ("B42",B42), nl,
        write ("B42", "face_milling"), nl,
        write ("B42", "side_milling"), nl,
        writedevice (myfile),
        write (B42),
        write ("face_milling"), nl,
        write ("side_milling"),    nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit41.one"),
        writedevice (myfile),
        write ("62"), nl,
        write ("63"), nl,
        write ("64"), nl,
        closefile (myfile).
```

```
surface_plane_external_multi_curve:-

        openwrite (myfile, "digit4.one"),
       jtype (cubic),
       plane_surface (external_multi_curve),
        writedevice (screen),
        operations_reqd (B43, plane_external_multi_curve),
        write ("B43", B43), nl,
        write ("B43", "face_mill"), nl,
        write ("B43", "side_mill"), nl,
        writedevice (myfile),
        write (B43),
        write ("face_mill"), nl,
        write ("side_mill"), nl,
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit41.one"),
        writedevice (myfile),
        write ("65"), nl,
        write ("66"), nl,
        write ("67"), nl,
        closefile (myfile).


surface_plane_external_keyway :-

        openwrite (myfile, "digit4.one"),
       jtype (cubic),
       plane_surface (keyway),
        writedevice (screen),
        operations_reqd (B44, plane_keyway),
        write ("B44",B44), nl,
        write ("B44", "pocket_mill"), nl,
        writedevice (myfile),
        write (B44), nl,
        write ("pocket_mill"), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit41.one"),
        writedevice (myfile),
        write ("68"), nl,
        write ("69"), nl,
        closefile (myfile).


surface_plane_external_groove :-

        openwrite (myfile, "digit4.one"),
       jtype (cubic),
       plane_surface (groove),
        writedevice (screen),
        operations_reqd (B45, plane_surface_groove),
        write ("B45", B45), nl,
        write ("B45", "pocket_mill"), nl,
        write ("B45", "slitting_mill"), nl,
        write ("B45", "shell_mill"), nl,
        writedevice (myfile),
        write (B45), nl,
        write ("pocket_mill"), nl,
```

```
        write ("slitting_mill"), nl,
        write ("shell_mill"), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit41.one"),
        writedevice (myfile),
        write ("70"), nl,
        write ("71"), nl,
        write ("72"),
        nl, write ("73"), nl,
        closefile (myfile).

surface_plane_slot :-

        openwrite (myfile, "digit4.one"),
       jtype (cubic),
       plane_surface (slot),
        writedevice (screen),
        operations_reqd (B46, plane_surface_slot),
        write ("B46", B46), nl,
        write ("B46", "slitting_mill"), nl,
        write ("B46", "shell_mill"), nl,
        writedevice (myfile),
        write (B46), nl,
        write ("slitting_mill"), nl,
        write ("shell_mill"), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit41.one"),
        writedevice (myfile),
        write ("74"), nl,
        write ("75"), nl,
        write ("76"), nl,
        closefile (myfile).


/* OPERATIONS REQUIRED TO MACHINE THE PART        */
/* CONSIDERING THE FIFTH DIGIT OF THE INPUT CODE. */

holes_none :-

        openwrite (myfile, "digit5.one"),
        writedevice (screen),
        jtype (cubic),
        holes (none),
        writedevice (screen),
        operations_reqd (B51, holes_none),
        write ("B51", B51), nl,
        writedevice (myfile),
        write (B51), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("81"), nl,
        closefile (myfile).


holes_axial_related :-
```

```
        openwrite (myfile, "digit5.one"),
        jtype (cubic),
        holes (axial_related),
        writedevice (screen),
        operations_reqd (B52, holes_axial_related),
        write ("B52",B52), nl,
        writedevice (myfile),
        write (B52),
        closefile (myfile), nl,
        delay (10000),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("82"),
        closefile (myfile).

holes_axial_unrelated :-

        openwrite (myfile, "digit5.one"),
       jtype (cubic),
       holes (axial_unrelated),
        writedevice (screen),
        operations_reqd (B53, holes_axial_unrelated),
        write ("B53",B53), nl,
        writedevice (myfile),
        write (B53), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("83"), nl,
        closefile (myfile).

holes_radial_related :-

        openwrite (myfile, "digit5.one"),
        jtype (cubic),
        holes (radial_related),
        writedevice (screen),
        operations_reqd (B54, holes_radial_related),
        write ("B54",B54), nl,
        writedevice (myfile),
        write (B54),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("84"),
        closefile (myfile).

holes_radial_unrelated :-

        openwrite (myfile, "digit5.one"),
       jtype (cubic),
       holes (radial_unrelated),
        writedevice (screen),
```

```
        operations_reqd (B55, holes_radial_unrelated),
        write ("B55",B55), nl,
        writedevice (myfile),
        write (B55),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("85"),
        closefile (myfile).


holes_multidirection :-

        openwrite (myfile, "digit5.one"),
       jtype (cubic),
       holes (multidirection),
        writedevice (screen),
        operations_reqd (B56, holes_multidirection),
        write ("B56",B56), nl,
        writedevice (myfile),
        write (B56),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("86"),
        closefile (myfile).


holes_related_multidirection :-

        openwrite (myfile, "digit5.one"),
       jtype (cubic),
       holes (related_multidirection),
       writedevice (screen),
        operations_reqd (B57, holes_related_multidirection),
        write ("B57",B57), nl,
        writedevice (myfile),
        write (B57), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("87"),
        closefile (myfile).




/* OPERATIONS REQUIRED TO MACHINE THE PART       */
/* CONSIDERING THE FIFTH DIGIT OF THE INPUT CODE. */


holes_none :-

        openwrite (myfile, "digit5.one"),
        writedevice (screen),
        jtype (cubic),
        holes (none),
        writedevice (screen),
```

```
        operations_reqd (B51, holes_none),
        write ("B51", B51), nl,
        writedevice (myfile),
        write (B51), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("81"), nl,
        closefile (myfile).


holes_axial_related :-


        openwrite (myfile, "digit5.one"),
        jtype (cubic),
        holes (axial_related),
        writedevice (screen),
        operations_reqd (B52, holes_axial_related),
        write ("B52",B52), nl,
        writedevice (myfile),
        write (B52),
        closefile (myfile), nl,
        delay (10000),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("82"),
        closefile (myfile).


holes_axial_unrelated :-


        openwrite (myfile, "digit5.one"),
       jtype (cubic),
       holes (axial_unrelated),
        writedevice (screen),
        operations_reqd (B53, holes_axial_unrelated),
        write ("B53",B53), nl,
        writedevice (myfile),
        write (B53), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("83"), nl,
        closefile (myfile).


holes_radial_related :-


        openwrite (myfile, "digit5.one"),
        jtype (cubic),
        holes (radial_related),
        writedevice (screen),
        operations_reqd (B54, holes_radial_related),
        write ("B54",B54), nl,
        writedevice (myfile),
        write (B54),
        nl, delay (10000),
        closefile (myfile),
```

```
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("84"),
        closefile (myfile).


holes_radial_unrelated :-


        openwrite (myfile, "digit5.one"),
        jtype (cubic),
        holes (radial_unrelated),
        writedevice (screen),
        operations_reqd (B55, holes_radial_unrelated),
        write ("B55",B55), nl,
        writedevice (myfile),
        write (B55),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("85"),
        closefile (myfile).


holes_multidirection :-

        openwrite (myfile, "digit5.one"),
        jtype (cubic),
        holes (multidirection),
        writedevice (screen),
        operations_reqd (B56, holes_multidirection),
        write ("B56",B56), nl,
        writedevice (myfile),
        write (B56),
        nl, delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("86"),
        closefile (myfile).


holes_related_multidirection :-

        openwrite (myfile, "digit5.one"),
        jtype (cubic),
        holes (related_multidirection),
        writedevice (screen),
        operations_reqd (B57, holes_related_multidirection),
        write ("B57",B57), nl,
        writedevice (myfile),
        write (B57), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit51.one"),
        writedevice (myfile),
        write ("87"),
        closefile (myfile).
```

```
/*  THIS SOFTWARE SEGMENT IDENTIFIES THE    */
/*  MACHINES THAT CAN BE USED TO MACHINE A  */
/*  SPECIFIC PART MATERIAL.                 */


      material_copper :-


            openwrite (myfile, "digit6.one"),
            material (copper),
            writedevice (screen),
            mach_material (B61, copper),
            write ("B61",B61), nl,
            write ("B61",mach2), nl,
            write ("B61", mach3), nl,
            write ("B61", mach4), nl,
            write ("B61", mach5), nl,
            writedevice (myfile),
            write ("B61", mach1), nl,
            write ("B61", mach2), nl,
            write ("B61", mach3), nl,
            write ("B61", mach4), nl,
            write ("B61", mach5), nl,
            delay (10000),
            closefile (myfile),
            openwrite (myfile, "digit61.one"),
            writedevice (myfile),
            write ("X91"),
            delay(1000),
            closefile (myfile).


      material_brass :-


            openwrite (myfile, "digit6.one"),
            material (brass),
            writedevice (screen),
            mach_material (B62, brass),
            write ("B62",B62), nl,
            write ("B62",mach2), nl,
            write ("B62", mach3), nl,
            write ("B62", mach4), nl,
            write ("B62", mach5), nl,
            writedevice (myfile),
            write ("B62", B62), nl,
            write ("B62", mach2), nl,
            write ("B62", mach3), nl,
            write ("B62", mach4), nl,
            write ("B62", mach5), nl,
            delay (10000),
            closefile (myfile),
            openwrite (myfile, "digit61.one"),
            writedevice (myfile),
            write ("92"),nl,
            closefile (myfile).


      material_aluminum :-


            openwrite (myfile, "digit6.one"),
            material (aluminum),
```

```
        writedevice (screen),
        mach_material (B63, aluminum),
        write ("B63",B63), nl,
        write ("B63", mach2), nl,
        write ("B63", mach3), nl,
        write ("B63", mach4), nl,
        write ("B63", mach5), nl,
        writedevice (myfile),
        write ("B63", B63), nl,
        write ("B63", mach2), nl,
        write ("B63", mach3), nl,
        write ("B63", mach4), nl,
        write ("B63", mach5), nl,
        closefile (myfile),
        delay (10000),
        openwrite (myfile, "digit61.one"),
        writedevice (myfile),
        write ("93"),
        closefile (myfile).

material_steel_LT3Carbon :-

        openwrite (myfile, "digit6.one"),
        material (steel_LT3Carbon),
        writedevice (screen),
        mach_material (B64, steel_LT3Carbon),
        write ("B64",B64), nl,
        write ("B64", mach2), nl,
        write ("B64", mach3), nl,
        write ("B64", mach4), nl,
        write ("B64", mach5), nl,
        writedevice (myfile),
        write ("B64", B64), nl,
        write ("B64", mach2), nl,
        write ("B64", mach3), nl,
        write ("B64", mach4), nl,
        write ("B64", mach5), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit61.one"),
        writedevice (myfile),
        write ("94"),
        closefile (myfile).

material_steel_LT6Carbon :-

        openwrite (myfile, "digit6.one"),
        material (steel_LT6Carbon),
        writedevice (screen),
        mach_material (B65, steel_LT6Carbon),
        write ("B65", B65), nl,
        write ("B65", mach2), nl,
        write ("B65", mach3), nl,
        write ("B65", mach4), nl,
        write ("B65", mach5), nl,
        delay (10000),
        closefile (myfile),
        writedevice (myfile),
        write ("B65", B65), nl,
```

```
        write ("B65", mach2), nl,
        write ("B65", mach3), nl,
        write ("B65", mach4), nl,
        write ("B65", mach5), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit61.one"),
        writedevice (myfile),
        write ("95"),
        closefile (myfile).


material_steel_LT1Carbon :-


        openwrite (myfile, "digit6.one"),
       material (steel_LT1Carbon),
       writedevice (screen),
        mach_material (B66, steel_LT1Carbon),
        write ("B66", B66), nl,
        write ("B66", mach2), nl,
        write ("B66", mach3), nl,
        write ("B66", mach4), nl,
        write ("B66", mach5), nl,
        writedevice (myfile),
        write ("B66", B66), nl,
        write ("B66", mach2), nl,
        write ("B66", mach3), nl,
        write ("B66", mach4), nl,
        write ("B66", mach5), nl,
        closefile (myfile),
        delay (10000),
        openwrite (myfile, "digit61.one"),
        writedevice (myfile),
        write ("96"),
        closefile (myfile).


material_stainless_steel :-


        openwrite (myfile, "digit6.one"),
        material (stainless_steel),
        writedevice (screen),
        mach_material (B67, stainless_steel),
        write ("B67", B67), nl,
        write ("B67", mach2), nl,
        write ("B67", mach3), nl,
        write ("B67", mach4), nl,
        write ("B67", mach5), nl,
        writedevice (myfile),
        write ("B67", B67), nl,
        write ("B67", mach2), nl,
        write ("B67", mach3), nl,
        write ("B67", mach4), nl,
        write ("B67", mach5), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit61.one"),
        writedevice (myfile),
        write ("97"),
        closefile (myfile).
```

```
material_iron_soft :-

        openwrite (myfile, "digit6.one"),
       material (iron_soft),
       writedevice (screen),
        mach_material (B68, iron_soft),
        write ("B68", B68), nl,
        write ("B68", mach2), nl,
        write ("B68", mach3), nl,
        write ("B68", mach4), nl,
        write ("B68", mach5), nl,
        writedevice (myfile),
        write ("B68", B68), nl,
        write ("B68", mach2), nl,
        write ("B68", mach3), nl,
        write ("B68", mach4), nl,
        write ("B68", mach5), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit61.one"),
        writedevice (myfile),
        write ("98"),
        closefile (myfile).

material_iron_medium :-

        openwrite (myfile, "digit6.one"),
       material (iron_medium),
       writedevice (screen),
        mach_material (B69, iron_medium),
        write ("B69", B69), nl,
        write ("B69", mach2), nl,
        write ("B69", mach3), nl,
        write ("B69", mach4), nl,
        write ("B69", mach5), nl,
        writedevice (myfile),
        write ("B69", B69), nl,
        write ("B69", mach2), nl,
        write ("B69", mach3), nl,
        write ("B69", mach4), nl,
        write ("B69", mach5), nl,
        delay (10000),
        closefile (myfile),
        openwrite (myfile, "digit61.one"),
        writedevice (myfile),
        write ("99"),
        closefile (myfile).


material_iron_hard :-

        openwrite (myfile, "digit6.one"),
       material (iron_hard),
       writedevice (screen),
        mach_material (B691, iron_hard),
        write ("B691", mach1), nl,
        write ("B691", mach2), nl,
        write ("B691", mach3), nl,
```

```
write ('B691', mach4), nl,
write ('B691', mach5), nl,
writedevice (myfile),
write ('B691', B691), nl,
write ('B691', mach2), nl,
write ('B691', mach3), nl,
write ('B691', mach4), nl,
write ('B691', mach5), nl,
delay (10000),
closefile (myfile),
openwrite (myfile, 'digit61.one'),
writedevice (myfile),
write ('99'),
closefile (myfile).
```

Appendix B. DYNAMIC SHOP STATUS SOFTWARE

```
      CHARACTER*15 MACHINE
      INTEGER NQTY, MATL, DIGIT2, JTYPE, DIGIT3, DIGIT4, DIGIT5
      REAL Q1, Q2, Q3, Q4, Q5, QTEMP1, QTEMP2, QTEMP3, QTEMP4, QTEMP5
      REAL OUTDIA, LENGTH, SPEED, FEED, DEPTH, MACTIME
      REAL TIME, DRLTIME, HOLDEP, FINAL, LONG, DIA

       open (unit =89, file = 'digit11.one', access ='sequential',
     $ status ='old')
       open (unit =90, file = 'digit12.one', access ='sequential',
     $ status ='old')
       open (unit =91, file = 'digit21.one', access ='sequential',
     $ status ='old')
       open (unit =92, file = 'digit31.one', access ='sequential',
     $ status ='old')
       open (unit =93, file = 'digit41.one', access ='sequential',
     $ status ='old')
       open (unit =94, file = 'digit51.one', access ='sequential',
     $ status ='old')
       open (unit =95, file = 'digit61.one', access ='sequential',.
     $ status ='old')
       open (unit =96, file = 'digit71.one', access ='sequential',
     $ status ='old')
       open (unit =97, file = 'digit81.one', access ='sequential',
     $ status ='old')
       open (unit =06, file = 'plan.one', access ='sequential',
     $ status ='old')


        Q1 = 0
        Q2 = 0
        Q3 = 0
        Q4 = 0
        Q5 = 0

      READ (90, 10) MACHINE
      READ (92, 10) DIGIT3
      READ (93, 10) DIGIT4
      READ (94, 10) DIGIT5
      READ (96, 10) FINISH

      READ (95, 10) MATL
10    FORMAT (I2)
11    FORMAT (A10)
12    FORMAT (A20)
C
```

```
IF (MATL. EQ. 1) THEN
  SPEED = 280.0
  FEED  = .045
  DEPTH = .002
  WRITE (6, 15)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 2) THEN
  SPEED = 280.0
  FEED  = .045
  DEPTH = .003
  WRITE (6, 16)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 3) THEN
  DEPTH = 550.0
  FEED  = .045
  DEPTH = .003
  WRITE (6, 17)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 4) THEN
  SPEED = 110.0
  FEED  = .018
  DEPTH = .001
  WRITE (6, 18)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 5) THEN
  SPEED = 80.0
  FEED  = .018
  DEPTH = .0005
  WRITE (6, 19)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 6) THEN
  SPEED = 50.0
  FEED  = .018
  DEPTH = .0003
  WRITE (6, 20)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 7) THEN
  SPEED = 125.0
  FEED  = .025
  DEPTH = .003
  WRITE (6, 21)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 8) THEN
  SPEED = 90.0
  FEED  = .025
  DEPTH = .003
  WRITE (6, 22)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
ELSEIF (MATL. EQ. 9) THEN
  SPEED = 90.0
  FEED  = .018
  DEPTH = .0002
  WRITE (6, 23)
  WRITE (6, 400)
  WRITE (6, 14) SPEED, FEED, DEPTH
```

```
      ELSEIF (MATL. EQ. 10) THEN
         SPEED = 90.0
         FEED = .020
         DEPTH = .0001
         WRITE (6, 24)
         WRITE (6, 400)
         WRITE (6, 14) SPEED, FEED, DEPTH
      ENDIF

400    FORMAT ('THE FOLLOWING ARE THE SUGGESTED MACHINING PARAMETERS')
14     FORMAT ('SPEED IN FPM = ', F10.3,
     $ 'FEED IN IPR = ', F10.3,'FEED PER TOOTH = ', F10.3)
15     FORMAT ('THE MATERIAL OF THE PART IS COPPER')
16     FORMAT ('THE MATERIAL OF THE PART IS BRASS')
17     FORMAT ('THE MATERIAL OF THE PART IS ALUMINUM')
18     FORMAT ('THE MATERIAL OF THE PART IS STEEL_LT3_CARBON')
19     FORMAT ('THE MATERIAL OF THE PART IS STEEL_LT6_CARBON')
20     FORMAT ('THE MATERIAL OF THE PART IS STEEL_LT1_CARBON')
21     FORMAT ('THE MATERIAL OF THE PART IS STAINLESS_STEEL')
22     FORMAT ('THE MATERIAL OF THE PART IS IRON_SOFT')
23     FORMAT ('THE MATERIAL OF THE PART IS IRON_MEDIUM')
24     FORMAT ('THE MATERIAL OF THE PART IS IRON_HARD')
C
C
      WRITE (*,*) 'JOB QUANTITY'
      READ (*, 35) NQTY


      READ (89,10) JTYPE
      IF (JTYPE .EQ. 1 ) THEN
         WRITE (*,*) 'WHAT IS THE JOB LENGTH ?'
         READ (*, 36) LENGTH
         WRITE (*,*) 'WHAT IS THE JOB DIAMETER ?'
         READ (*,36) OUTDIA
         WRITE (6,37)
         WRITE (6,38) LENGTH, OUTDIA
      ELSEIF (JTYPE. EQ. 2) THEN
         WRITE (*,*) 'WHAT IS THE JOB LENGTH ?'
         READ (*, 36) LENGTH
         WRITE (*,*) 'WHAT IS THE JOB DIAMETER ?'
         READ (*,36) OUTDIA
         WRITE (6,37)
         WRITE (6,38) LENGTH, OUTDIA
      ELSEIF (JTYPE. EQ. 3) THEN
         WRITE (*,*) 'WHAT IS THE JOB LENGTH ?'
         READ (*, 36) LENGTH
         WRITE (*,*) 'WHAT IS THE JOB DIAMETER ?'
         READ (*,36) OUTDIA
         WRITE (6,37)
         WRITE (6,38) LENGTH, OUTDIA
      ELSEIF (JTYPE. EQ. 4) THEN
         WRITE (*,*) 'WHAT IS THE JOB LENGTH ?'
         READ (*, 36) LENGTH
         WRITE (*,*) 'WHAT IS THE JOB WIDTH ?'
         READ (*,36) WIDTH
         WRITE (*,*) 'WHAT IS THE JOB HEIGHT ?'
         READ (*,36) HEIGHT
         WRITE (6,37)
         WRITE (6,30) LENGTH, WIDTH, HEIGHT
      ELSEIF (JTYPE. EQ. 5) THEN
         WRITE (*,*) 'WHAT IS THE JOB LENGTH ?'
         READ (*, 36) LENGTH
         WRITE (*,*) 'WHAT IS THE JOB WIDTH ?'
```

```
        READ (*,36) WIDTH
        WRITE (*,*) 'WHAT IS THE JOB HEIGHT ?'
        READ (*,36) HEIGHT
        WRITE (6,37)
        WRITE (6,30) LENGTH, WIDTH, HEIGHT
      ELSEIF (JTYPE. EQ. 6) THEN
        WRITE (*,*) 'WHAT IS THE JOB LENGTH ?'
        READ (*, 36) LENGTH
        WRITE (*,*) 'WHAT IS THE JOB WIDTH ?'
        READ (*,36) WIDTH
        WRITE (*,*) 'WHAT IS THE JOB HEIGHT ?'
        READ (*,36) HEIGHT
        WRITE (6,37)
        WRITE (6,30) LENGTH, WIDTH, HEIGHT
      ENDIF
C
       DO 151 I7 = 1, 10
        READ (90, 10, END=1006) DIGIT1
          IF (DIGIT1. EQ. 1) THEN
            WRITE (6, 180)
          ELSEIF (DIGIT1. EQ. 2) THEN
            WRITE (6, 181)
          ELSEIF (DIGIT1. EQ. 3) THEN
            WRITE (6, 182)
          ELSEIF (DIGIT1. EQ. 4) THEN
            WRITE (6, 183)
          ELSEIF (DIGIT1. EQ. 5) THEN
            WRITE (6, 184)
          ENDIF

C
C
180     FORMAT ('USE MACHINE ONE - TURNING CENTER')
181     FORMAT ('USE MACHINE TWO - TURNING CENTER')
182     FORMAT ('USE MACHINE THREE - MACHINING CENTER')
183     FORMAT ('USE MACHINE FOUR - MACHINING CENTER')
184     FORMAT ('USE MACHINE FIVE - TURNING CENTER')
C
C
       DO 31 I1 = 1, 10
        REWIND (91)
        READ (91, 10, END=1001) DIGIT2
         IF (DIGIT2 .EQ. 21) THEN
           NRPM = (4*SPEED)/OUTDIA
           IF (NRPM .GT. 4000 ) NRPM = 4000
           MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY
           WRITE (6,37)
           WRITE (6,47) MACTIME
         ELSEIF (DIGIT2 .EQ. 22) THEN
           NRPM = (4*SPEED)/OUTDIA
           IF (NRPM .GT. 4000 ) NRPM = 4000
           MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 1.5
           WRITE (6,37)
           WRITE (6,39)
           WRITE (6,47) MACTIME
         ELSEIF (DIGIT2. EQ. 23) THEN
           NRPM = (4*SPEED)/OUTDIA
           IF (NRPM .GT. 4000 ) NRPM = 4000
           MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 1.5
           WRITE (6,37)
           WRITE (6,40)
           WRITE (6,47) MACTIME
         ELSEIF (DIGIT2. EQ. 24) THEN
```

```
      NRPM = (4*SPEED)/OUTDIA
      IF (NRPM .GT. 4000 ) NRPM = 4000
      MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 1.5
      WRITE (6,37)
      WRITE (6,41)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 25) THEN
      NRPM = (4*SPEED)/OUTDIA
      IF (NRPM .GT. 4000 ) NRPM = 4000
      MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 1.2
      WRITE (6,37)
      WRITE (6,42)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 26) THEN
      NRPM = (4*SPEED)/OUTDIA
      IF (NRPM .GT. 4000 ) NRPM = 4000
      MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 1.2
      WRITE (6,37)
      WRITE (6,43)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 27) THEN
      NRPM = (4*SPEED)/OUTDIA
      IF (NRPM .GT. 4000 ) NRPM = 4000
      MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 1.2
      WRITE (6,37)
      WRITE (6,44)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 28) THEN
      NRPM = (4*SPEED)/OUTDIA
      IF (NRPM .GT. 4000 ) NRPM = 4000
      MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 1.5
      WRITE (6,37)
      WRITE (6,45)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 29) THEN
      NRPM = (4*SPEED)/OUTDIA
      IF (NRPM .GT. 4000 ) NRPM = 4000
      MACTIME = ((LENGTH)/(FEED * NRPM)) *NQTY * 2.0
      WRITE (6,37)
      WRITE (6,46)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 30) THEN
      NRPM = (4*SPEED)/4.0
      IF (NRPM. GT. 4000) NRPM = 4000
      MACTIME = 0.0
      WRITE (6,48)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 31) THEN
      NRPM = (4*SPEED)/4.0
      IF (NRPM. GT. 4000) NRPM = 4000
      MACTIME = (NQTY)*(LENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
      WRITE (6,49)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 32) THEN
      NRPM = (4*SPEED)/4.0
      IF (NRPM. GT. 4000) NRPM = 4000
      MACTIME = (NQTY)*(LENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
      WRITE (6,50)
      WRITE (6,47) MACTIME
    ELSEIF (DIGIT2. EQ. 33) THEN
      NRPM = (4*SPEED)/4.0
      IF (NRPM. GT. 4000) NRPM = 4000
      MACTIME = (NQTY)*(LENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
```

```
        WRITE (6,51)
        WRITE (6,47) MACTIME
      ENDIF
31    CONTINUE
1001  CONTINUE

30    FORMAT ('THE LENGTH IS', F10.3, 'THE WIDTH IS', F10.3,
     $ 'THE HEIGHT IS', F10.3)
35    FORMAT (I5)
36    FORMAT (F10.3)
37    FORMAT ('THE PART IS ROTATIONAL - TURN PART')
38    FORMAT ('THE LENGTH IS', F10.3, 'THE DIAMETER IS', F10.3)
39    FORMAT ('THE PART HAS AN EXTERNAL THREAD - SINGLE
     $ POINT THREADING')
40    FORMAT ('THE PART HAS AN EXTERNAL THREAD - TAPPING USED')
41    FORMAT ('THE PART HAS AN EXTERNAL THREAD - THREADING DIE USED')
42    FORMAT ('THE PART HAS A GROOVE - TURNING USED')
43    FORMAT ('THE PART HAS A SLOT - SLOT_MILL AFTER INITIAL TURNING')
44    FORMAT ('THE PART HAS A SLOT - END_MILL AFTER INITIAL TURNING')
45    FORMAT ('THE PART IS STEPPED ONE SIDE - MULTIPLE TURNING
     $ PASSES ARE REQUIRED')
46    FORMAT ('THE PART IS STEPPED BOTH SIDES - MULTIPLE TURNING
     $ PASSES ARE REQUIRED')
47    FORMAT ('THE MACHINING TIME REQUIRED IS', F10.3)
48    FORMAT ('THE PART IS PRISMATIC - NO PLANE MACHINING')
49    FORMAT ('THE PART IS PRISMATIC - PLANE MACHINING REQUIRED -
     $ SIDE MILLING REQUIRED')
50    FORMAT ('THE PART IS PRISMATIC - PLANE MACHINING REQUIRED -
     $ FACE MILLING REQUIRED')
51    FORMAT ('THE PART IS PRISMATIC - PLANE MACHINING REQUIRED -
     $ PLAIN MILLING REQUIRED')

      WRITE (*,*) 'NUMBER OF THROUGH BORES ?'
      READ (*, 52) NBORE
      IF (NBORE. EQ. 0 ) GOTO 54
      WRITE (*,*) 'WHAT IS THE LENGTH OF BORE ?'
      READ (*,52) LONG
      WRITE (*,*) 'WHAT IS THE DIAMETER OF BORE ?'
      READ (*,52) DIA
54    CONTINUE

      DO 53 I3 = 1, 10
        REWIND (92)
        READ (92, 10, END = 1002) DIGIT3
          IF (DIGIT3. EQ. 40) THEN
            WRITE (6,61)
            MACTIME = 0.0
            WRITE (6, 47) MACTIME
          ELSEIF (DIGIT3. EQ. 41) THEN
            NRPM = (4*SPEED)/DIA
            MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
            MACTIME = MACTIME * NQTY
            WRITE (6,62)
            WRITE (6,47) MACTIME
          ELSEIF (DIGIT3. EQ. 42) THEN
            NRPM = (4*SPEED)/DIA
            MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
            MACTIME = MACTIME * 1.5 * NQTY
            WRITE (6,63)
            WRITE (6,47) MACTIME
          ELSEIF (DIGIT3. EQ. 43) THEN
            NRPM = (4*SPEED)/DIA
            MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
```

```
          MACTIME = MACTIME * 2.0 * NQTY
          WRITE (6,64)
          WRITE (6,47) MACTIME
        ELSEIF (DIGIT3. EQ. 44) THEN
          NRPM = (4*SPEED)/DIA
          MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
          MACTIME = MACTIME * 0.5 * NQTY
          WRITE (6,65)
          WRITE (6,47) MACTIME
        ELSEIF (DIGIT3. EQ. 45) THEN
          NRPM = (4*SPEED)/DIA
          MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
          MACTIME = MACTIME * 0.75 * NQTY
          WRITE (6,66)
          WRITE (6,47) MACTIME
        ELSEIF (DIGIT3. EQ. 46) THEN
          NRPM = (4*SPEED)/DIA
          MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
          MACTIME = MACTIME * 0.50 * NQTY
          WRITE (6,67)
          WRITE (6,47) MACTIME
        ELSEIF (DIGIT3. EQ. 47) THEN
          NRPM = (4*SPEED)/DIA
          MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
          MACTIME = MACTIME * 1.25 * NQTY
          WRITE (6,68)
          WRITE (6,47) MACTIME
        ELSEIF (DIGIT3. EQ. 48) THEN
          NRPM = (4*SPEED)/DIA
          MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
          MACTIME = MACTIME * 1.25 * NQTY
          WRITE (6,69)
          WRITE (6,47) MACTIME
        ELSEIF (DIGIT3. EQ. 49) THEN
          NRPM = (4*SPEED)/DIA
          MACTIME = (LONG + 0.02 + 0.02 + (0.3 * DIA))/(NRPM*FEED)
          MACTIME = MACTIME * NBORE * NQTY
          WRITE (6,70)
          WRITE (6,47) MACTIME
        ENDIF
53    CONTINUE
1002   CONTINUE

52     FORMAT (I1)
61     FORMAT (' NO INTERNAL MACHINING IS REQUIRED')
62     FORMAT (' SMOOTH INTERNAL HOLE - DRILL HOLE AND REAM')
63     FORMAT (' SMOOTH INTERNAL HOLE - STEPPED TO ONE SIDE -
   $ MULTIPLE DRILLING - AND REAMING REQUIRED ')
64     FORMAT (' SMOOTH INTERNAL HOLE - STEPPED TO BOTH SIDES - USE
   $ MULTIPLE DRILLING AND REAMING')
65     FORMAT (' SMOOTH INTERNAL SLOT - DRILL HOLE AND BORE')
66     FORMAT (' INTERNAL THREAD - USE INTERNAL TAPPING')
67     FORMAT (' INTERNAL GROOVE - DRILL THROUGH HOLE
   $ AND BORE GROOVE')
68     FORMAT (' TWO THROUGH BORES - MULTIPLE DRILL AND REAM')
69     FORMAT (' MULTIPLE THROUGH BORES - MULTIPLE DRILL AND REAM')
70     FORMAT (' MULTIPLE THROUGH BORES - NON PARALLEL - GANG DRILL
   $ AND REAM HOLES')


       DO 91 I4 = 1, 10
        REWIND (93)
         READ (93, 10, END = 1003) DIGIT4
```

```
NRPM = (4*SPEED)/4.0
IF (NRPM. GT. 4000) NRPM = 4000
 IF (DIGIT4. EQ. 61) THEN
  WRITE (6, 100)
  MACTIME = MACTIME + 0.0
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 62) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE PLANE SURFACE ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  READ (*,99) PLENGTH
  WRITE (6, 101)
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 63) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE PLANE SURFACE ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  READ (*,99) PLENGTH
  WRITE (6, 102)
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 64) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE PLANE SURFACE ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  READ (*,99) PLENGTH
  WRITE (6, 103)
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 65) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE PLANE SURFACE ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  READ (*,99) PLENGTH
  WRITE (6, 104)
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 66) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE PLANE SURFACE ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  READ (*,99) PLENGTH
  WRITE (6, 105)
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 67) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE PLANE SURFACE ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  READ (*,99) PLENGTH
  WRITE (6, 106)
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 68) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE SLOT ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  MACTIME = MACTIME * 0.25
  READ (*,99) PLENGTH
  WRITE (6, 107)
  WRITE (6, 47) MACTIME
 ELSEIF (DIGIT4. EQ. 69) THEN
  WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE SLOT ?'
  READ (*, 99) PLENGTH
  MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
  MACTIME = MACTIME * 0.25
  READ (*,99) PLENGTH
  WRITE (6, 108)
```

```
          WRITE (6, 47) MACTIME
        ELSEIF (DIGIT4. EQ. 70) THEN
          WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE GROOVE ?'
          READ (*, 99) PLENGTH
          MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
          MACTIME = MACTIME * 0.25
          READ (*,99) PLENGTH
          WRITE (6, 109)
          WRITE (6, 47) MACTIME
        ELSEIF (DIGIT4. EQ. 71) THEN
          WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE GROOVE ?'
          READ (*, 99) PLENGTH
          MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
          MACTIME = MACTIME * 0.25
          WRITE (6, 110)
          WRITE (6, 47) MACTIME
        ELSEIF (DIGIT4. EQ. 72) THEN
          WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE GROOVE ?'
          READ (*, 99) PLENGTH
          MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
          MACTIME = MACTIME * 0.25
          WRITE (6, 111)
          MACTIME = (LENGTH*WIDTH)/SPEED * 0.1 * NQTY
          WRITE (6, 47) MACTIME
        ELSEIF (DIGIT4. EQ. 73) THEN
          WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE GROOVE ?'
          READ (*, 99) PLENGTH
          MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
          MACTIME = MACTIME * 0.25
          WRITE (6, 112)
          WRITE (6, 47) MACTIME
        ELSEIF (DIGIT4. EQ. 74) THEN
          WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE SLOT ?'
          READ (*, 99) PLENGTH
          MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
          MACTIME = MACTIME * 0.25
          WRITE (6, 113)
          WRITE (6, 47) MACTIME
        ELSEIF (DIGIT4. EQ. 75) THEN
          WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE SLOT ?'
          READ (*, 99) PLENGTH
          MACTIME = (NQTY)*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
          MACTIME = MACTIME * 0.25
          WRITE (6, 114)
          WRITE (6, 47) MACTIME
        ELSEIF (DIGIT4. EQ. 76) THEN
          WRITE (*,*) 'WHAT IS THE LENGTH OF THE SURFACE SLOT ?'
          READ (*, 99) PLENGTH
          MACTIME = NQTY*(PLENGTH + 2*(4.0))/(NRPM * DEPTH * 18)
          MACTIME = MACTIME * 0.25
          WRITE (6, 115)
          WRITE (6, 47) MACTIME
      ENDIF

91    CONTINUE
1003   CONTINUE
C
C
C
99    FORMAT (F10.4)
100    FORMAT (' NO PLANE SURFACE MACHINING IS REQUIRED ')
101    FORMAT (' PLANE EXTERNAL SINGLE CURVE REQUIRED - USE PLAIN
```

```
   $ MILLING PROCEDURES TO PRODUCE CURVE')
102    FORMAT (' PLANE EXTERNAL SINGLE CURVE REQUIRED - USE FACE
   $ MILLING PROCEDURES TO PRODUCE CURVE')
103    FORMAT (' PLANE EXTERNAL SINGLE CURVE REQUIRED - USE SIDE
   $ MILLING PROCEDURES TO PRODUCE CURVE')
104    FORMAT (' PLANE EXTERNAL MULTIPLE CURVE REQUIRED - USE PLAIN
   $ MILLING PROCEDURES TO PRODUCE CURVE')
105    FORMAT (' PLANE EXTERNAL MULTIPLE CURVE REQUIRED - USE FACE
   $ MILLING PROCEDURES TO PRODUCE CURVE')
106    FORMAT (' PLANE EXTERNAL MULTIPLE CURVE REQUIRED - USE SIDE
   $ MILLING PROCEDURES TO PRODUCE CURVE')
107    FORMAT (' PLANE EXTERNAL KEYWAY - USE END MILLING PROCEDURES')
108    FORMAT (' PLANE EXTERNAL KEYWAY - USE POCKET MILLING
   $ PROCEDURES')
109    FORMAT (' PLANE SURFACE GROOVE NEEDS TO BE MACHINED - USE END
   $ MILLING PROCEDURES')
110    FORMAT (' PLANE SURFACE GROOVE NEEDS TO BE MACHINED - USE
   $ POCKET MILLING PROCEDURES')
111    FORMAT (' PLANE SURFACE GROOVE NEEDS TO BE MACHINED - USE
   $ SLITTING MILLING PROCEDURES')
112    FORMAT (' PLANE SURFACE GROOVE NEEDS TO BE MACHINED - USE
   $ SHELL MILLING PROCEDURES')
113    FORMAT (' PLANE SURFACE SLOT NEEDS TO BE MACHINED - USE
   $ POCKET MILLING PROCEDURES')
114    FORMAT (' PLANE SURFACE SLOT NEEDS TO BE MACHINED - USE
   $ SLITTING MILLING PROCEDURES')
115    FORMAT (' PLANE SURFACE SLOT NEEDS TO BE MACHINED - USE
   $ SHELL MILLING PROCEDURES')


      WRITE (*,*) 'NUMBER OF HOLES IN PART ? '
      READ (*, 117) NHOLE
      IF (NHOLE. EQ. 0) GOTO 119
      WRITE (*,*) 'DEPTH OF HOLES '
      READ (*, 118) HDEPTH
      WRITE (*,*) 'WHAT IS THE DIAMETER OF THE HOLES'
      READ (*, 118) HDIA
      NURPM = (4*HDIA)/SPEED

117    FORMAT (I1)
118    FORMAT (F10.4)
119    CONTINUE


       DO 121 I5 = 1, 10
        REWIND (94)
         READ (94, 10, END = 1004) DIGIT5
           IF (DIGIT4. EQ. 81) THEN
             WRITE (6, 130)
             MACTIME = MACTIME + 0.0
             WRITE (6, 47) MACTIME
           ELSEIF (DIGIT4. EQ. 82) THEN
             WRITE (6, 131)
             MACTIME = (NHDEPTH + 0.02 + 0.02 + (0.3 * HDIA))/
   $                 (NRPM*FEED)
             MACTIME = MACTIME * NQTY
             WRITE (6, 47) MACTIME
           ELSEIF (DIGIT4. EQ. 83) THEN
             WRITE (6, 132)
             MACTIME = (NHOLE)*(NHDEPTH+0.02+0.02+(0.3 * HDIA))/
   $                 (NRPM*FEED)
             MACTIME = MACTIME * NQTY
             WRITE (6, 47) MACTIME
```

```
          ELSEIF (DIGIT4. EQ. 84) THEN
            WRITE (6, 133)
            MACTIME = (NHDEPTH + 0.02 + 0.02 + (0.3 * HDIA))/
     $              (NRPM*FEED)
            MACTIME = MACTIME * NQTY
            WRITE (6, 47) MACTIME
          ELSEIF (DIGIT4. EQ. 85) THEN
            WRITE (6, 134)
            MACTIME = (NHOLE)*(NHDEPTH + 0.02 + 0.02 + (0.3 * HDIA))/
     $              (NRPM*FEED)
            MACTIME = MACTIME * NQTY
            WRITE (6, 47) MACTIME
          ELSEIF (DIGIT4. EQ. 86) THEN
            WRITE (6, 135)
            MACTIME = (NHOLE)*(NHDEPTH + 0.02 + 0.02 + (0.3 * HDIA))/
     $              (NRPM*FEED)
            MACTIME = MACTIME * NQTY
            WRITE (6, 47) MACTIME
          ELSEIF (DIGIT4. EQ. 87) THEN
            WRITE (6, 136)
            MACTIME = (NHOLE)*(NHDEPTH + 0.02 + 0.02 + (0.3 * HDIA))/
     $              (NRPM*FEED)
            MACTIME = MACTIME * NQTY
            MACTIME = (NLONG*SPEED) * NQTY
            WRITE (6, 47) MACTIME
        ENDIF
121     CONTINUE
1004    CONTINUE

151     CONTINUE
1006    CONTINUE
C
C
130     FORMAT ('NO HOLES IN PART - NO DRILLING REQUIRED')
131     FORMAT ('AXIAL HOLES IN PART - RELATED BY GRADUATIONS AROUND A
     $ CIRCLE - USE MULTIPLE DRILLING AND REAMING')
132     FORMAT ('AXIAL HOLES IN PART - UNRELATED BY GRADUATIONS
     $ AROUND A CIRCLE - USE GANG DRILLING  AND REAMING')
133     FORMAT ('RADIAL HOLES IN PART - RELATED BY GRADUATIONS
     $ AROUND A CIRCLE - USE MULTIPLE DRILLING AND REAMING')
134     FORMAT ('RADIAL HOLES IN PART - UNRELATED BY GRADUATIONS
     $ AROUND A CIRCLE - USE GANG DRILLING AND REAMING')
135     FORMAT ('MULTIDIRECTIONAL HOLES IN PART - UNRELATED BY
     $ GRADUATIONS AROUND A CIRCLE - USE GANG DRILLING AND REAMING')
136     FORMAT ('MULTIDIRECTIONAL HOLES IN PART - RELATED BY
     $ GRADUATIONS AROUND A CIRCLE - USE GANG DRILLING AND
     $ REAMING')

        CLOSE (90)
        CLOSE (91)
        CLOSE (92)
        CLOSE (93)
        CLOSE (94)
        CLOSE (95)
        CLOSE (96)
        CLOSE (97)
        CLOSE (06)


        STOP
        END
```

B.1.1 CUTTING INFORMATION AND MACHINING TIME COMPUTATION

B.1.1.1 Rotational Operations

Circumference of part = 3.14 * D

Speed in feet per minute = RPM * D/4

Speed in revolutions per minute = 4 * FPM/D

Feedrate in inches per minute = Feed per revolution * RPM

Time for machining = Length of cut / Feedrate

Horsepower Required = Volume removed per minute * Unit HP / Efficiency

Volume removed per minute = Feed * Depth of cut * FPM

Efficiency assumed = 75 percent

Length of cut for facing = Radius of job / Feedrate

Feed = Pitch = 1/ Threads per inch.

Time for thread = Length / Feedrate.

B.1.1.2 Drilling Operations

Distance - blind hole = Clearance + Pilot + Depth of hole.

Distance - through hole = 2 * Clearance + Pilot + Depth of hole.

Time required = Distance / Feedrate in inches per minute

Volume removed per minute = (3.14 * D * D/4) * Feed * RPM

Horsepower Required = Volume * Unit HP / Efficiency

Efficiency assumed = 75 percent

B.1.1.3 Milling Computations

Approach of cutter = (d (D - d) * 0.5)

Length of cut = ((Approach + Clearance) * 2) + Part Length

Time of machining operation = Length / Feed per minute

Assumed that face, side, and plain mills corresponding to

Widths machined are available.

Number of teeth on milling cutter = 18

Diameter of cutter = 4.00 inches

Horsepower required = (Width * Depth * Feed * Unit HP) / Efficiency

The vita has been removed from the scanned document