AN INVESTIGATION INTO THE DEVELOPMENT OF PROCESS PLANS FROM SOLID GEOMETRIC MODELING REPRESENTATION

By

HSIANG-KUAN KUNG

Bachelor of Engineering
Chung Yuan Christian College
of Science and Engineering
Chungli, Taiwan, R.O.C.
1972

Master of Science
University of Rhode Island
Kingston, Rhode Island
1978

Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
DOCTOR OF PHILOSOPHY
December, 1984

AN INVESTIGATION INTO THE DEVELOPMENT OF
PROCESS PLANS FROM SOLID GEOMETRIC
MODELING REPRESENTATION

Thesis Approved:

Thesis Adviser

Dean of the Graduate College

PREFACE

This research addresses the integration of Computer-Aided Design and Computer-Aided Process Planning. A prototype system named FREXPP (Feature Recognition and Expert Process Planning system) is presented. This system extracts and orders the form features of a part that is represented by a solid geometrical modeler and automatically selects the manufacturing process for each recognized form feature.

Chapter I describes the proposed system and the assumptions made in this research. An introduction to Artifical Intelligence and expert systems is given in Chapter II. Chapter III presents the literature review of the process planning systems. The development of FREXPP is described in Chapters IV and V. Chapter IV describes the procedure for feature recognition. Chapter V describes the expert process planning system that is based on the EXPERT system developed at Rutgers University [77]. The conclusion and the suggestion for future study are presented in Chapter VI.

I would like to take this opportunity to express my gratitude to Dr. Philip M. Wolfe for his invaluable assistance in the development of this dissertation and for his guidance and encouragement throughout my doctoral studies.

In particular, my special appreciation is due to Dr. John W. Nazemetz for his invaluable suggestion and help in this dissertation. I want to thank Dr. Gary Hansen for his expert suggestions and directions to this dissertation, and extend my sincere thanks to Dr. Wayne C. Turner and Dr. Donald W. Grace for their interest and assistance.

I express my heartfelt appreciation to my wife, Ya-Jen, and my son, Timothy, without whose understanding and encouragement I would never have attained this level of educational development. Especially, I thank my aunt, Dr. Edith G.D. Hsiung, and my parents, Mr. and Mrs. Li-Hwa Kung, for their support and encouragement.

And finally, but foremost, I would like to dedicate this work to the almighty God, my heavenly father, Who is a great source of inspiration and encouragement to me.

TABLE OF CONTENTS

Chapter Page

LIST OF TABLES

LIST OF FIGURES

Figure		Page

CHAPTER I

THE RESEARCH PROBLEM

Purpose

Process planning is one of the most important areas of production planning, especially in batch type manufacturing. It is an input to production planning and scheduling and is a major determinant of manufacturing cost. The purpose of process planning is to establish a sequence of manufacturing processes so that a quality product can be made economically according to the design data, predetermined materials, and available tools.

Planning may be performed manually or with computer assistance. This study focuses on computer-aided process planning. Recently, a new computer technique called expert computer system or expert system has been applied to several areas, such as medical diagnosis, geological mineral analysis, and electronic circuit design. Expert systems are developed to help solve important and difficult problems which usually require considerable expertise. Many engineers have begun to consider using expert systems to do some of the reasoning involved in manufacturing processes planning.

The major interests of this research are to integrate

Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) and to help process planners in making process planning decisions by providing the following:

1. A procedure for extracting the form features from the part design data of a solid geometrical modeler and converting it into the data format of an expert process planning system.

2. An expert system for generating the process plans for machined parts.

The use of expert system approach is new to the industrial area. This research demonstrates the flexibility of using this approach to link CAD and CAM.

Introduction

This section provides a brief introduction to expert systems, geometric modeling, and process planning.

Expert System

An expert system has been described as a computer system that contains knowledge about a specialized area, such as medical diagnosis, chemical structure generation, or electronic circuit design, to help solve important and complicated problems which usually require considerable expertise. There are three major components in an expert system -- a general data base, a knowledge base, and a knowledge interpreter.

A general data base describes and stores the facts of a problem. Facts can be entered by the user through an interactive process. A knowledge base stores the experts' knowledge about the specific area. A knowledge interpreter is also called a reasoning processor or a rule interpreter and is the control center of an expert system. It uses the information found in the knowledge base to manipulate the data stored in the general data base and makes decisions.

The major difference between an expert system and a conventional program is in the implementation of problem solving logic. Figure 1 depicts their differences. The problem solving logic in conventional programming is implemented as procedures. In expert systems, the problem solving logic (decision rules) is stored in the knowledge base.

The knowledge found in the knowledge base is usually procedural in nature. It tells how the data for a problem may be manipulated in order to solve a problem. The coding sequence of a procedure program is important and affects the execution of the program. The coding sequence of decision rules in the knowledge base does not affect the execution of an expert system. The execution of an expert system is heuristical. A decision rule will be executed when its condition statements or conclusion statements are set true. When the solution procedure for a problem is well understood, the conventional program is the best way to represent the knowledge. However, when a precise series of steps to solve a problem does not exist, the expert system approach is the better way. A detailed description of the expert system approach will be given in Chapter III.

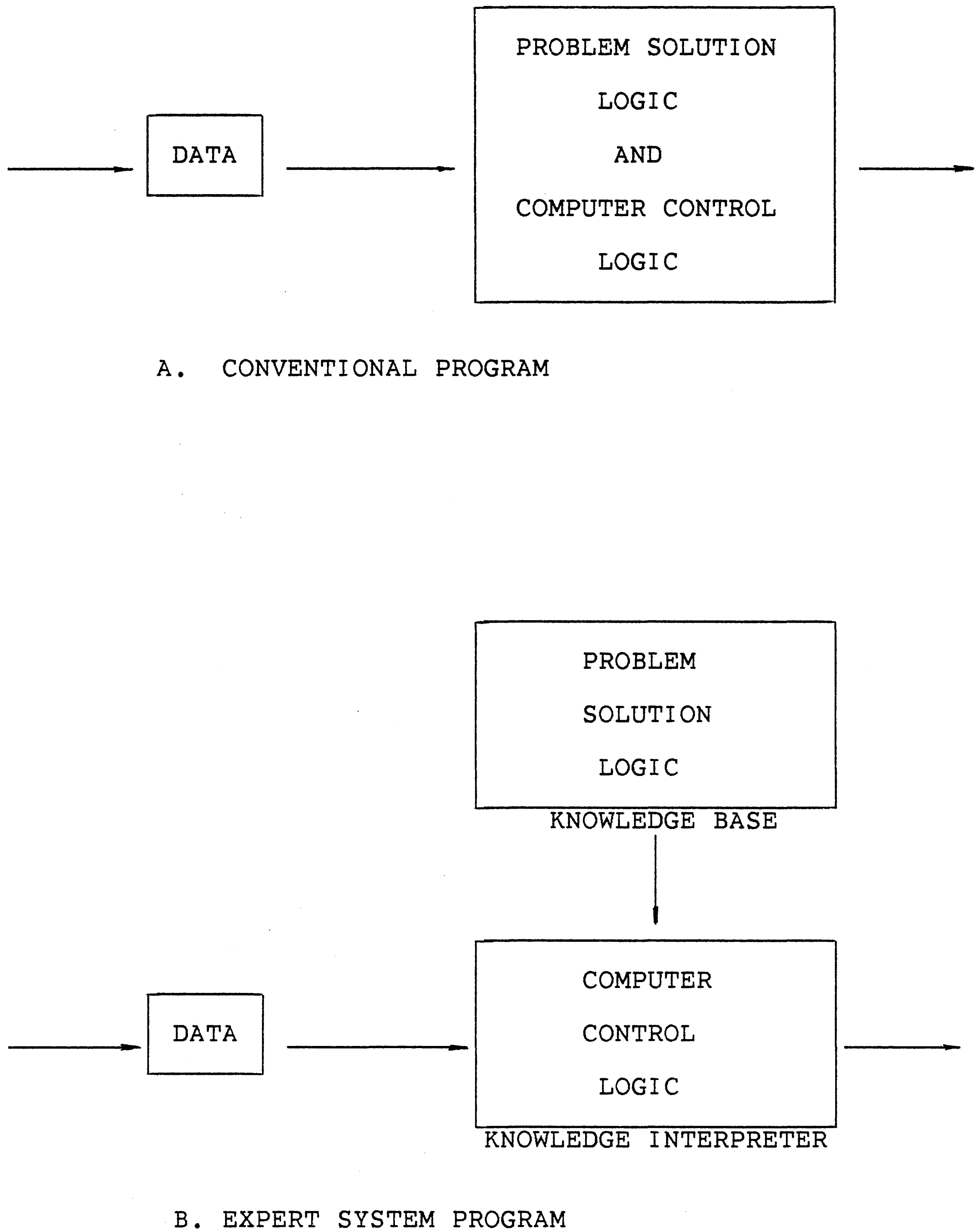


Figure 1. Comparison of Program Types

Geometric Modeling

A geometrical modeler provides a means for the designer to construct and display a geometrical model on a graphical terminal. The modeler then converts the pictorial representaton into a mathematical model and stores it in a data base for future use, such as mass analysis. Geometric modeling is an important feature in a CAD/CAM system. Many functions in a CAD/CAM system may depend on the model, such as computer-aided drafting, NC (Numerical Control) program preparing, process planning, and so on.

There are 2D models to represent two dimensional flat objects, 2-D models to represent three dimensional objects with no side-walls information, and 3D models to represent the full three dimensional objects. Researchers have put more emphasis on the development of 3D modeling than the other two systems. The representation techniques used in 3D modeling are wire frames, surface models, and solid models. Most 3D modeling is done with the wire frame technique which represents the part shape by specifying points and lines in space. Since the wire frame technique provides no information about the part surfaces, this type of system is mainly used for display purposes.

Surface modeling is the second technique used in 3D modeling systems. This technique precisely defines the outside geometry of a part. The representations are useful in NC program preparation and other tasks for which the boundary representaion is critical. Since this type model represents only a shell of the part geometry, it is not suitable for engineering analysis, such as determining weights, volume, and center of gravity.

A newly developed geometrical modeling technique is 3D solid modeling. In a solid modeling approach, the user produces a solid geometry model by sizing, adding, and subtracting geometrical solids called primitives. Primitives include spheres, circular and elliptical cylinders and cones, ellipsoids, orthogonal blocks, wedges and tori. This technique better represents the true nature of parts and assumes that most complex objects can be represented by these primitives. The representations from this model are useful in solving engineering analysis and design problems. The goal of developing 3D geometric modeling systems is to combine different representation techniques into a single system.

Process Planning

Historically, process planning has been an art rather than a science. Process planning was performed manually and heavily depended on the background and the experience of process planners. Halevi [31] made several studies in order to determine the process planner's planning strategies. He gave four process planners eight engineering drawings of different complexity and asked them to make process plans. He concluded that no two of them recommended the same process plan for any given part. In addition, he found that manual process planning revealed a variety of problems.

1. Inconsistency in routings and tooling. This inconsistency causes the increase of cost and labor requirements in a company.
2. Long turn around time. A planner usually spends several hours to several days developing a process plan, for he has to manage a great deal of information and retrieve many documents.
3. Scarcity of skilled process planners. The retirement rate is higher than the hiring rate of skilled process planners.

In order to overcome manual process planning problems, computers have been applied to the process planning area. Basically, computers are information handling machines. The application of computers to process planning can provide process planners consistent and timely information. Many computer-aided process planning systems have been developed in the attempt to reduce cost, increase productivity, and solve personnel problems.

Most of the existing computer-aided process planning systems are developed either for a certain class of part, such as rotational parts, or for a certain type of process, such as drilling or turning. These systems are very difficult to transplant, modify, or extend. Manually coded input are also required in most of these systems. Some computer-aided process planning systems use the output of a CAD system as their input, but they were designed for known manufacturing process, such as drilling and known machined surfaces, such as holes.

Recently, two process planning systems have been developed by using the Artificial Intelligence technique. GARI was developed by Descotte and Latmobe [18] and TOM (Technostructure Of Machining) was developed by Matsushima [52]. GARI is a general problem solver and is structured like an expert system. It was designed to generate process plans for machining rectangular parallelopiped parts. A specific part description model was developed for describing the machined part. This part model requires a large amount of coding work, especially when the part geometry is complicated. TOM is an expert system which is designed to generate a part program for drilling holes based on the output of a CAD system. TOM assumes that all features to be processed are holes.

It is recognized that it would be better if the computer system could automatically extract rather than manually enter the form features of a part from the geometric model. These features then could be supplied to the process planning system where a process plan could be developed using an expert system similar to those used by GARI or TOM. A form feature is a specific geometric configuration formed on the surface, or corner of a workpiece. It is designed to modify the outward appearance or to aid in achieving a given function of a workpiece [39].

The Need

Integrated CAD/CAM systems have been marked as the most significant opportunity for increasing productivity in industry today [53]. Since many production functions, such as production scheduling and planning, routing, labor requirement, and tooling, depend on process planning, computer-aided process planning becomes one of the most important features in a CAD/CAM system.

Process planning is a complicated problem. Since there are many ways to manufacture a part, process planning requires not only manufacturing data but also subjective and specialized knowledge, such as technological rules and economic considerations. The subjective and specialized knowledge is the process planning decision logic of a manufacturing company and it differs from company to company.

In order to create an integrated CAD/CAM system, an expert process planning system seems a major requirement. For in a truly integrated manufacturing system, a sub-system that can utilize the design data from a CAD system and generate the economical process plans for the machined parts is needed. The problem solving logic implemented in an expert system is not coded as procedures but as decision rules stored in the knowledge base. Different decision rules can be easily added to or amended in the knowledge base of an expert system.

The Proposed System

In this section, a prototype system FREXPP (Feature Recognition and Expert Process Planning system) is proposed. This system will extract the form features of a part that is represented by a solid modeling system. These features and related information, such as tolerances and surface finishes, would then be placed in a data base to be used by an expert system to generate a process plan. Figure 2 depicts how this system works.

Proposed System Details

The starting point for the proposed FREXPPS system is a solid geometrical modeler. Using this geometrical modeler, the engineer can design a part that is of particular interest; this part would then be represented in three dimensions. PADL-1 (Part and Assembly Description Language) [61], was chosen as the solid geometrical modeler for the proposed system. Using PADL-1, a part can be constructed using two types of primitives, blocks and cylinders. This system was chosen because it is in the public domain, and it is relatively simple to use.

The PADL-1 processor includes four sub-systems (input processor, boundary evaluator, graphical output generator, and dimension and tolerance processor) and maintains two internal representations of a defined object (a tree-structured representation and a boundary representation). The input processor uses the input data to construct a tree-structured representation of the input data. The boundary representation is derived from the input data by the boundary evaluator. It represents the object boundary as a collection of bounding "faces". The graphical output generator displays the objects on the screen. An

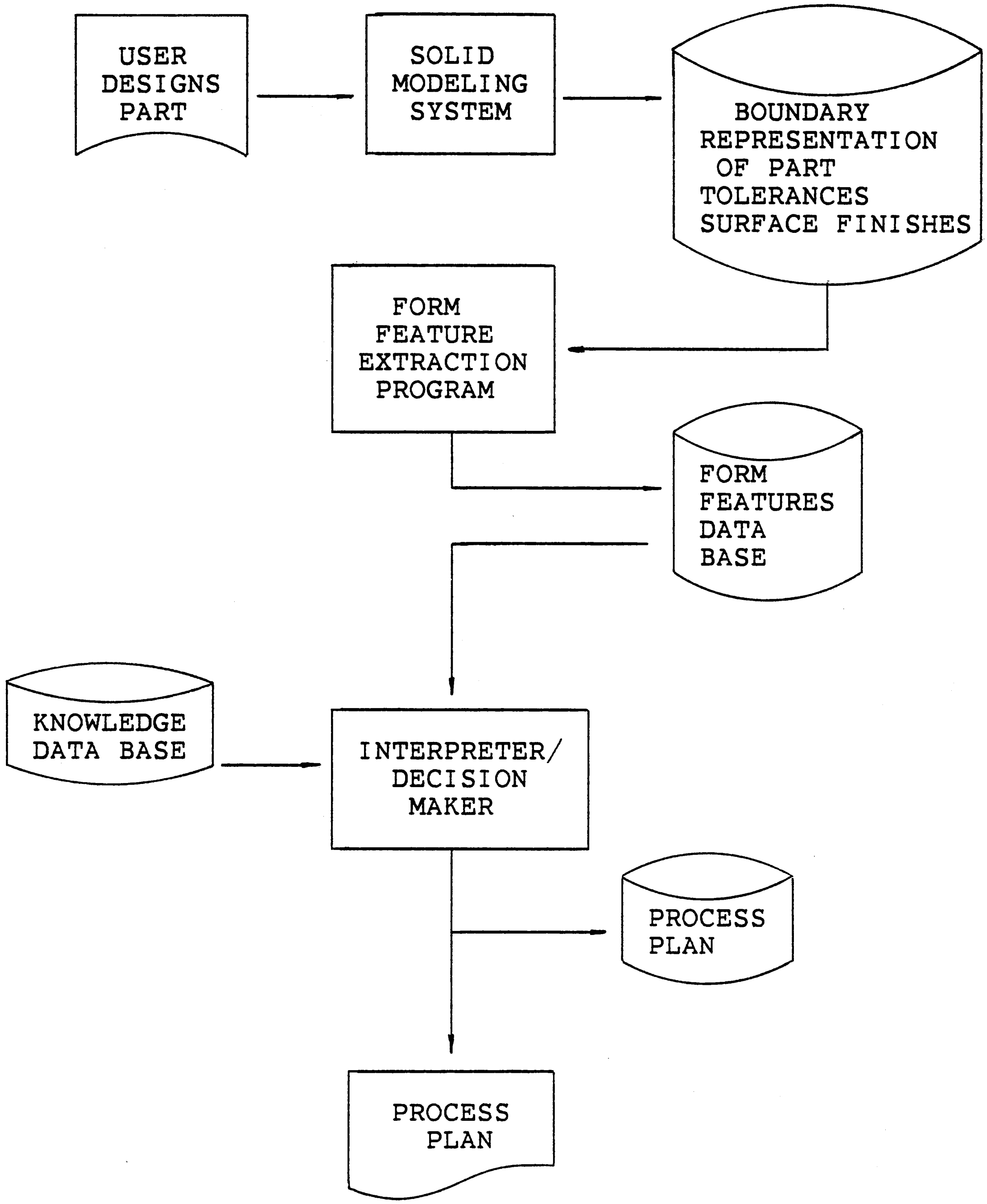


Figure 2. FREXPP System Flow Diagram

engineering drawing containing three views can also be generated and displayed on the screen. The dimension & tolerance processor is used for checking and displaying dimension, tolerance, and mechanical attribute information of a designed object.

Figure 3 contains an example of a part that was created using PADL-1; this figure also contains the instructions that were used to create the represented part. PADL-1 is limited in the types of parts that it can represent because the faces of blocks are always perpendicular to one of the coordinate system axes, and the axes of cylinders are always parallel to one of the coordinate system axes. In addition, the parts considered are limited to those that can be made from block primitives and the form features defined in a CAM-I publication called, "CAM-I's Illustrated Glossary of Work Piece Form Features" [39]. The features that the system can recognize are listed in Appendix D.

Figures 4 illustrates a part with a cylindrical feature (shaded area) that will not be recognized because the feature crosses and intersects the intersection of three block primitives B, C and D. Figure 5 illustrates a part with a non-cylindrical feature (shaded area) that will not be recognized because the feature is not defined. The proposed system will be able to generate a process plan for the part in Figure 6.

Another simplification made is that the parts are to be made from cast aluminum (356 alloy). This simplification will limit the number of rules required to define valid

```
         Size in X, Y, & Z
                          Lower-left-rear Corner Coordinates
10 &B1=$B(4.25,3.25,3.75) AT (0,0,0)      Define block B1
                                          (size & location)
20 &B2=$B(1.5,1.8,0.5) AT (0,1.45,3.75)   Define block B2

30 &B=&B1 .UN. &B2                        Perform union of
                                          blocks B1 and B2
      Diameter, Length (y-axis)
                           Bottom Center Point Coordinates
40 &C1=$CY(1.0,2.0) AT (2.5,1.5,1.5)      Define Cylinder C1
                                          (size & location)
50 &C2=$CY(0.625,4.0) AT (2.5,0,1.5)      Define cylinder C2

60 &BB=&B .DIF. &C1 .DIF. &C2             Perform difference
                                          operations (remove
                                          cylinders C1 & C2)
```

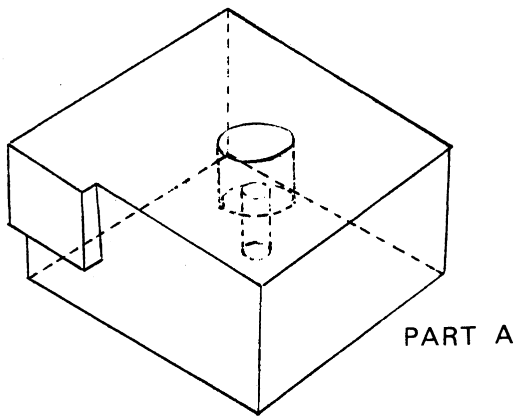


Figure 3. Part Created by Using PADL-1

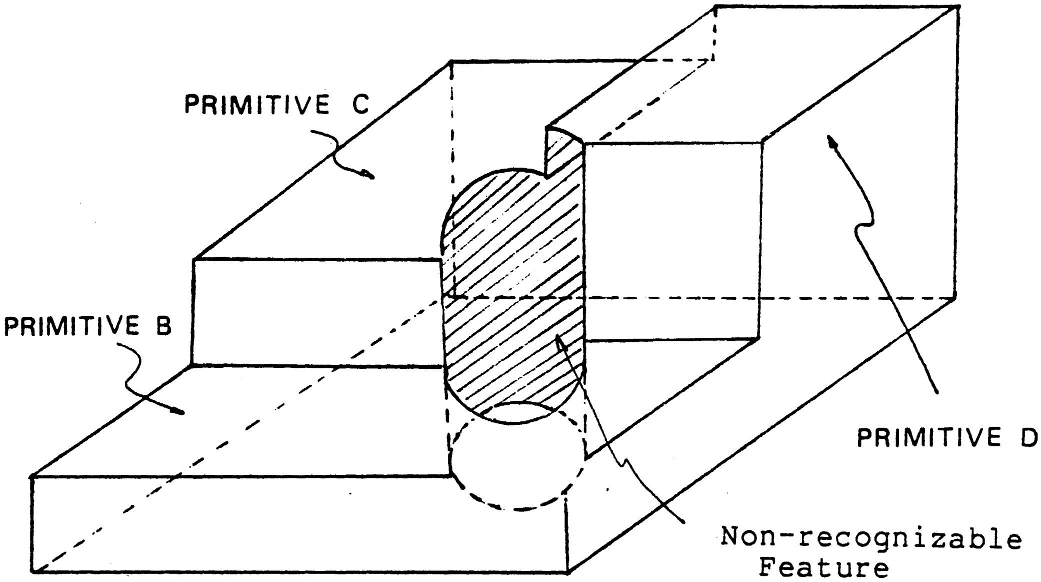


Figure 4. Part with a Non-recognizable Cylindrical Feature

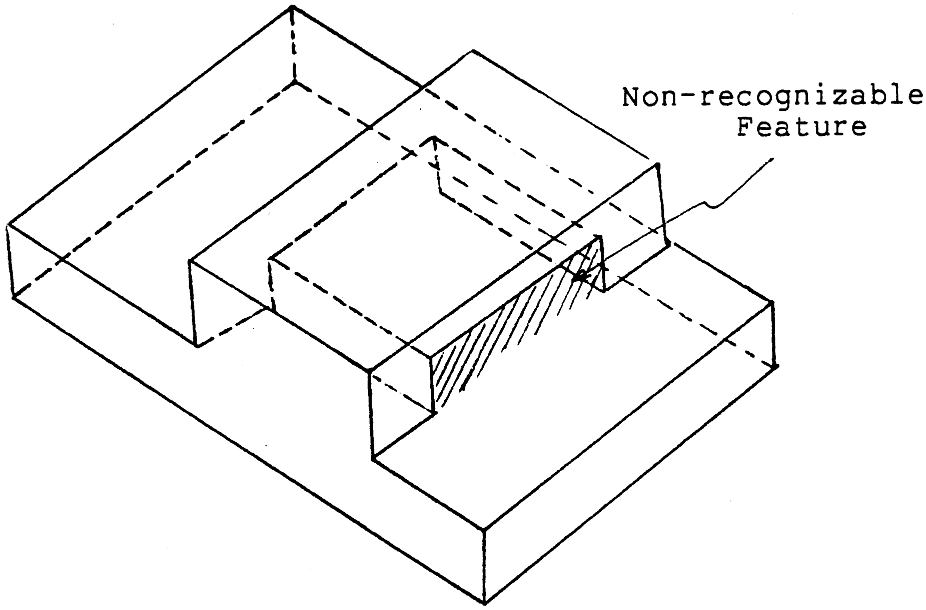


Figure 5. Part with a Non-recognizable Non-cylindrical Feature

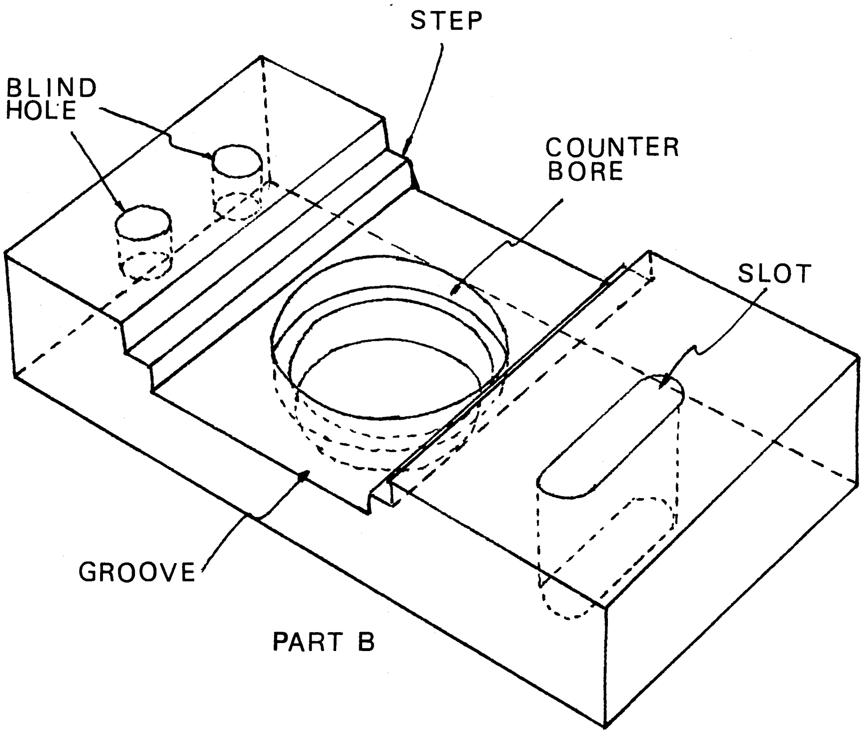


Figure 6. Features Which can be Recognized by the Proposed System

machining operations that must be developed for the knowledge data base. Another simplification involves fillets; if a fillet is required it must be made when the part is cast. This latter simplification was made because of the difficulties involved in representing a fillet in the PADL-1 system.

In the system being developed, a solid model of a part is constructed using PADL-1. The next step, is to use a set of developed FORTRAN computer programs (these programs are actually an elementary expert system) to extract the form features of a part from a boundary file created by PADL-1. The data structure of a PADL-1 boundary file is discussed in Chapter IV.

Some additional information has to be collected from the design phase, size tolerances and surface finishes. PADL-1 will accommodate size tolerance but will not accept surface finish requirements. Therefore, interactive software was developed so that a design engineer can enter the surface finish requirements. Location tolerances are not considered in the current work. As the form features are identified, this information is input to a data base which will later be used by the expert process planning system.

The process planning portion of the system was developed as an expert system. It is made up of the form feature data base, which contains form feature details, size tolerances, and surface finish requirements; a knowledge data base, which is made up of rules that are used to develop the process plan; and an interpreter/decision maker, which combines the information from the form feature data base and the knowledge base to develop a process plan.

The decision rules represented in the knowledge base are the manufacturing knowledge for the mass reduction processes. These types of processes assume that the size of the original workpiece is sufficiently large, so that the final geometry can be produced by removing material from it. The parts are assumed to be made from cast aluminum (356 alloy). Only two machining cuts, one roughing cut and one finishing cut, are needed to produce the desired surface finish. The manufacturing processes considered in this research are center drilling, drilling, boring, reaming and milling.

The expert process planning system is developed by using EXPERT developed at Rutgers University [77]. EXPERT is an expert system building tool for designing consultation type expert systems. Using the EXPERT system, the data for each problem is entered interactively. However, in this research, the expert process planning system automatically reads in the form feature data from the form feature file, selects the manufacturing process for each form feature, and generates the rough process plans for machining the part. The detailed operating parameters, such as feeds, speeds, and chucking types and methods, are not included in this system.

Summary of Assumptions and Limitations

The scope of this research is defined in the following four areas:

1. Part Material. Parts are made from cast aluminum (356 alloy).

2. Part Feature. The parts considered are limited to those that can be constructed by PADL-1. In addition, this class of parts will be reduced to those consisting of primitive blocks and form features described in Appendix D, that do not have a form feature intersecting the intersection of two primitives, such as the parts illustrated in Figures 4 and 5, and consisting of 20 or fewer features. It is assumed that the fillets must be made when the parts are cast.

3. Geometry Information. Since basic dimensioning is used for defining the locations of points and the length of lines, location tolerances are not considered in this research. The size tolerances of hole diameters are assumed to have the default values +.001 inch and -.001 inch. All the surfaces are assumed to have the value of 63AA (micro-inches) finish roughness unless otherwise specified by the designer.

4. Manufacturing Processes. Manufacturing processes to be considered are: milling, drilling, center drilling, boring and reaming. Only two machining cuts, one roughing cut and only one finishing cut are needed for each machined surface.

Process planning is a broad and complicated problem. And it was necessary to impose these assumptions to limit the scope of the study.

Summary of Research Objectives

Based on the above discussion, the primary objective of this research is:

> To develop a prototype automated process planning system that helps integrate CAD/CAM and generates the process plans for machined parts.

In order to accomplish this major objective, several other objectives are included.

1. Develop an elementary expert system that can extract the part design data from the PADL-1 system and transfer them to the expert system in a workable data format.
2. Provide a procedure that prompts the user for the surface finishing attributes for each machinable surface of a part and stores them in the general data base.
3. Represent and organize the manufacturing decision logic and store it in the knowledge base.
4. Modify the EXPERT system so that the expert system reads the form feature data automatically from the general data base and generate the process plan.

Each of the above objectives must be completed in order to achieve the primary objective. The elementary expert system is used for identifying the form features of parts and preparing the general data base for the expert process planning system. The surface finishing attributes acquisition algorithm helps to describe the machined parts. The manufacturing knowledge will be properly organized and represented in the knowledge base so that the knowledge interpreter of the EXPERT system can make deduction efficiently. Finally, the expert system will generate the process plan for the part. This generated process plan could also be used for things such as, scheduling, tool design, and preparing NC programs.

Contributions

The successful completion of this research provides benefits to both theoreticians and practitioners. This study becomes the first of its kind to provide a system that reads the geometric shape information of a part from the internal design data of a solid modeling system, then automatically generates the process plans for the recognized form features.

Process planners will benefit from this research because most of the manual operations in process planning are eliminated. The job of process planning will be less tedious and time consuming. In addition, the expert approach makes it possible to transplant and expand the basic system in different companies, because an expert system permits process planners to modify the manufacturing decision logic contained in the system. The work of this research demonstrates that the computer-aided process planning can be integrated with computer-aided design. It is a major step toward integrated computer-aided manufacturing.

CHAPTER II

ARTIFICIAL INTELLIGENCE AND EXPERT SYSTEMS

Introduction

"Artificial Intelligence", as stated in The Handbook of Artificial Intelligence [2, p. 3], "is the part of computer science concerned with designing intelligent computer systems, that is, systems that exhibit the characteristics we associate with intelligence in human behavior -- understanding language, learning, reasoning, solving problems and so on." In other words, Artificial Intelligence (A.I.) is concerned with making computers perform tasks that would require intelligence if the tasks were performed by human beings.

Two approaches have been used by A.I. researchers in the development of A.I. systems: The first approach is to use computer to mimic the same logical process as the human brain and nervous system. The second approach is to make computer suggest intelligent decisions irrespective of how the brain system works [26]. No matter which one of the two approaches is used, the ultimate goal of A.I. is to make a computer that, by its output, simply could not be distinguished from a human mind.

Artificial Intelligence is a branch of computer science. Computers are the tools of A.I. and it is the computer programs that make these tools perform tasks that people would say require the intelligence of a human being. Since the research of A.I. started in the mid 1950's, researchers have invented dozens of programs and techniques that mimic various intelligent behaviors. Today's A.I. researchers have brought these techniques from the laboratory to the real world. Various systems have been developed to help humans solve some difficult and complicated problems in chemistry, biology, geology, law, engineering, and medicine at an expert level of performance.

Artificial Intelligence is a relatively new subject in computer science and its research area has no bounds yet. At the present stage of development, specialized areas in A.I. include problem solving, natural language processing, automatic programming, expert systems and related areas, such as A.I. tools and software. In the following sections, each of these areas will be introduced.

Problem Solving

Problem solving is one of the earliest A.I. research area. In general, problem solving encompasses all of computer science because any computational task can be regarded as a problem to be solved. However, they are not all A.I. oriented. The main topics involved in the problem solving area are puzzle solving, game playing, mathematical problem solving, and automatic theorem proving. A variety of problem solving programs have been developed by A.I. researchers. Today's A.I. programs play at the champion level in checkers and backgammon, and play at the expert level in chess. There are some other programs that have been used by scientists and engineers to solve mathematical problems, such as solving algebra equations, symbolic integration, and factorization of polynomials.

Learning ability has been built into most of the game playing programs so that the playing strategies can be adjusted to a variety of environments. Computer learning techniques have been implemented either by self learning programs or by teaching programs. Self learning programs make strategy changes in response to experience in the environment. Teaching programs change the rules of the knowledge base through conversation with a human or other programs. Self learning programs have had very limited success. At the present stage, teaching is the most popular approach used in A.I. programs.

Information retrieving and problem representation are the two major tasks in problem solving. Different search strategies have been employed in the problem solving area by A.I. researchers. Trial and error search or blind search techniques play important roles in trivial problems. For a nontrivial problem, where the solution space is extremely large or the alternatives are numerous, heuristic search techniques are required. The implementation of a search technique depends on how a problem was formulated. A problem can be formulated as a state-space search problem, a problem-reduction solving problem, or as a theorem to be proved. These three approaches are introduced in the next two sections.

State Space Approach

The state space approach is a very popular problem solving representation. It formulates a problem with problem states, a set of operators, a search method of how the various states can be reached by different actions, and the specifications of a final, desired situation, or goal. A problem state is a particular configuration of a problem. An operator is a set of rules which transforms the problem from state to state. The state space of a problem is all the states that can be reached from a given initial state through a series of transformations. A solution to this type of problem can be obtained by a search process that applies operators to the initial state to produce new states, then applies operators to these new states, and so on until the goal state is produced.

For example, the initial state of an 8-puzzle is shown in Figure 7-A. An operator "move blank to the top" will transform the initial state to a new state as shown in Figure 7-B. A sequence of different operators will transform the current state 7-B to the final state in Figure 7-C.

Various search procedures have been developed for solving the state-space represented problems. Examples of such procedures are:

1. Breadth-first search, in which all paths that lead

A. INITIAL STATE

1	4	2
6		3
8	7	5

B. SECOND STATE

1		2
6	4	3
8	7	5

.
.
.

C. FINAL STATE

1	2	3
8		4
7	6	5

Figure 7. States of an 8-puzzle Tie

from one state to other states are searched at the same speed.

2. Depth-first search, in which the most recently expanded state is always searched first.
3. Heuristic search, in which various heuristic rules are used to determine which path or paths should be extended next.

The way that computers can solve large state-space problems is through heuristic search procedures. In order to do the searching efficiently, the selection of a particular data structure to represent the state of a problem is important. A variety of ways can be used to represent the state of a problem, such as symbol strings, vectors, arrays, trees, and lists. The selection of a particular data structure depends on the size and the complexity of the problem.

Problem Reduction Approach

When a problem is too large, the problem solvers usually segregate the problem into several small portions. Using the problem reduction approach, an analysis is made of the original problem, then an operator is employed to transform the original problem to a set of sub-problems. The new set of sub-problems is simpler and easier to solve than the original problem. Solutions to the sub-problems imply solutions to the original problem. For example, consider the problem of driving a car from Stillwater, Oklahoma, to Dallas, Texas. This problem could be reduced to sub-problems:

1. Drive from Stillwater to Oklahoma City, and
2. Drive from Oklahoma City to Dallas.

Here a solution to these two sub-problems would produce a solution to the original problem.

For any given problem, there may be many reduction operators that are applicable to generate the sub-problems. However, some of the generated subproblems may not be solvable. To avoid the occurence of this situation, a search procedure is required to detect this type of problem. As stated in the state-space approach, several search procedures have been developed and various data structures are available for representing problems.

Theorem Proving

Mathematical problems often require some sort of proof or logical analysis instead of simply finding solutions for them. In order to do automatic logic reasoning, a formal language is needed to describe the problems and make valid logical deductions. First order predicate calculus has been used by most of A.I. researchers to represent problems in developing the automatic theorem proving techniques. It is a system of logic which can express mathematical statements. For example,

$$(\forall x)(\forall y)\{[G(x,0) \wedge G(y,0)] \Longrightarrow G(TIMES(x,y),0)\}$$

This statement says that for all x and y, if x is greater than 0 and y is also greater than 0, so is the product of x and y. In addition to formulating mathematical problems, non-mathematical problems can also be formulated by the first order predicate calculus. In particular, theorem proving techniques can be used in information retrieval systems where deductions must be made on a data base of facts in order to answer a query. For example,

HEAD(COMP_CNTR,DR. BUMM)

WORK_IN(COMP_CNTR,MR. MAGEE)

{[WORK_IN(x,y)∧HEAD(x,z)] ==> BOSS_OF(y,z)}

These three statements express the facts that Dr. Bumm is the head of computer center, Mr. Magee works in the computer center, and z is the boss of y if y works in x and z is the head of x. An intelligent retrieval system might be expected to answer a query like "Who is Mr. Magee's boss ?". This query might be stated as the following theorem to be proved.

(∃x)BOSS_OF(MR. MAGEE,x)

A proof that an x exists would provide an answer to the query. The theorem proving procedure is based on the resolution principle which is extensively discussed in Nilsson's book [55]. Many programs that can prove assertions in first order predicate calculus form have been developed [56].

Natural Language Understanding

The most convenient way for people to deal with a computer is to use their natural languages that are the languages that living creatures use for communication, such as English, Chinese, etc.. That is why the area of natural language understanding has been intensively studied by A.I. researchers. Researchers are trying to build machines that can understand natural languages. Reading machines are one of the practical things that came out from this area at study. Hundreds of reading machines have been built to help blind and handicapped people. Among all the developed reading machines, Kurzweil Reading machines are the most advanced machines [38]. These machines can recognize 300 fonts of each alphabetic letter in both upper and lower case and convert them into spoken English.

Pattern recognition techniques have been heavily used in research into natural language understanding. It is a fundamental technique. Pattern recognition includes pattern classification and pattern matching. A pattern is defined as a collection of objects and each of the objects has the properties that satisfy certain criteria known as pattern rules [42]. For example, the pattern rules for the letter "A" described in Kurzweil Reading machines are "The capital letter A has a concave area at the base and a loop at the top. The top is a completely closed area of white with extensions at the west side and east side." [38, p. 89].

Pattern classification means that given an object and a collection of pattern rules, determine which subset of the pattern rules are satisfied by the object. For example, when a letter is read into the reading system, the rules will be used to identify what the letter is. Pattern matching means that given a pattern rule and a collection of objects, find which of those objects satisfy the pattern rule. For example, in order to find all of the letter "A" from the given letters, all the letters are compared with the rules for the letter "A".

Pattern recognition techniques have been used in developing the reading machines. Letters of a word to be pronounced are processed one at a time. Once all the letters of a word has been recognized, the reading machine is ready to pronounce the word. Kurzweil machines process pronunciation and articulation of words by synthesizing these words from single letters according to 1000 rules and 1500 exceptions to rules that apply to English.

Interpreting a language and translating a language to another one is also an important area in A.I. natural language research. In the early A.I. research, this interpretation and translation was done by word for word substitution using a number of rules dealing with grammar. Only 80 percent of the translations were satisfactory, and modification were required to enable comprehension for the other 20 percent. Languages are filled with expressions based on special meaning. Understanding a language involves knowledge and reasoning about the nature of the world.

The new approach of natural language research is to build the world knowledge and information inference system into the computer system. For example, in order to make the computer correctly interpret the sentence "The policeman stopped the car with his hand". The world knowledge about a policeman must be built into the computer system. With the help of an inference system, the computer will conclude that it is the authority of the policeman that stopped the car and not the power of the policeman's hand. At the present stage, many story interpreting systems have been successfully developed by using this new approach, such as SAM (Scripture Applier Mechanism), FRUMP (Fast Reading Understanding and Memory Program), and PAM (Plan Applier Mechanism) [63].

Listening is the hardest part in the natural language research. At the present stage, listening computers can handle only limited vocabularies. Some of the applications are airline reservation systems, telephone directories, and commands for robots.

Automatic Programming

The goal of A.I. in automatic programming is to build computer information systems which will yield good programming solutions from the description of a problem either in a formal language, such as the first order predicate calculus, or in a natural language, such as English. The basic tools used in developing automatic programming systems are automatic theorem proving, pattern recognition and a deduction system. A deduction system is made up of many rules expressed as small functions or programs. At the present stage of development, only a few examples have been worked out in the automatic programing area [44]. Compilers for high level computer languages are the early results of automatic programming research.

Intelligent Robots

Robots are generally described as creatures or machines that function under their own power and control. Artificial Intelligence researchers involved in this area have looked at everything from optimal movements of robot arms to methods of planning a sequence of actions to achieve a goal. Thousands of robots have been implemented in assembly lines. They can be programmed to perform a variety of jobs. However, those robots can only work on parts in fixed positions. When parts are placed in different positions, these types of robots are not able to adjust their positions to complete the job as a human would.

Robots with vision have been called intelligent robots or second generation robots. The second generation robots can see through a TV camera and can respond to the environment. Analog signals that come from the TV camera are converted to digital information and stored in the computer memory. Then, through the pattern recognition process, the input information is compared with the image of objects that have been stored in the memory, the computer can recognize the correct objects. For example, robots with vision have been applied to jobs in quality control. Autovision II is a system developed by Automatrix, Inc.. It has been used to inspect, identify, count, sort, position and orient parts. It also rejects defective parts [38]. Robot with vision is one of the most popular A.I. research today. The result of the research will largely affect the industry and the way people live.

Expert Systems

Early work in A.I. was aimed at developing general problem-solving systems. Several such systems were successfully developed in handling small problems. However, these systems failed when they were faced with large and complicated problems. Eventually, it was realized that human beings solve real world problems by using their knowledge and experience rather than alogrithmic solutions. This realization led to the development of the "expert systems" -- systems that make use of large amounts of knowledge about a specific subject. Each expert system encompasses a quantity of knowledge to help people solve the important and difficult problems which usually require a decision made by an expert.

Expert systems differ from conventional computer programs in two ways:

1. Programming structures are different. Figure 1 (page 4) depicts the different programming structures of these two types of systems. An expert system basically contains three components: a general data base (data), a knowledge base (the solution logic), and the inference engine (computer control logic). The solution logic and

computer control logic are implemented in the conventional program. When a decision has to be made, a "decision tree" is the basic approach used in the conventional program. For a new problem or a change, this approach requires that the entire process be analyzed in advance, then coded into a data structure. However, when the decision rules for expert systems are stored in a knowledge base, the knowledge base can be modified independently without affecting the entire process.

2. Problems to be solved are different. Conventional programming techniques are most effectively applied to problems of a repetitive or algorithmic nature. The knowledge for solving this type of problem is firm, fixed and formalized. However, when (i) the knowledge for solving a problem is subjective and judgemental, (ii) the precise steps for solving problems do not exist, the expert system approach is the better way. Besides, expert systems are particularly useful in situations where expertise is not available on a continuing basis. Data processing techniques for conventional programming are basically designed for increasing the productivity of clerical work. Expert systems are designed for helping the managerial and executive tasks.

Knowledge Engineering

Expert systems are also called knowledge based systems, because they utilize the facts and heuristics which real experts employ for solving problems. The facts are the body of information that is widely shared and publicly available, such as teachings in books. The heuristics are mostly private rules of thumb, rules of plausible reasoning, rules of good judgement, or rules of good guessing that enable an expert to make good decisions in his field. An expert is often unaware of how he comes to his conclusions and cannot give hard and fast rules. Thus, a great deal of interaction is required between A.I. scientists and the experts before making usable knowledge rules for expert systems. The work of incorporating and converting the human expert's knowledge and experience into expert systems has been delegated to the knowledge engineers.

The activity of knowledge engineering can be defined as follows: "The knowledge engineer practices the art of bringing the principles and tools of A.I. research to bear on difficult applications problems requiring experts' knowledge for their solution. The technical issues of acquiring this knowledge, representing it and using it appropriately to construct and explain lines-of-reasoning, are important problems in the design of knowledge-based systems .... The art of constructing intelligent agents is both part of, and an extension of, the programming art. It is the art of building complex computer programs that represent and reason with knowledge of the world [33, p. 89]."

Knowledge acquisition, representation and utilization are the most important work in knowledge engineering. The expert's knowledge provides the key to expert performance, while knowledge representation and inference schemes provide the mechanisms for its use.

The Components of Expert Systems

General Data base. A general data base describes and stores the facts of a problem. Facts are the inputs to an expert system. However, many developed expert systems prompt the users to enter the facts of a problem rather than describing the facts in a data base.

Knowledge Base. A knowledge base stores the knowledge about a specific area. It contains three items: parameters, rules, and confidence levels. Parameters are variables that are subject to changes during the execution of a program. Rules are the representations of the experts' decision logic. A confidence level states the degree of confidence in a rule when it is used. The confidence level implies that a rule may not be universally applicable.

The first important work in constructing a knowledge base is developing a method to represent the expert knowledge. Different methods have been used to represent knowledge. IF-THEN rules are the most popular method used in designing expert systems. IF-THEN rules are also called situation-action rules or production rules. All of the IF-THEN rules representation have the following form:

IF (condition 1 is true) and
   (condition 2 is true) and

.

.
.

(condition m is true)

THEN (action 1)

(action 2)

.
.
.

(action n)

This rule simply means that if a certain kind of situation arises, a certain kind of action can be taken. The conditional part can be thought of as patterns to be matched against the facts in the data base. If all the conditions of a rule are matched, the actions will be performed. For example, the economical way to make a hole in a sheet of metal can be EDM (Electrical Discharge Machining), if the thickness of the sheet metal is less than .375", the hole diameter is between 0.62" and 3.00", the hole interior finish is greater than 63 AA, and both the maximum and minimum tolerances of the location and the hole size are +0.002" and -0.002" respectively. This knowledge can be expressed in a rule as shown in the Table I.

The most recently developed A.I. knowledge representation scheme is the "Frame" [2]. This technique is still in its early stage. Basically, a frame is a data structure that includes declarative and procedural information in predefined internal relations. Thus, a generic frame for a person might have knowledge slots for facts that are typically known about a person, like the

TABLE I

A DECISION RULE

```
Rule 2.
  IF (hole type is equal to 1)
     (material thickness is not less than .375")
     (hole diameter is between .062" and 3.00")
     (interior finish is greater than 63 AA)
     (hole size tolerance is greater than +.002")
     (location tolerance is greater than +.002")
THEN (the hole will be produced by EDM technique)
```

name, sex, class, major, and an "attached procedure" for finding out what the class is if it is not known. A college student frame might look like this:

Generic College Student Frame

Name: a proper name

Sex : male or female

Class: freshman, sophomore, junior, or senior

Major: a department (If-Needed: find a department with Name=student name)

OSU-STUDENT Frame

Name: Jim Chern

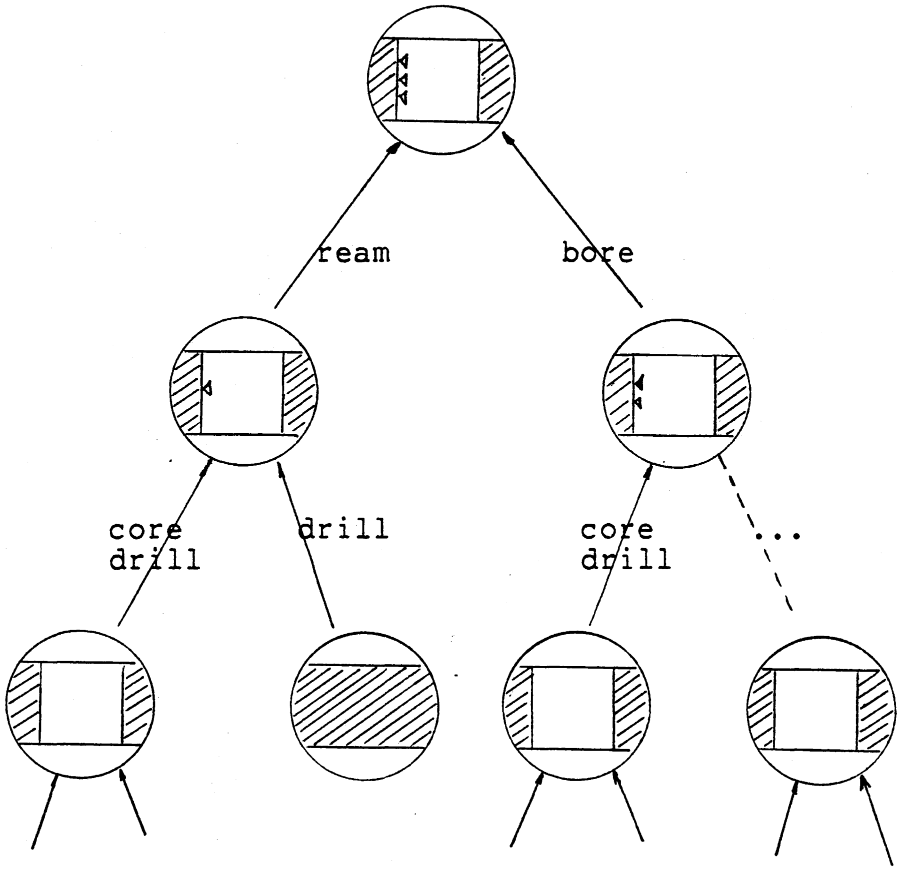
Sex : male

Class: junior

Major: computer science

There are some other representation methods, such as semantic nets for natural language, and logic for problem solving.

Another important task in constructing a knowledge base is building the relations between decision rules so that inference can be automatically deducted. For example, a high surface finish hole can be produced by reaming or boring. These two processes can be used only when the hole already exists with a lesser degree of a surface finish. Again, this pre-existing hole can be produced in a variety of ways, such as drilling, core drilling, etc. Figure 8 illustrates these process relations graphically.



Source: Matsushima, K., N. Okada, T. Sata. "The Integration of CAD and CAM by Application of Artificial Intelligence Techniques." <u>CAM-I Special Projects</u>, 1983.

Figure 8. Hole Process Flow Diagram

<u>Knowledge Interpreter</u>. A knowledge interpreter is also called an inference engine, a reasoning processor or a rule interpreter. It is the control center of an expert system. Two kinds of control strategies have been developed to find the enabled rules and make decisions on which rules should be applied. They are either forward chaining or backward chaining strategies.

Forward chaining means working from facts to conclusions. It starts with a collection of facts and tries all available rules over and over, adding new facts as it goes, until either a goal state is reached or no more applicable rules are found. The forward chaining problem solver looks for rules that depend only on already known facts. For example, if all of the condition statements in the Rule 2 shown in Table I (page 39) are known, then the rule interpreter notes the rule. The problem solver concludes that the hole will be processed by the EDM technique. Henceforward, this fact is added to the data base; it can help trigger other rules.

Backward chaining means working from a goal to facts. It starts with a goal and tries to achieve it. The strategy involves finding rules that demonstrate the goal and then verifying the facts that enable the rule to work. For example, if the rule interpreter is trying to identify that a hole will be processed by the EDM method, it notes Rule 2 as stated in Table I. Using this rule, the problem solver observes that a hole will be processed by the EDM method, if all the conditions are matched. The first condition "hole type equal to one" of the rule becomes a new goal to be achieved, and the same procedure is applied recursively. If there is no rule which can be used to establish the new goal, the rule interpreter will ask the user for the necessary facts and enter them to the data base.

Existing Expert Systems

During the last two decades, a number of A.I. systems have been developed that specialize in several areas, such as medical diagnosis, chemical structure generation, mineral exploration, and electronic circuit design. A famous expert system is DENDRAL [48]. which analyzes mass spectrogram and nuclear magnetic resonance to infer the chemical structures of an unknown compound. For some families of molecules, it operates more accurately and quickly than the best human mass-spectrum analysts.

MACSYMA, developed at MIT [50], incorporates hundreds of rules generated from experts in applied mathematics. The users command MACSYMA to perform various operations on equations and expressions, such as differentiation and integration. MACSYMA surpasses most human experts in this area.

There have been several expert systems developed in the area of medical diagnosis and treatment. MYCIN [64], one of the earliest and best known expert systems, is a consultation system to assist physicians to diagnose bacterial infections and suggest therapy. PUFF [25] is a similar system for diagnosing pulmonary function disorders, given case histories and results of various lab tests. Output from each system has demonstrated a very high degree of agreement with the real doctors.

Some systems have also been developed for helping industrial firms, such as R1 [53] for configuring customer requests for VAX computer systems at the Digital Equipment Coorporation, and PROSPECTOR [19], an interactive aid for geologists involved in mineral exploration. PROSPECTOR has discovered a molybdenum deposit whose ultimate value will probably exceed a hundred million dollars.

Hayes-Ruth et al. [32] have published an excellent survey on existing expert systems. They have classified existing expert systems, into different types according to the application area. Table II summarizes this classification. Interpretation systems infer situation descriptions from sensor data. Prediction systems infer likely consequences of given situations. Diagnosis systems infer system malfunction from observables. Design systems configure objects under constraints. Planning systems design actions. Monitoring systems compare observations of system behavior to plan features. Debugging systems create specifications and recommendations for identifying and correcting malfunctions. Instruction systems diagnose, debug, and modify students behavior. Control systems interpret, predict, repair and monitor system behaviors.

Summary

Artificial Intelligence is actually a methodology that

TABLE II

APPLICATIONS OF EXPERT SYSTEMS

Category	Application area
Interpretation:	surveillance, speech understanding, image analysis, signal interpretation.
Prediction:	weather forecasting, demographic predictions, traffic predictions, crop estimations, and military forcasting.
Diagnosis:	medical, electronic, mechanical, and software diagnosis.
Design:	circuit layout, building design, and budgeting.
Planning:	automatic programming, robot, project, route, communication, experiment, and military planning problems.
Monitoring:	nuclear power plant, air traffic, disease, and regulatory.
Debugging:	computer, network, and computer maintenance.
Instruction:	student's behavior, air traffic control.
Control:	business management, battle management, and mission control.

Source: F. Hayes-Roth, D. A. Waterman, D. B. Lenat, Building Expert Systems. Addison-Wesley Publishing Company, Inc., 1983.

can be applied to many fields. This chapter presents a brief introduction to A.I. based on the application areas of problem solving, natural language, automatic programming, intelligent robots, and expert systems. Although different areas are classified in A.I., they are not all independent. For example, in order to solve a problem, a natural language may be used to communicate with a computer, then the computer may invoke the information processing system to understand the problem. Once the problem is understood, the computer may invoke the proper problem solving technique to solve the problem.

The fundamental aspects of A.I. that underlie these applications are problem representation, knowledge (data) representation, search technique, pattern recognition, reasoning process, and learning ability. In this research, the expert system approach was chosen to develop a general process planning system.

CHAPTER III

LITERATURE REVIEW

Introduction

In this research, the primarily focus is on batch-type manufacturing; therefore, processing is defined as a series of operations for shaping raw materials into designed forms. "Planning" means to formulate a program to accomplish or achieve an objective. It implies that a strategy, such as maximizing the profits, minimizing the processing time, etc., is applied to accomplish a goal. The purpose of process planning is to establish a sequence of manufacturing processes so that a product can be made economically according to its design, predetermined materials, and available tools. The functions like shop planning and scheduling, methods and work standards, tool design, purchasing are based on the information provided by a process plan.

Manual Process Planning

Process planning is still performed manually in most firms. It depends heavily on the experience and the background of the process planner. CAM-I European members [7] have interviewed a large number of process planners to find out how they work and think. The following is a summary of their studies in discovering how process planning may be done. Manual process planning can be described in four stages:

1. Recognizing the dominating characteristics. The process planner starts by judging the entire manufacturing task. He would recognize the dominating characteristics of this task from the past experience of knowing previous products that had the same characteristics.
2. Finding the process steps of the plan. According to the characteristics found in the first stage, a process planner is able to look up a previous plan and extract what he needs and adds what is missing for the current task.
3. Checking the feasibility of the plan. At this stage, the process planner will check the designed plan against the product requirements, time, and cost. If the process steps have not been fully detailed, the planner repeats the first stage for the incomplete manufacturing task.
4. Issuing the final process plan.

Chang [11] itemizes the elements in the process planning function as follows:

1. Machining surface identification,
2. Determination of the process to be used for each of the machined surfaces,
3. Operation sequencing,

4. Machines selection,
5. Tools selection,
6. Fixture and work holding method selection,
7. Machining parameter selection (feed, speed, etc.),
8. Cutter path determination,
9. Inspection method and equipment selection, and
10. Machining time and cost estimation.

However, the elements of the process planning function vary from company to company. At one extreme, only the rough routing is planned. At the other extreme, all the elements stated above are planned. Manual process planning has revealed many problems, such as, long turn around time, inconsistent routings and tooling, and the scarcity of skilled process planners.

Process planning should be considered during a product design. The FREXPP system is an interactive process planning system. The ultimate goal of this system allows the user to design the part, do the finite analysis, select material, and enter different surface finish requirements so that various process plans can be generated and an economical plan can be decided automatically.

Computer-Aided Process Planning

Computer-Aided Process Planning (CAPP) is also known as Automated Process Planning. It was mentioned as early as 1965 [81]. Since the idea of automated process planning was first proposed, there has been a growing interest in the development of computerized process planning systems in

Europe, Japan, and in the United States. A number of process planning systems have been developed to reduce cost and increase productivity in different industries. In general, there are two types of computer-aided process planning systems -- variant and generative. Spur was perhaps the first one to define them and discuss the difference of their implementation [81]. Variant type systems and generative type systems are introduced in the next two sections separately.

Variant Type Systems

The variant type system logic is similar to manual process planning. Parts in the variant type system are segregated into families. Part families are grouped according to the similarity of design attributes, such as the geometric shape and size of parts, or manufacturing attributes, such as the sequence of processing steps required, or both of these attributes. A coding system is always associated with the classification system to distinguish each part family. In addition to the grouping of part families, a set of standard process plans is required for each part family in a variant type system. Standard process plans were established by the experienced process planners based on the common characteristics of each part family and stored in the computer memory. These standard process plans will be retrieved and modified later for a new or revised machined part.

Both the part family classification code and part number are used to relate a part to a set of process plans in a variant type system. The grouping method of part family is based on the concept of Group Technology. "Group Technology," stated by Groover [30, p. 538], " is a manufacturing philosophy in which similar parts are identified and grouped together to take advantage of their similarities in manufacturing and design." It is aimed at increasing the productivity in manufacturing the small quantity job.

The typical variant type system is CAM-I's CAPP system [9]. It was developed primarily to demonstrate the feasibility of a computer aided process planning system. The logic is based on the Group Technology method of classification and part coding. Figure 9 illustrates the information flow diagram of the CAPP system. To generate a process plan, the first input to the CAPP's system is a part family number. Then, the header information, such as the part classification code, the part number, the design date, the name of the designer, etc., are required to be updated. After updating the header information, the user is asked to modify the sequence of operations and the detail elements of each operation sequentially. The completed information is then stored in the process planning file. The hard copy of a stored process plan is obtained by using the process plan formatter that retrieves the information from the process planning file. Several versions of CAM-I's CAPP system have been developed and are currently used by some manufacturing oriented companies.

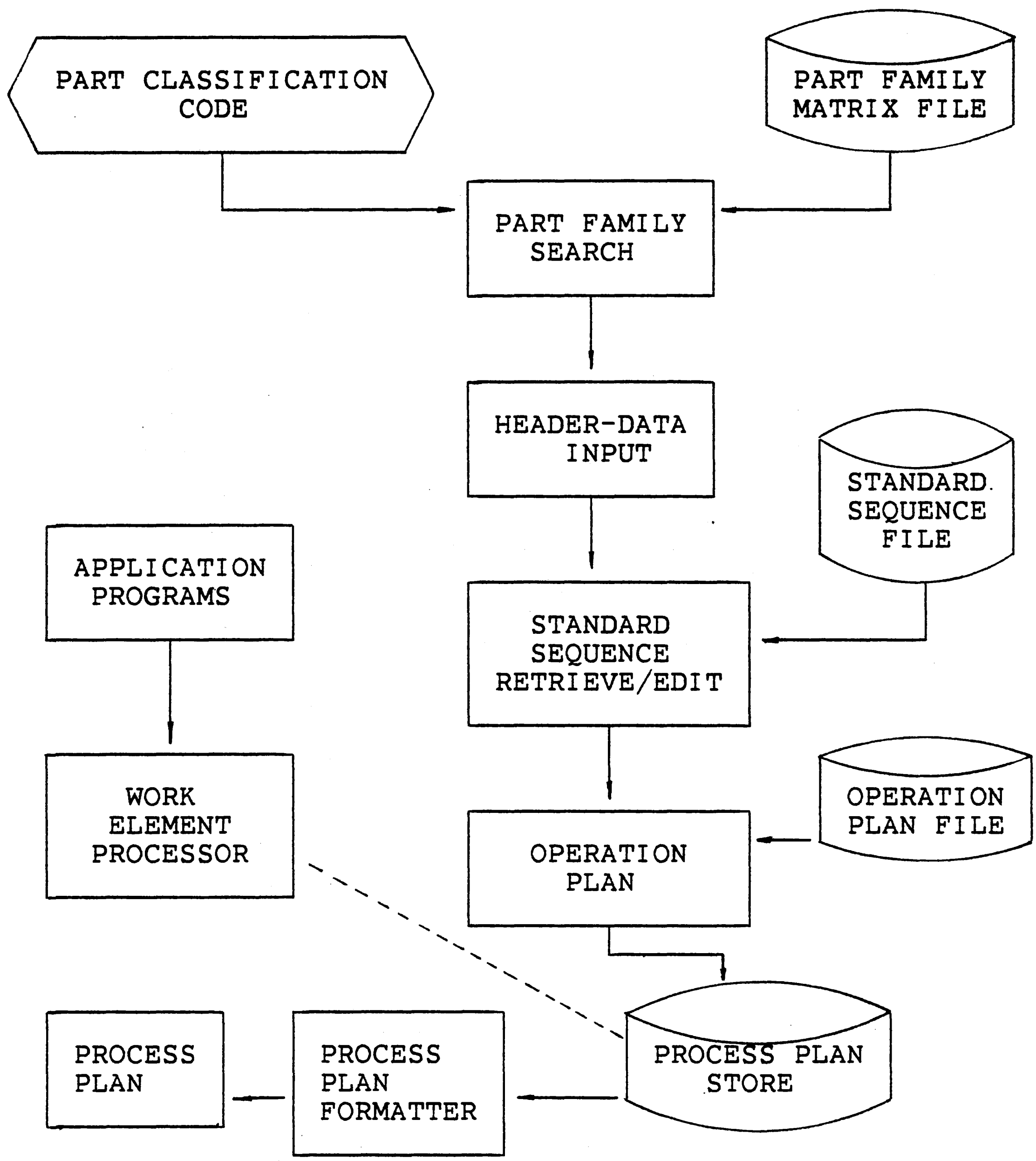


Figure 9. CAM-I's CAPP System Flow Diagram

The MIPLAN process planning system is based on the MICLASS classification and coding system and was developed by the Organization for Industrial Research Inc.. It is an interactive system and has flexible-on-line editing capability. Since the MICLASS system can generate the classification code for each part through the question-answer procedure, the up-front work of classification is not necessary. MIPLAN provides the users with four different options to create the process plan: (i) a plan can be created from the scratch, (ii) an incomplete plan can be retrieved from the computer, (iii) a plan can be retrieved by entering the part number, and (iv) a process can be retrieved through the part classification code for the same part or similar ones.

Generative Type Systems

The concept of a generative type process planning system is to use the computer to create process plans from information which is available in a manufacturing data base. A manufacturing data base contains the part description data and technological information, such as machining data and tooling information. A generative process planning system consists of a manufacturing data base and the manufacturing process decision logic program to manipulate the data in the data base. Preliminary works, such as classification, coding, and establishing standard processes are not required.

Different degrees of generative process planning systems were first proposed by Scheck [60]. Four groups of generative process planning systems are classified as shown in Figure 10. Figure 10-A illustrates the first class or the truly generative process planning system. A truly generative process planning system will scan and interpret the part description data which is stored in the data base, then automatically and properly fit these data to the requirements of the manufacturing decision logic to generate an optimal process plan. This class of system would be universally applicable. It means that if any part is presented to this kind of system, the computer will produce an optimal process plan for this part. However, the truly generative process planning system does not exist yet.

Figure 10-B shows the second class of generative process planning system in which a human coding from the engineering drawing data is required. Only one system, the Experimental Planning System (XPS), claims to be in this class. XPS is a generative process planning system which is currently under development by the contractor of CAM-I Inc. [15]. XPS will be able to execute an individual company's process planning logic to produce a sequence of work elements for manufacturing a part. This system will execute the decision tables that contain the logic for selecting and sequencing the work elements. The part description data is entered through a query system and is compatible with the CAM-I's variant CAPP system.

Figure 10-C illustrates the third kind of generative process planning system in which a CAD system is used to

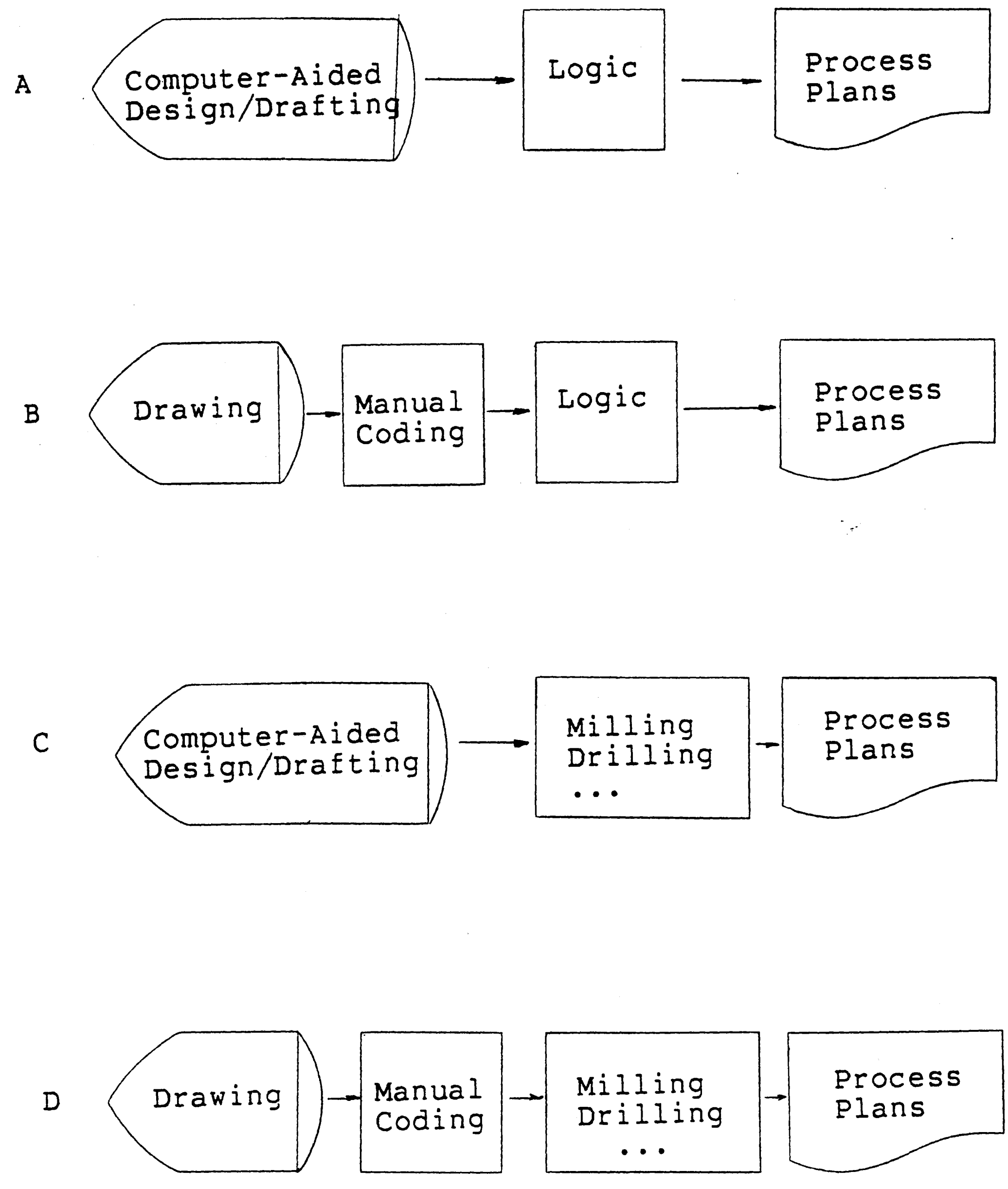


Figure 10. Classification of Automated Process Planning System

describe the part information and only selected manufacturing processes are developed. AUTAP, developed in West Germany [27] was designed for rotational parts (such as disks, rings, gears, wheels, and bolts) and sheet metal parts. This system uses a special part input language so that the same data base can be used to support part drawing, process plan generating, and NC program preparation.

The Computer-Aided Design and Computer-Aided Manufacturing process planning (CADCAM) system was developed by Chang [10]. CADCAM is an extension of the APPAS system. It allows the users to enter the design data interactively and displays the engineering drawing on a graphic system. This system automatically transforms the part geometric and technological information to the input code of the APPAS system. However, it can only create process plans for drilling.

The Totally Integrated Process Planning System (TIPPS), developed by Chang and Wysk [11], is a research system. TIPPS takes the design data in a boundary file, requires the user to locate the machining surfaces, then, generates the process plans for the identified machining surface.

Matsushima [52] has developed a process planning system named TOM (Technostructure Of Machining) by using the expert system technique. This system was designed to generate the optimal machining sequence from given geometry which is the output of a CAD system. However, it can generate only process sequences for machining holes. Things such as complicated geometry, selection of the optimal machining tool set, and different manufacturing processes planning cannot be handled by this system.

Figure 10-D illustrates the fourth class of generative process planning system in which only selected manufacturing processes are developed and human coding for the engineering data is required. A number of generative process planning systems of this class have been developed.

The Automated Process Planning And Selection (APPAS) was developed by Wysk [81] at Purdue University. This system generates process plans by analyzing a precoded number. It uses the COding FOR Machining (COFORM) coding system to describe the surface of a part. The COFORM system describes each individual surface of a part rather than describing the entire part. APPAS was primarily designed for milling and drilling work.

The Computer Managed Process Planning (CMPP) system, developed by United Technologies Research Center [20], is an advanced system for process planning of machined cylindrical parts. Similar to English, this problem oriented computer process planning language is used to state manufacturing processes for families of parts and stores the process description in the data base. Interactive techniques are used to collect detailed geometric data and technological data of a part. To generate the process plans, CMPP executes the process planning procedure which interacts with the system data base.

The GENerative process PLANning (GENPLAN), developed by Lockheed Georgia Company [74], has the capabilities to do part configuration analysis and instantly creates the work instructions for the manufacturing of subassemblies and aircraft parts. A special classification and coding system based on geometry, size, and manufacturing processes has been developed to describe the geometric data and manufacturing properties of the part. Since the process plans generated by this system require minor fill-ins, a trained process planner is required in the loop.

The Computer Aided Planning System (CAPSY-system), developed by Spur, Anger, Kunzendorf, and Stuckman [65], generates process plans for parts that require turning and drilling operations. The planning steps of the CAPSY system are arranged in four levels: management, procedure, machining area, and operation. The CAPSY-system is a dialogue system which allows the user to monitor the system during the generation of process plans.

The Automatic Computer Assisted Planning System (AUTOCAP) was developed by El-Midany and Davies [22]. AUTOCAP system was designed for turning operations and is a shop floor, dedicated mini-computer based system. The part information from the engineering drawing is entered by the user through an interactive program.

GARI, developed by Descotte and Latombe [18] in France, is a problem solver that creates process plans for machining parts. It is structured like an expert system. GARI consists of a specialized knowledge base and a general purpose solver. A part model has been developed to describe the geometrical and technological information of a part in terms of its attributes, such as holes, grooves, notches, and faces. This manual input is very complicated in use. GARI generates process plans for rectangular parallelopiped parts.

In addition to the pure variant or generative type systems, the combination of variant and generative techniques are also found in several planning systems. The Interactive Process Planning System for Prismatic Parts (ICAPP) was developed by Eskicioglu and Davies [24] at University of Manchester Institute of Science and Technology (UMIST). The ICAPP system is feature oriented and is capable of processing plane and cylindrical types of features. In the ICAPP system, a composite part is designed for each part family; parts in each family can be derived from its composite part. The variant planning data and the parameters of the generative logic for each composite part are kept in the Cutting Technology File (CTF). The ICAPP system selects the manufacturing methods from CTF for each part according to its feature type, dimensions, and tolerances.

The Rotating Part Operation (RPO), developed by Tipnis, Vogel and Lamb [72], was designed for aircraft engine rotational parts. This system stores the standard process plans according to each part family classification code, and automatically generates a detailed plan of each part by using the generative approach. Papers by El-Midany et al. [23] and Weill et al. [76] provide a good survey of existing NC programming generating systems and CAPP systems.

Summary

This chapter presents the background of process planning, the procedure of manual process planning, and a survey of the literature on the development of computer-aided process planning systems. Basically, CAPP systems are classified into two types of systems -- variant and generative. The difference of these two types of systems are based on the information processing.

The variant type system can be described as the sophisticated information retrieval system. The input and output information are totally dependant on the designer or the user. The generative type system analyzes the input information, transforms the information, makes the comparison, and generates the output information.

The logic of the variant type systems is based on the concept of Group Technology. These are general purpose systems. However, a certain amount of preliminary effort is required to implement a variant type CAPP system, such as establishing a suitable classification and coding system based on Group Technology to establish part family grouping and standard plans for every part family. This type of system is not suitable for use in an integrated CAD/CAM system, because a slight change on a part will require another special process plan.

The concept of generative type systems is to use computers to create process plans automatically. The ideal system of this type can create an optimal process plan for any given part without human intervention after the part is designed. Existing systems of this type either require manual coded information or they are restricted to a preassigned type of work. This research is aimed at designing and developing an expert process planning system that can automatically analyze the geometric shape of a part and create the process plan for a designed part.

CHAPTER IV

FEATURES RECOGNITION PROCEDURE

Introduction

A human being has the capability to observe an object and organize the information obtained from the object at the same time. However, at the present time, a computer does not have this cognitive capability. A computer with a single central processing unit does not process two pieces of information simultaneously, nor retrieve information automatically from the computer memory. It always requires an application program to retrieve, organize, and generate the information from the data stored in the computer memory. In order to retrieve the data efficiently, usually a data base is created to store the related data.

The starting point for the FREXPP system is the PADL-1 solid geometrical modeler. Using the PADL-1 geometrical modeler, the engineer designs a particular part that is of interest by sizing, adding, subtracting rectangular blocks and/or cylinders. The resulting part is then presented in three dimensions. The engineer may modify the designed part through the PADL-1 commands, if the designed part is not satisfactory. Once a satisfactory part is designed, FREXPP starts to build a data base for the feature recognition application program. Then, a form feature data base is created for FREXPP to generate process plans.

As described above, the automation of feature recognition requires both a data base and an application program to retrieve the information from the data base of the designed part, manipulate it, and identify the features of the part. Prior to the discussion of the data base creation and the form feature recognition procedure, the PADL-1 boundary file will be discussed. Then, a sequence of procedures for feature recognition will be described.

PADL-1 Boundary File

Internally, the PADL-1 geometrical modeler uses two kinds of representation schemes, Constructing Solid Geometry (CSG) and Boundary Representations (B-Reps), to represent the solids. "Boundary", in PADL-1, has a precise mathematical meaning and is defined as "if S is a compact regular set of points in E3 which models a solid, then a point P is in the boundary of S if there exists points arbitrarily close to P which are in S, and points arbitrarily close to P which are not in S" [32 p.3]. "Regular" means bounded, closed and homogeneous. E3 stands for a three dimensional Euclidean space. One can consider the boundary as the "skin" that encloses the solid. The boundary of a machined part is a collection of surfaces that enclose the part completely. The CSG scheme represents solids as a combination of primitive solids, and the B-Reps scheme define solids in terms of faces and edges. A face is a subset of the boundary of a primitive solid. An edge is the intersection of two faces.

The B-Reps of a solid are derived from the CSG representation through the boundary evaluator in the PADL-1 processor. Figure 11 shows these two representations employed in the PADL-1 system. The B-Reps scheme provides data for generating graphic displays, and the CSG representation provides data mainly for displaying shaded areas. The data provided by B-Reps is used as the input data for the feature recognition program. The boundary of a solid is represented by an ordered set of faces in a "boundary file" which is called a B-file in the PADL-1 system.

Figure 12 shows the data structure of a single part boundary file. This structure is known as a directed graph. There are three types of nodes: object (boundary) node, face nodes and edge nodes. The object node contains information, such as the boundary file name, number of faces, number of edges, the centroid point of an object, the size of the object, and all the face names. Each face node contains information used to describe a particular face, such as the face-type (X-type, Y-type, and Z-type), the number of edges of the face, the position of the face, and the edge names. Each edge node contains information about a particular edge, such as the edge-type (X-type, Y-type, and Z-type), the starting point and ending point of an edge, and a pair of face names which share the edge.

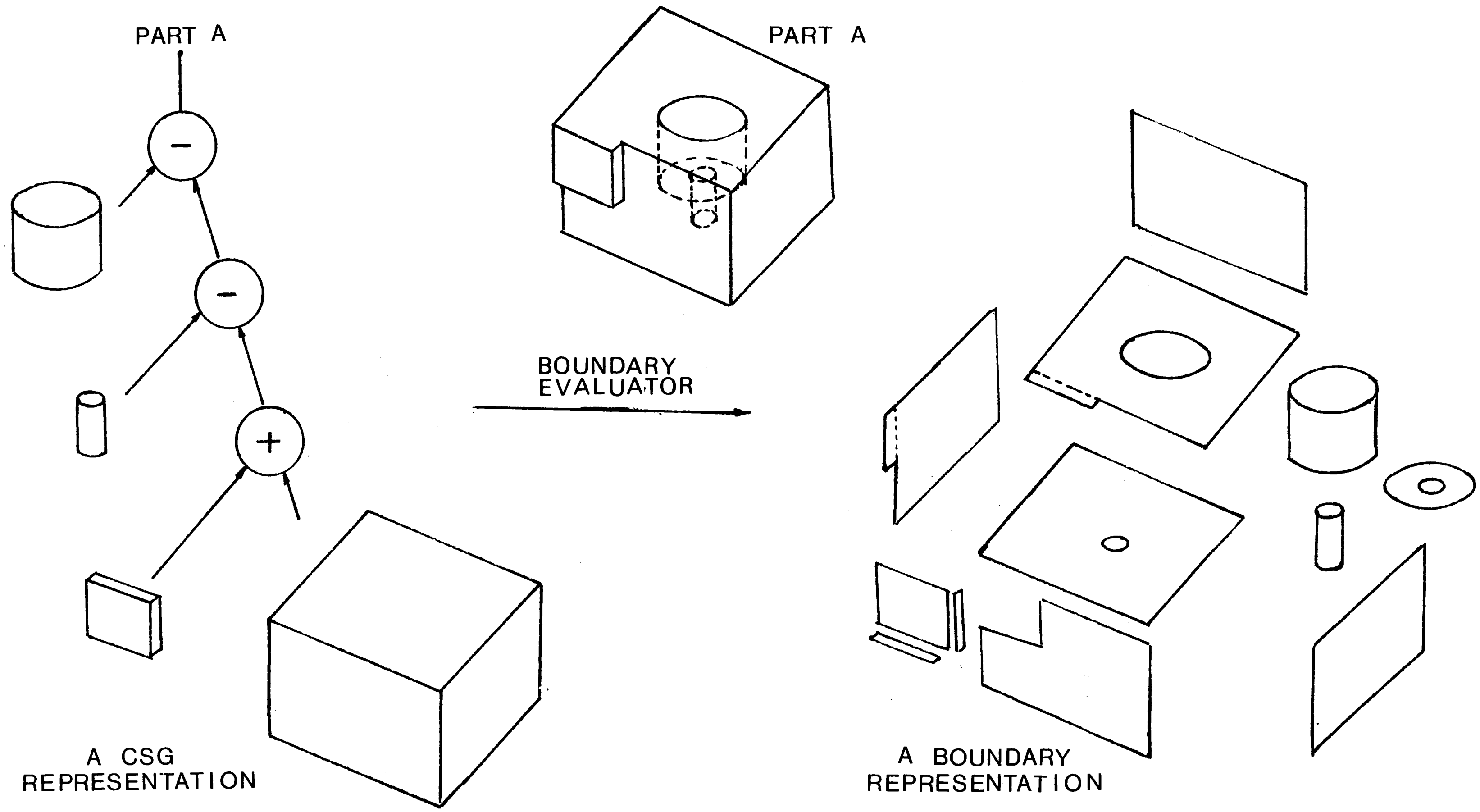


Figure 11. A CSG and Boundary Representation

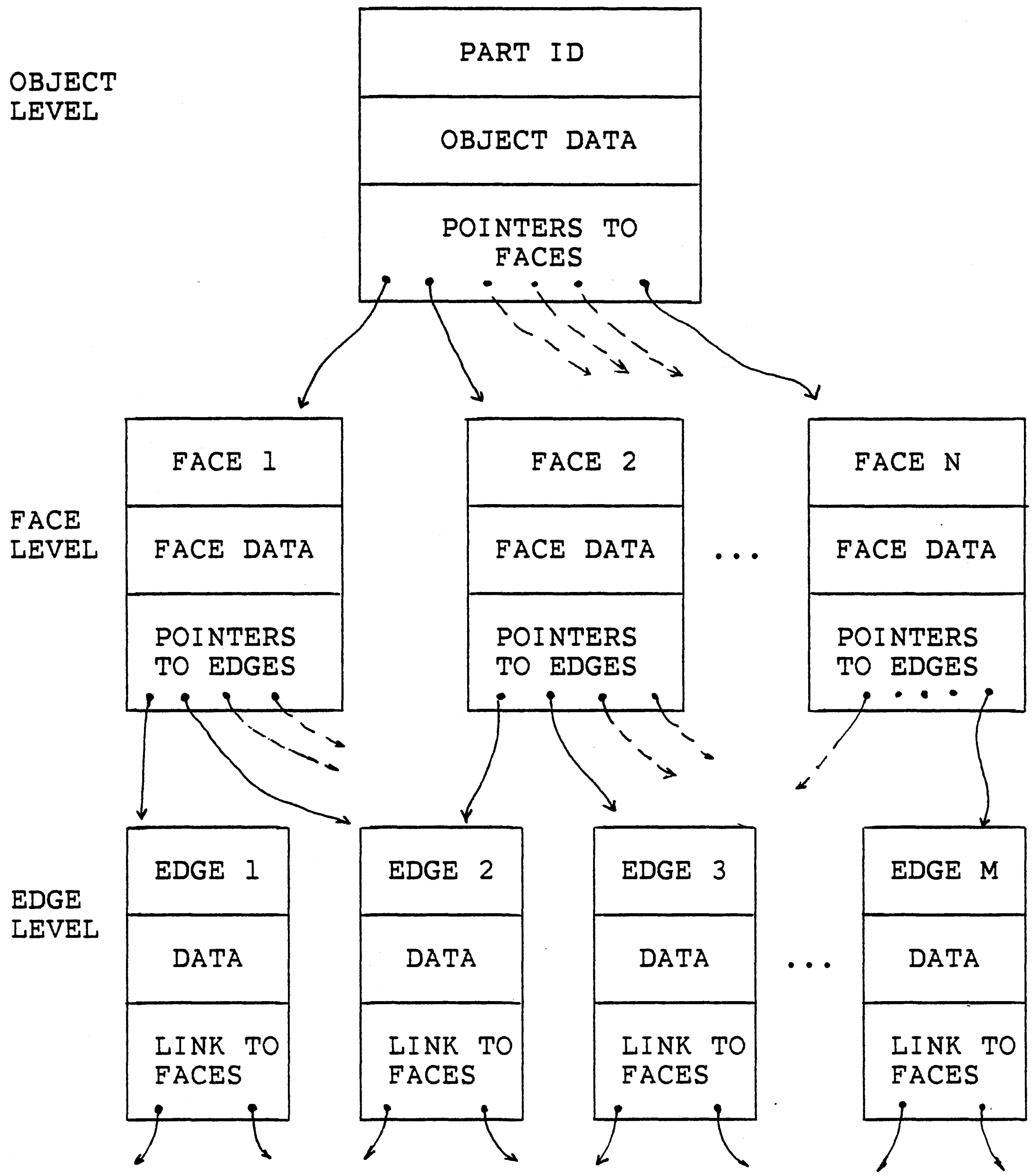


Figure 12. A Single Part Boundary Representation

Representations of Faces and Edges in PADL-1

Each type of face of rectangular solids is distinguished by a face-type code in the PADL-1 system. Each face-type code implies the direction of the "surface normal" in a Euclidean space. "Surface normal" is the normal vector of a face which points outward from the face of a solid. The complete definition of a surface normal and the face-type coding strategy are described in Appendix A.

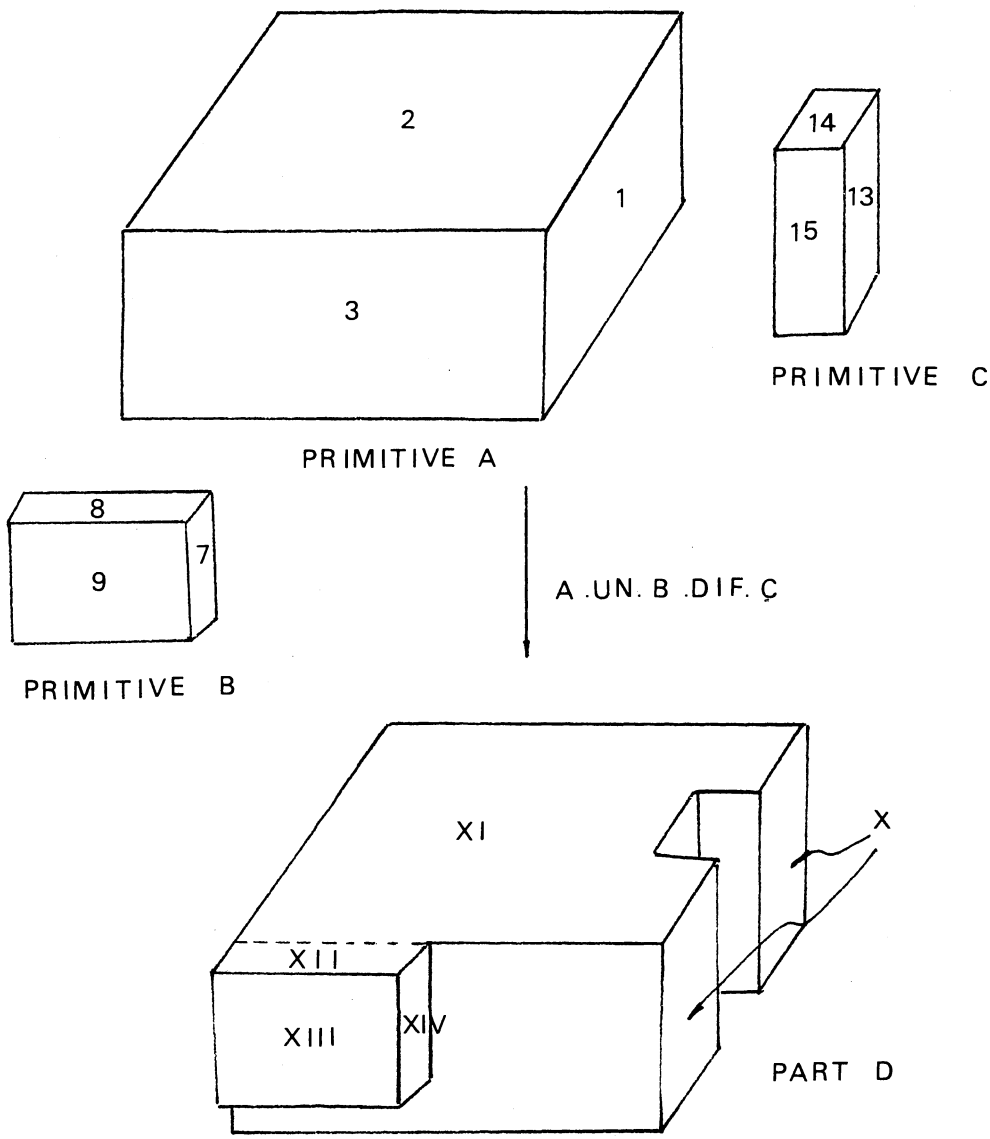
Table III contains a list of the surface normal directions and the face-type codes. The faces which are perpendicular to the X axis are called X-type faces with the face-type codes 101 and 99. The X-type faces whose surface normals point in the positive direction (+) have the 101 face-type code. Y-type surfaces and Z-type surfaces are those surfaces which are perpendicular to Y axis and Z axis respectively. The X-type disk faces have the face-type codes 201 and 199. The X-type cylindrical faces have the face-type codes 301 (for the solid cylinder) and 299 (for the hollow cylinder). The surface normals of the hollow cylinders point toward the axes of the cylinders. The surface normals of the solid cylinders point away from the axes of the cylinders.

In PADL-1, two types of face names are used. A "p-face" (primitive face) represents a face of a primitive. A primitive is a geometrical solid, such as rectangular blocks and cylinders. For example, in Figure 13, face 1, 2, and 3

TABLE III

FACE-TYPE CODES

Type of Face		Face Normal Direction	Face-type Code
Plane	X	-	99
		+	101
	Y	-	98
		+	102
	Z	-	97
		+	103
Disk	X	-	199
		+	201
	Y	-	198
		+	202
	Z	-	197
		+	203
Cylinder	X	-	299
		+	301
	Y	-	298
		+	302
	Z	-	297
		+	303



Note: Arabic numbers stand for p-face numbers
Roman numbers stand for b-face numbers

Figure 13. Illustration of P-faces and B-faces

of primitive A are p-faces. A "b-face" (boundary face) is the boundary of a part. A b-face either has the shape of the original p-face or has a reduced shape. For example, part D is the result of the operations of adding primitive B to primitive A and subtracting primitive C from the union of primitives of A and B. Face number XI is a b-face which is the result of subtracting a portion from p-face number 2. Face number XII is the b-face of p-face number 8. Face number X is a b-face containing two parts which is the result of subtracting a portion of p-face number 1. The association of b-face XI and XII will be discussed in the next section.

B-faces are bounded by edges. Edges in PADL-1 are classified as LINE edges, ARC edges, and CEDGE edges. LINEs are the collection of straight lines. ARCs are the collection of circular arcs. CEDGEs are the collection of the intersection of two cylindrical faces. CEDGEs are not considered in this research. LINEs and ARCs are subdivided into X, Y, and Z type edges. An X-type line is a line that is parallel to X axis. An X-type arc is the intersection of an X-type cylindrical face and an X-type plane or disk face. Each type of edge has been assigned a specific edge code. LINE type edge codes are 1001, 1002 and 1003 for X, Y, and Z type lines respectively. ARC edge codes are 2001, 2002, and 2003 for X, Y, and Z type arcs respectively.

Edges are further identified as outer boundary edges and inner boundary edges. The outer boundary edges represent the outer rims of each b-face. The inner boundary edges bound the area which is not a portion of a b-face within the outer boundary edges. A string of connected outer boundary edges is called an "outer boundary loop". A string of connected inner boundary edges is called an "inner boundary loop".

For example, in Figure 14, the front face of the solid is a b-face with section I and section II. This is the result of subtracting a block from the front middle of a block, a cylinder from the left leg and a block from the right leg. The outer boundary edges are edges a, b, c, d; and e, f, g, and h. The inner boundary edges are edges i; and j, k, l, and m. Section I of this b-face is a surface which is bounded by the outer boundary loop 1 and the inner boundary loop 3. Section II of this b-face is a surface which is bounded by the outer boundary loop 2 and the inner boundary loop 4.

The PADL-1 data structure for b-faces is good for displaying a part on a graphical system, but not for representing a feature. A b-face may contain two sections which represent the surfaces of two different features, such as sections I and II in Figure 14. When two primitives are added together the combined surface is represented as two b-faces. Each b-face represents a part of a surface, such as the b-faces XI and XII shown in Figure 13.

Surfaces of Form Features

In this research, the surfaces of a form feature are called "f-faces". F-faces are used to identify surfaces

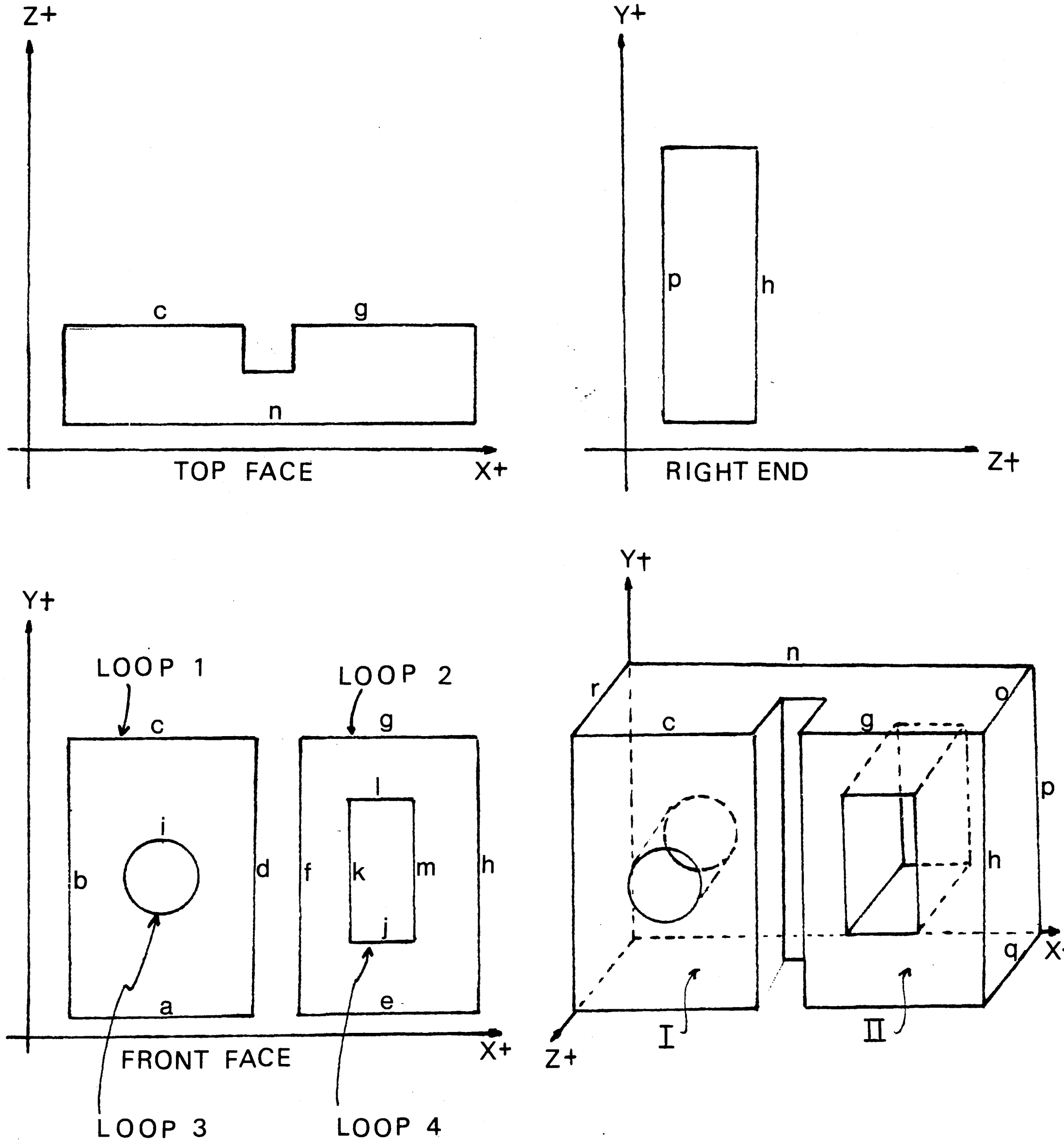


Figure 14. Boundary Loops of B-faces

that can be formed during a single machine step. A plane f-face contains only one outer boundary loop with or without inner boundary loops. The shape of a plane f-face can be the same shape of a b-face, such as b-face XIII in Figure 13 page 68, or identified as the combination of b-faces, such as b-faces XI and XII of Figure 13 which form one f-face. It can also be identified as a portion of a b-face, such as section I and II in Figure 14 forming two f-faces. A cylindrical b-face is considered as an f-face. F-faces are the primary key for the FREXPP feature recognition procedure.

The procedure to generate f-faces is described in the following three sections:

- Creation of the B-face FACE File and the EDGE File
- Construction of Boundary Loops
- Creation of F-faces

Creation of the B-face FACE File and the EDGE File

PADL-1 does not maintain a permanent boundary file. The boundary file described in the previous section is calculated in PADL-1 as needed. It was designed mainly to facilitate displaying the geometry of a designed part. However, to identify the form features, FREXPP needs the surface information and edge information for the part.

The first stage for generating the f-faces is to build a b-face FACE file and an EDGE file. The input to the b-face FACE file and the EDGE file creation procedure is the boundary file of the part and a temporary file. The boundary file is a sequential file which contains all of the b-face information and the b-face boundary edge information. The temporary file contains the b-face name, face-type and number of edges. These two files are generated during boundary evaluation of the PADL-1 system.

During the execution of the FREXPP file creation procedures, the face information and edge information in the boundary file are separated and stored in the b-face FACE file and EDGE file respectively. The contents of the b-face FACE file and the EDGE file are listed in Appendix B. The following tasks are completed while separating the face and the edge information:

1. Edges are assigned integer names and ordered according to the sequence of the edges appeared in the boundary file.
2. Edges which are not the boundary edges of b-faces are not linked to the b-face record and not stored in the EDGE file. For example, in Figure 15, edge b of primitive C is expressed as edges b1 and b2 after the union operation in PADL-1; and edge d is expressed as edge d1 and d2. Edges a, b1 and d1 are not the boundary edges of b-face 1.
3. Two plane faces, which share an edge and form a new plane surface, are connected through a linked field. This shared edge is not stored in the EDGE file and is not linked to both b-face records, for example, edge e in Figure 15. The shared edge is

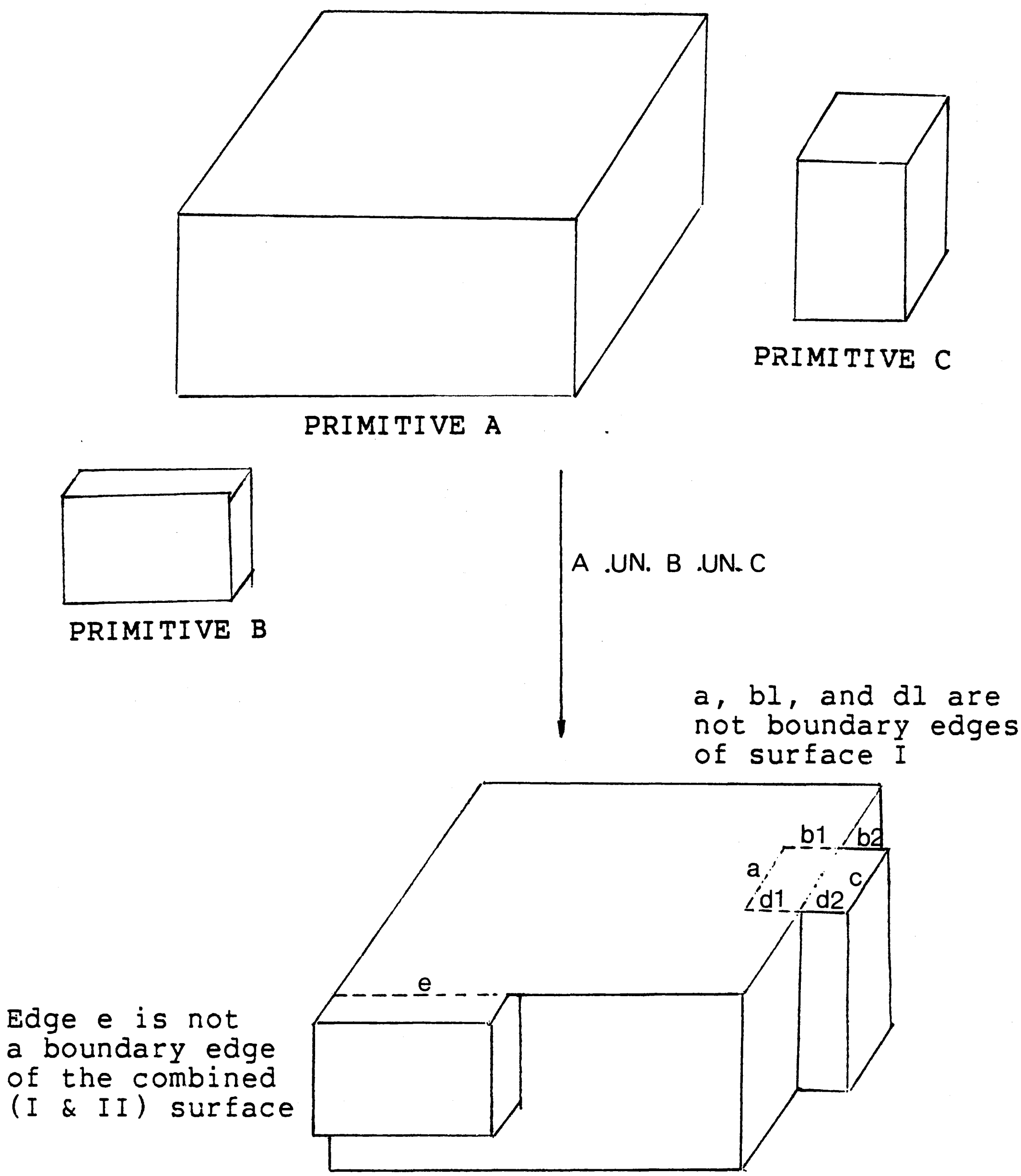


Figure 15. Edges Which are not Boundary Edges of Faces

not a boundary edge of the face.

4. If an edge is the intersection of two cylindrical faces, an overflow record in the EDGE OVERFLOW file is used to store the excessive edge data. This arrangement is designed to save the storage space and allow for future development. Features of two intersecting cylinders are not considered in this research.
5. The b-face name and its associated information are stored in the b-face FACE file.
6. If a b-face has more than 12 boundary edges, an overflow record in the FACE OVERFLOW file is used to store the excessive names of these edges.

The newly created FACE file and EDGE file are random access files. The record numbers of the FACE file correspond to the face names. The record numbers of the EDGE file correspond to the edge names.

Construction of Boundary Loops

A b-face may contain more than one outer boundary loop. However, an f-face can be bounded by only one outer boundary loop. The second stage for generating f-faces is to construct the boundary loops and identify the outer boundary loops for each f-face.

Background. In PADL-1, the measuring of angles of arcs is based on a local three dimensional coordinate system. The three coordinates of the system are assigned as (RIGHT, UP, and FRONT). This local system is adopted in FREXPP to express edges on a plane surface.

The relation between the PADL-1 coordinate system and a local coordinate system is listed in Table IV. For an X-type face (expressed by Y and Z coordinates), X coordinate of PADL-1 corresponds to FRONT coordinate and is a constant; Z and Y coordinates correspond to RIGHT and UP axes respectively. For a Y-type face (expressed by X and Z coordinates), Y coordinate corresponds to FRONT coordinate and is a constant; X and Z coordinates correspond to RIGHT and UP coordinates respectively. For a Z-type face (expressed by X and Y coordinates), Z coordinate corresponds to FRONT coordinate and is a constant; X and Y coordinates correspond to RIGHT and UP coordinates respectively. If Z axis corresponds to FRONT axis, then X and Y axes correspond to RIGHT and UP axes respectively.

<u>On Line Representation of Edge Information</u>. Information describing each edge is brought into main memory from the secondary memory by using an EDGE-LINK node as a key. Since a LINE edge, in the PADL-1 system, is parallel to one of the axes, the starting point and the ending point are located at the same distance from that axes. Therefore, only three parameters are needed to represent a LINE edge, axis offset, edge starting position and edge ending position.

An EDGE-LINK node for describing a LINE edge on a surface contains the following information:

TABLE IV

PADL-1 COORDINATE SYSTEM VERSUS LOCAL COORDINATE SYSTEM

	LOCAL SYSTEM		
	FRONT	RIGHT	UP
PADL-1 SYSTEM	X	Z	Y
	Y	X	Z
	Z	X	Y

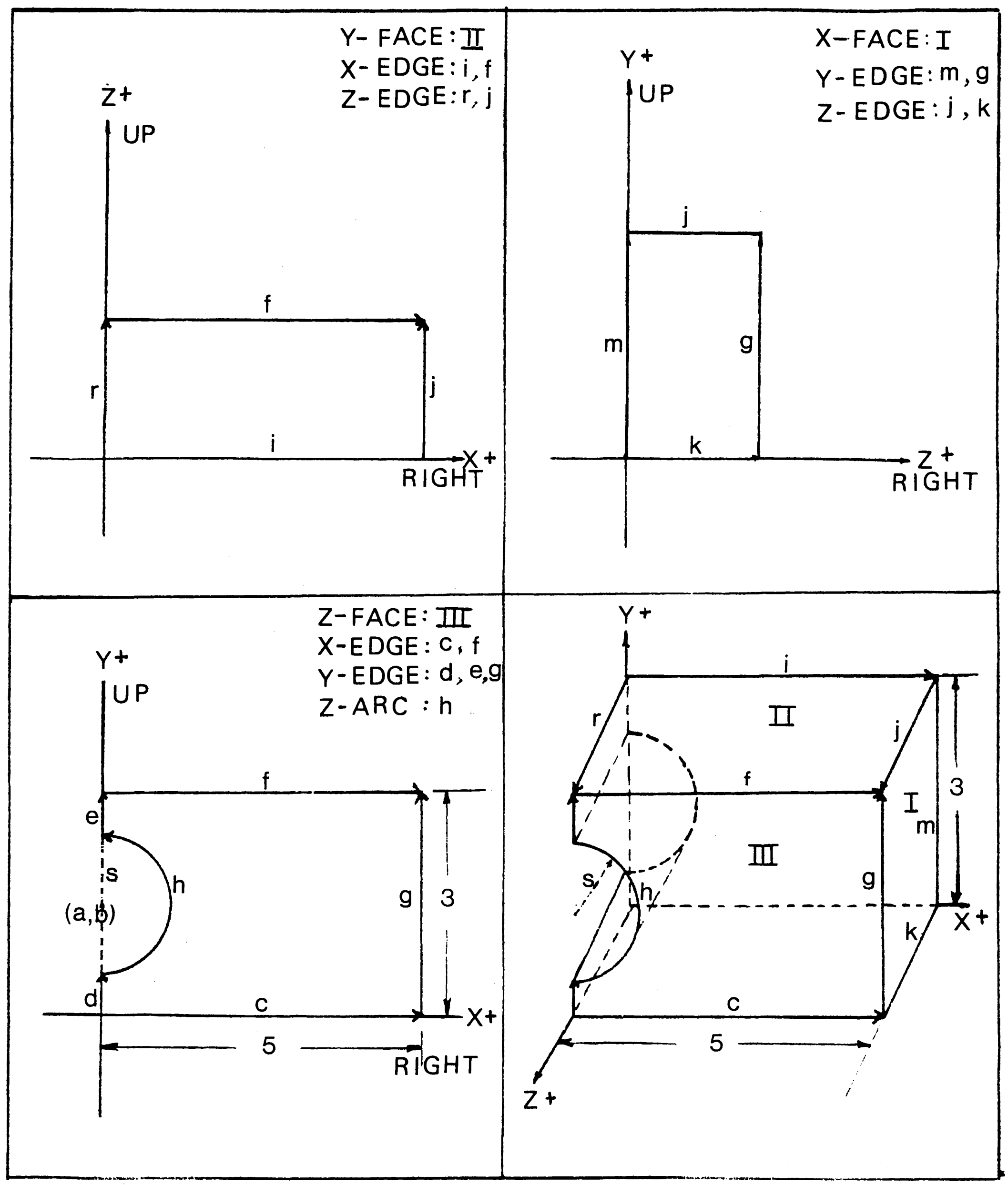


Figure 16. Directed Edges on a Local Two Dimensional Coordinates System

1. Edge name. For example, letter c, d, e, f, and g in Figure 16 are LINE edge names.
2. Edge Axis offset: It is the distance of the edge with respect to the local axis which is parallel to the edge. For example, the offset of edge g and f in Figure 16 is 5 and 3 respectively.
3. Edge starting position: It is the coordinate of the starting point. For example, the starting position of edge g in Figure 16 is 0.
4. Edge ending position: It is the coordinate of the ending point. For example, the ending position of edge g in Figure 16 is 3.
5. Sharing face name: It is the other surface which shares this edge. For example the sharing face name of edge g of face III in Figure 16 is face I.

An EDGE-LINK node for an ARC edge contains the following information:

1. Edge name. For example, letter h in Figure 16 is the ARC edge name.
2. Position of the center point of the ARC edge. For example, letter a and b in Figure 16 represent the position of the center point of the ARC edge h.
3. Radius of the ARC edge. For example, letter s in Figure 16 represents the radius of the ARC edge h.
4. Sharing face name: The other surface which shares the circular edge. The Sharing face of ARC edge h of face III in Figure 16 is cylindrical face IV.

An ARC edge is defined by the radius, a center point, and the bounding region which is defined by a minimum angle and a maximum angle. The measuring of these two angles is based on the RIGHT and UP axes of a sub-local coordinate system. When measuring the minimum and the maximum angles of an arc, the center point of the arc is assumed to be the original point of a sub-local system. Because of the effect of the directed graph, the angles must be measured in a certain direction (counterclockwise). The RIGHT axis is assigned with 0 degree and is the measuring reference line. The angle between RIGHT and UP is 90 degree measured counterclockwise.

The range of the minimum angle, measured counter-clockwise from the RIGHT axis to the line that links the center point to the starting point of an arc, is between 0 and 360 degrees. The range of the maximum angle, measured from the RIGHT axis to the ending point of an arc, is between the minimum angle and 360 degrees plus minimum angle.

For example, In Figure 17, arcs a, b, c, and d are Z-type arc edges which are defined by X and Y coordinates. Z coordinate corresponds to the FRONT coordinate. Arc a has a minimum angle of 0 degree and a maximum 180 degrees; arc b has 90 degrees for the minimum angle and 270 degrees for the maximum angle; arc c has a minimum angle of 180 degrees and a maximum angle of 360 degrees; arc d has 270 degrees for the minimum angle and 450 degrees for the maximum degrees.

The coordinates of the starting point and ending point of an arc are calculated by using the following equation.

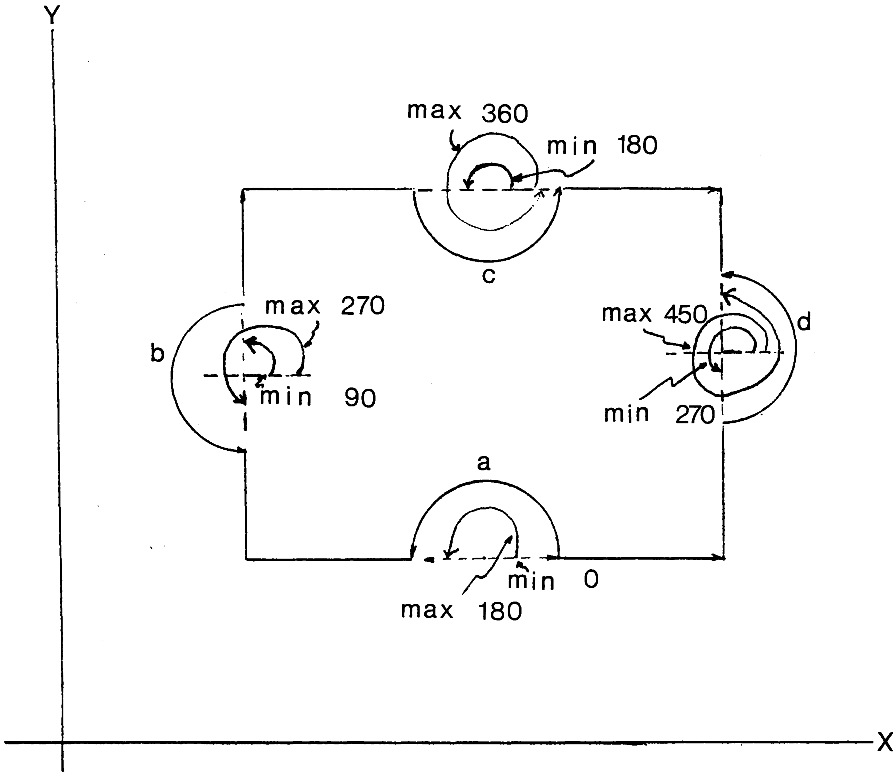


Figure 17. Angles of the Arcs on a Z-type Face

Ps = (a + r*cos(min), b - r*sin(min))

Pe = (a + r*cos(max), b - r*sin(max))

where, Ps is the starting position of an arc,

Pe is the ending position of an arc,

(a,b) is the center point,

r is the radius,

min is the minimum angle, and

max is the maximum angle.

The coordinate of the starting point (Ps) and the ending point (Pe) of arc h in Figure 16 page 79 are calculated as follows:

Ps = (a + r*cos(270), b - r*sin(270))

= (a + r*0, b - r*(-1))

= (a, b + r)

Pe = (a + r*cos(450), b - r*sin(450))

= (a + r*0, b - r*1)

= (a, b - r)

The measure of the angles is obtained from the EDGE file. When all the edge information has been brought into the computer main memory, the next stage is to construct the boundary loops.

Constructing the Loops. In this stage, edges of a surface are first sorted according to the edge-type (X-type, Y-type, or Z-type) code. They are grouped into three categories named GROUP A, GROUP B and GROUP C. GROUP A contains all the edges which are parallel to the RIGHT axis. GROUP B contains all the edges which are parallel to the UP axis. GROUP C contains all the arc edges.

LINE edges are ordered from the smallest edge axis offset to the largest edge axis offset in each of groups A and B. Edges which have the same axis offset are ordered by the starting position of an edge. For example, Edges of the top surface shown in Figure 18 are grouped and ordered as the following manner:

GROUP A: 1, 6, and 4

GROUP B: 2, 3, 5, and 7

GROUP C: 8

The purpose of ordering the edges are:

1. To facilitate identifying the edges which are part of the same line, and
2. For the efficiency in finding unlinked edges.

A closed loop is formed by linking the edges. The following steps are developed to construct the closed loops. Figure 19 shows a flow chart of this procedure.

1. Determine a loop number. The loop number is set to one initially. It is increased by 1 whenever a loop is constructed.
2. Select a starting edge. The strategy used in this search is to find the edge which is currently located at the lowest edge axis offset among all edges available in GROUP A. An edge availability means that this edge has not been used in any other loop. If there is no edges available in GROUP A, then a search in GROUP B is started. If both GROUP B has no edges available, then a search in GROUP C

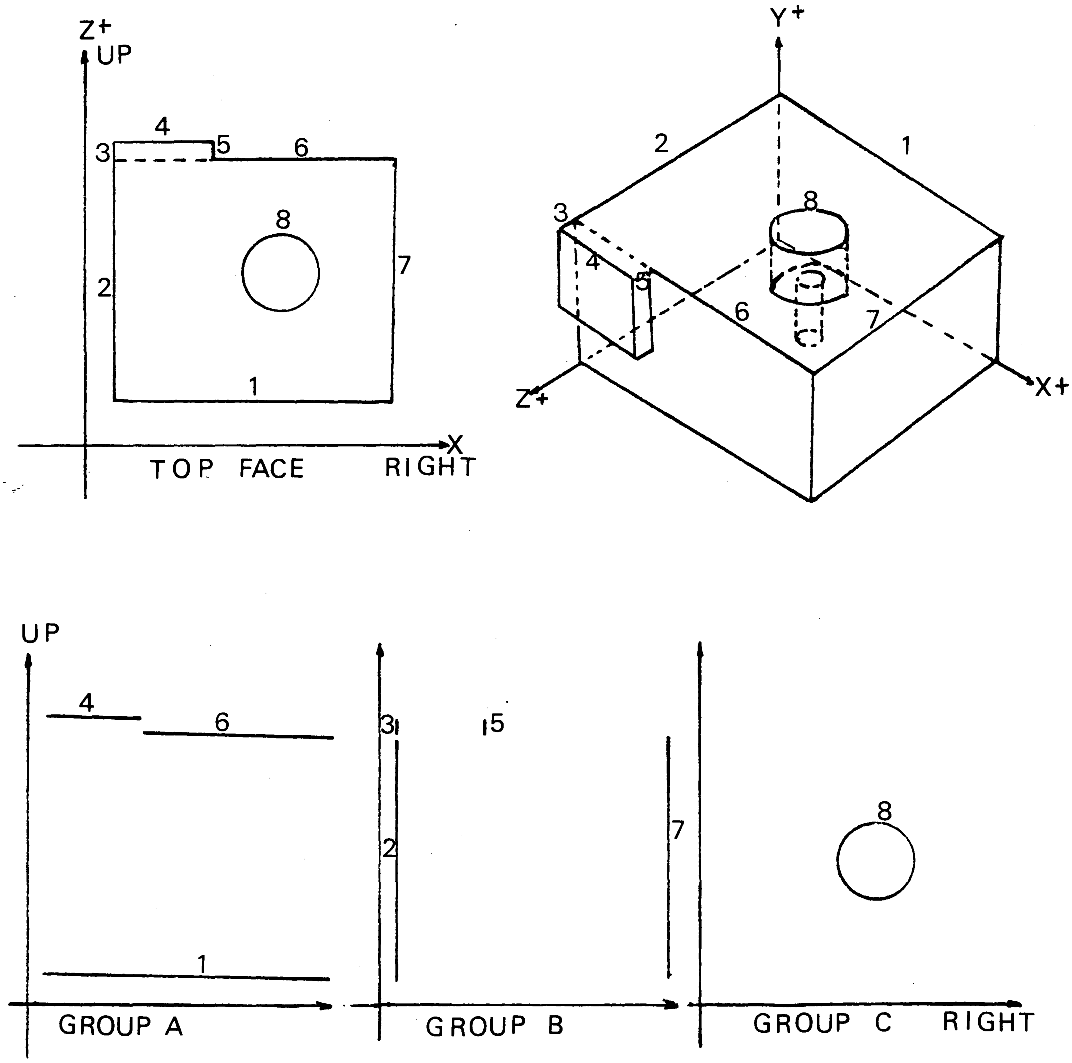


Figure 18. Edge Groups of a B-faces

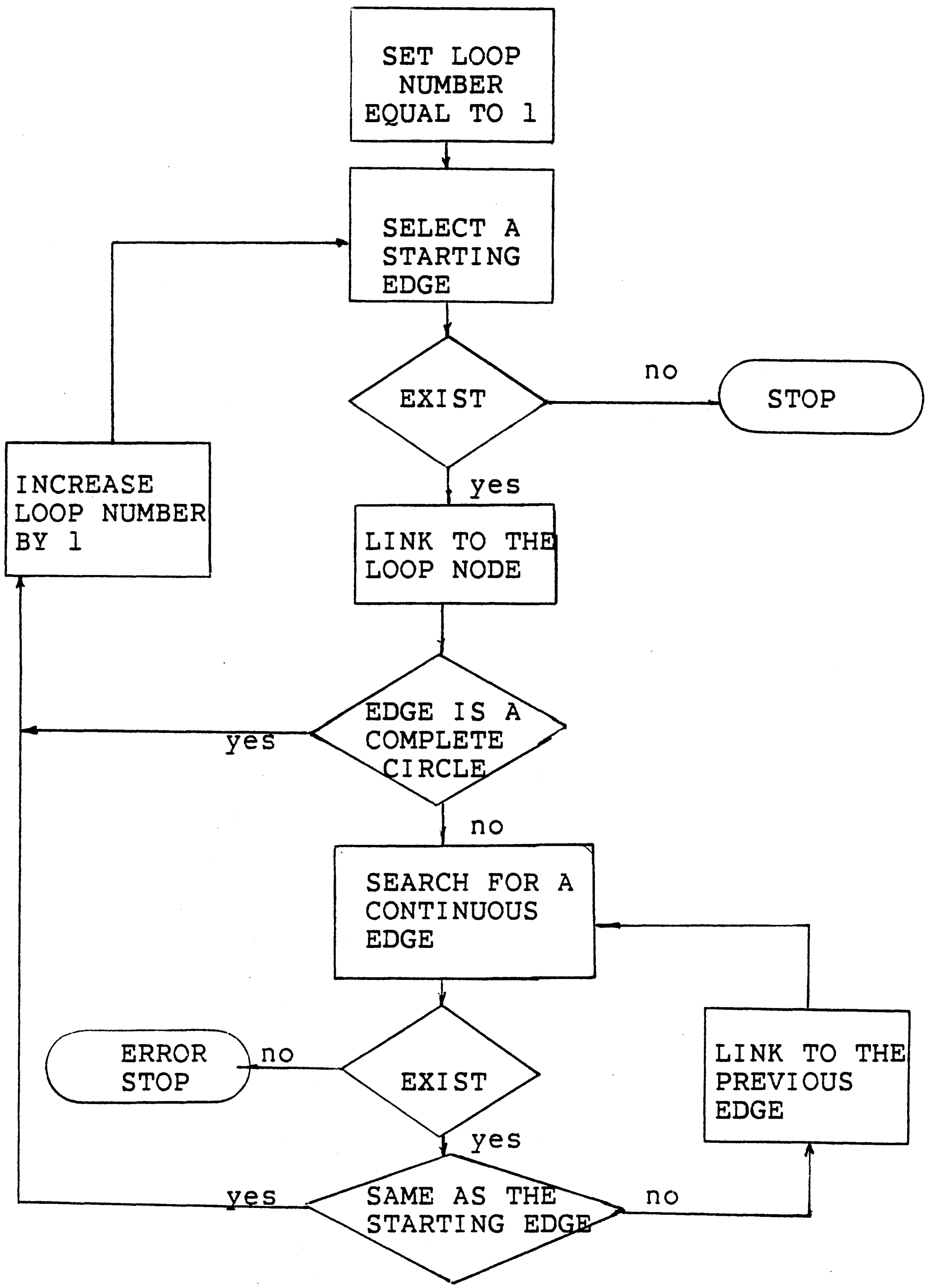


Figure 19. Boundary Loop Constructing Flow Chart

is started. If no available edges are found in GROUP C, the loop constructing procedure is complete.

3. Mark the edge. If an edge is found, mark the edge as an unavailable edge. If this edge is a complete circle, then go to step 1.

4. Search a matching edge. If this edge is not a complete circle, then set the edge as "Edge to be Linked" and search a matching edge of the edge. The groups to be searched are listed in the following manner:

Group of Previous Edge to be Linked	The First Search Group	The Second Search Group	The Third Search Group
A	B	A	C
B	A	B	C
C	A	B	

5. Link the edge. If a matching edge is found and is not the starting edge, then link the matching edge to the previous edge and go to step 4. If the matching edge is the starting edge, then go to step 1.

6. Error. If a matching edge is not found, then the system stops and notes the error of an open loop.

For example, Table V is a list of loop constructing steps for the top surface shown in Figure 18. Two loops are formed in this example. On the completion of these steps, all of the possible loops of a b-face or combined b-faces are formed. The next step is to generate the f-faces.

TABLE V

AN EXAMPLE OF LOOP CONSTRUCTING PROCEDURE

STEP	LOOP NUMBER	STARTING EDGE	EDGE TO BE LINKED	SEARCH GROUP	MATCHED EDGE	END LOOP
1	1					
2	1			A		
3	1	1		A		
4	1	1	1	B		
5	1	1	1		7	NO
4	1	1	7	A		
5	1	1	7		6	NO
4	1	1	6	B		
5	1	1	6		5	NO
4	1	1	5	A		
5	1	1	5		4	NO
4	1	1	4	B		
5	1	1	4		3	NO
4	1	1	3	A		
4	1	1	3	B		
5	1	1	3		2	NO
4	1	1	2	A		
5	1	1	2		1	YES
1	2					
2	2			A		
2	2			B		
2	2			C		
3	2	8				
1	3			A		
1	3			B		
1	3			C		
1						STOP

Creation of F-faces

F-faces are used to identify surfaces that can be formed by a single machine step. An f-face is enclosed by one outer loop with or without inner loops. In order to generate f-faces, related outer loops and inner loops must be associated. The strategy used in constructing the boundary loops starts with an edge which is located at the lowest edge axis offset among all of the available edges of the same type (X-type, Y-type, and Z-type). This strategy assures that:

1. The first constructed boundary loop is an outer boundary loop, since the starting edge of the first boundary loop is an outer-most edge, and
2. An inner boundary loop will not be constructed prior to the construction of its outer boundary loop.

Based on the above two premises, the following rules are developed to examine if a boundary loop is (i) an inner boundary loop with respect to a known outer boundary loop, such as loop IV in Figure 20 which is an inner loop with respect to loop I, or (ii) an outer boundary loop, such as loops II and III which are outer boundary loops.

Outer and Inner Boundary Loop Construction Rules. In these rules, the "acting loop" is the boundary loop to be classified. The "acting surface" is the surface defined by the acting loops. The sharing surface is the surface which shares the starting edge with the acting surface. The part

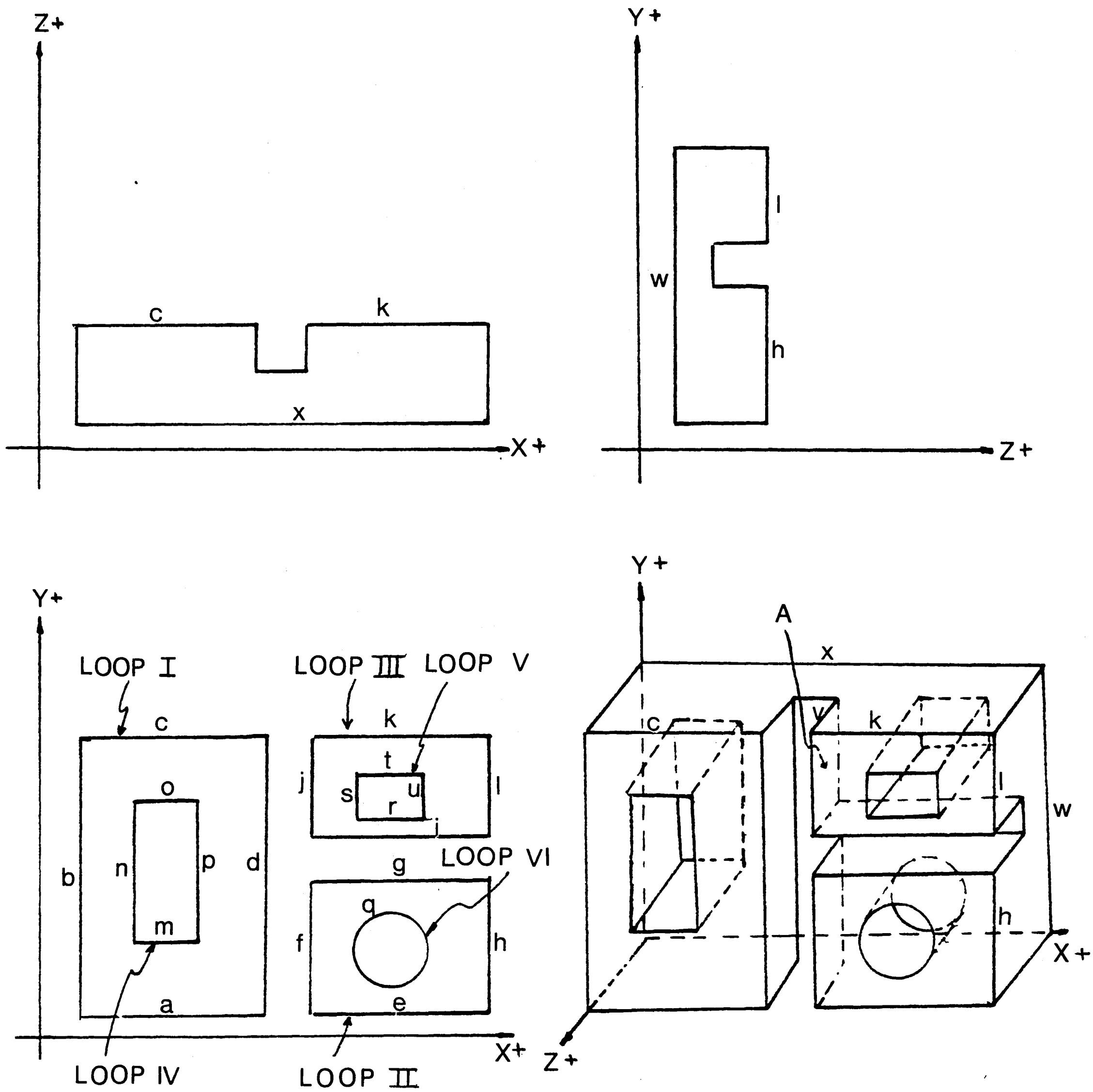


Figure 20. Inner Loops and Outer Loops of an Abstract Part

shown in Figure 20 is used to illustrate the following rules:

1. The first constructed boundary loop is always an outer boundary loop. For example, loop I is an outer boundary loop, because edge a has the least edge axis offset.
2. If the lowest edge axis offset of the same edge-type of the acting loop is lower or equal to the lowest edge axis offset of the same edge-type of the known outer boundary loop, then the acting loop is an outer boundary loop. For example, the edge axis offset of edge e of loop II is equal to the edge axis offset of edge a of loop I. Loop II is an outer boundary loop with respect to loop I.
3. If the highest edge axis offset of the same edge-type of the acting loop is higher or equal to the highest edge axis offset of the same edge-type of the known outer boundary loop, then the acting loop is an outer boundary loop. For example, the edge axis offset of edge k of loop III is equal to the axis offset of edge c of loop I. Loop III is an outer boundary loop with respect to loop I.
4. If the lowest edge axis offset of the same edge-type (X-type, Y-type, and Z-type) of the acting loop is higher than the highest edge axis offset of the same edge type of the known outer boundary loop, then the acting loop is an outer boundary loop. For example, the edge axis offset of edge r

of loop III is higher than the edge axis offset of edge g of Loop II. Loop III is an outer boundary loop with respect to loop II.

5. If the highest edge axis offset of the same edge-type of the acting loop is lower than the lowest edge axis offset of the same edge type of the known outer boundary loop, then acting loop is an outer boundary loop. For example, the edge axis offset of edge g of loop II is lower than the edge axis offset of edge i of loop III. Loop II is an outer boundary loop with respect to loop III.

The above developed rules are used for identifying the individual outer boundary loops. If a loop cannot be identified by the above rules, the decision table illustrated in Table VI is used to identify the inner and outer loops. For example, Rule 6 can be interpreted as follow:

6. If the surface normal of the sharing surface points in the negative direction, the starting point of an edge of the sharing surface which is perpendicular to the acting surface is not on the acting surface, and the surface normal of the acting surface points in positive direction, then this boundary loop is an outer boundary loop. For example, the surface normal of surface A points in the negative direction. The starting point of edge v is not on the acting surface. Loop III is an outer boundary loop with respect to loop I.

TABLE VI

OUTER AND INNER LOOPS DECISION TABLE

Rule Number	6	7	8	9	10
IF					
Surface Normal Direction of the Acting Surface	-	-	-	-	•
Surface Normal Direction of the Sharing Surface	+	-	+	-	+
The Starting Point of an Edge, Which is on the Sharing Surface and is Perpendicular to the Acting Surface, is not the Acting Surface	F	T	T	F	•
Then					
Inner Boundary Loop	F	F	T	T	T
Outer Boundary Loop	T	T	F	F	F

T: True; F: False

As all the boundary loops of a b-face or combined b-faces are identified, the next step is to link the inner boundary loops to their outer boundary loop.

Associating Inner Loops with Outer Loops. To associate an inner boundary loop with the proper outer boundary loop, FREXPP starts to find an edge which is located at the lowest edge axis offset of all the edges of the same edge type of an inner loop. This edge is named as a REFERENCE edge. FREXPP then finds an edge of the same type (X-type, Y-type, or Z-type) with all the following characteristics. The part shown in Figure 20 page 90 is used to illustrate these characteristics.

1. The edge axis offset of the edge is lower than the edge axis offset of the REFERENCE edge. For example, in Figure 20, edge s is the REFERENCE edge which is located at the lowest edge axis offset of Y-type edge of the inner loop V. Edges b, n, p, d, f, and j belong to the Y-type edge and their edge axis offsets are lower than the edge axis offset of edge s.
2. The edge is on an outer boundary loop. For example, in Figure 20, edges b and d are on the outer loop I and edges f and j are on the outer loops II and III respectively. Edges n and p have the edge axis offsets lower than the edge axis offset of edge s, but they are on the inner loop IV.
3. The range (the distance between the starting point

and the ending point) of the edge must encompass the range of the REFERENCE edge or have overlap with the range of the REFERENCE edge for it to be an inner loop. For example, the ranges of edges b, d, and j encompass the range of edge s.

4. The axis offset of the edge is the highest that has the characteristics found in 1, 2, and 3. For example, edge b is the identified edge, since the edge axis offset of edge j is higher than the edge axis offset of edges b and d.

After finding the edge, the inner boundary loop which contains the REFERENCE edge is associated with the outer boundary loop of the identified edge. For example, in Figure 20, the inner loop V is associated with the outer loop III because edge j meets all three characteristics compare to edges b and d. The data of these two loops is stored in one FACE record.

If an inner loop is formed by the arcs or the combination of arcs and lines, then the location of the center point of an arc is assumed to be the lowest edge axis offset of the loop. By using the same procedure, the related outer boundary loop can be found. For example, in Figure 20, the inner loop VI is associated with the outer loop II because edge f meets all three characteristics compare to edges b and d.

After all of the boundary loops of a b-face or combined b-faces are examined, the following observations can be made:

1. This b-face is identified as a single f-face, when the b-face is not combined with other b-faces and contains only one outer boundary loop.
2. A new f-face is formed, when the b-face is combined with other b-faces and they are enclosed by a single outer boundary loop.
3. More than one f-face is formed, when the b-face or combined b-faces have more than one outer boundary loop.

A single b-face which is not an f-face is marked as a non-f-face in the b-face FACE file, for example, b-face II and III in Figure 18, page 85, are marked as non-f-faces when they form an f-face. New FACE records and names are generated for each of the newly generated f-faces. EDGE records are updated with the f-face names. The newly created f-faces are assumed to have the same face-type code that the combined b-faces have. The face-type code of combined plane surfaces and disk surfaces is assigned to the plane surfaces code.

For example, Table VII A and B display a partial list of the b-face FACE file and the updated b-face FACE file respectively. Faces 15 and 16 are the two newly generated f-faces. Faces 2, 4, 8, and 10 are marked as non-f-faces. Table VIII A and B display a partial list of the EDGE file and the updated EDGE file respectively. Edges 1, 5, 6, 7, and 8 were shared by faces 2 and other faces. In the updated EDGE file, face number 2 is replaced by face number 15.

TABLE VII

CONTENTS OF B-FACE FACE FILE

	FACE NUMBER	FACE-TYPE CODE	NUMBER OF EDGES	P-FACE NAME	F-FACE FLAG
	1	101	4	8	-1
	2	102	5	9	-1
	3	103	6	10	-1
	4	99	4	11	-1
	5	98	5	12	-1
	6	97	4	13	-1
	7	101	4	15	-1
A.	8	102	3	16	-1
INITIAL	9	103	4	17	-1
STATE	10	99	3	18	-1
	11	98	4	19	-1
	12	202	2	23	-1
	13	298	2	24	-1
	14	298	2	31	-1
	1	101	4	8	-1
	2	102	5	9	0
	3	103	6	10	-1
	4	99	4	11	0
	5	98	5	12	-1
	6	97	4	13	-1
	7	101	4	15	-1
B.	8	102	3	16	0
UPDATED	9	103	4	17	-1
FILE	10	99	3	18	0
	11	98	4	19	-1
	12	202	2	23	-1
	13	298	2	24	-1
	14	298	2	31	-1
	15	102	8	16	-1
	16	99	7	18	-1

TABLE VIII

CONTENTS OF EDGE FILE

	EDGE NUMBER	EDGE-TYPE CODE	EDGE UPDATE FLAG	DUMMY VARIABLES	FACE NAME	FACE NAME
A. INITIAL STATE	1	1003	-1	0 0 0	1	2
	2	1002	-1	0 0 0	1	3
	3	1003	-1	0 0 0	1	5
	4	1002	-1	0 0 0	1	6
	5	1001	-1	0 0 0	2	3
	6	1003	-1	0 0 0	2	4
	7	1001	-1	0 0 0	2	6
	8	2002	-1	0 0 0	2	13
	9	1002	-1	0 0 0	3	4
	10	1001	-1	0 0 0	3	5
	11	1002	-1	0 0 0	3	7
	12	1003	-1	0 0 0	3	11
	. . .	. . .	. . .	. . .	. . .	. . .
	25	2002	-1	0 0 0	12	13
	26	2002	-1	0 0 0	12	14
B. UPDATED FILE	1	1003	0	0 0 0	1	15
	2	1002	-1	0 0 0	1	3
	3	1003	-1	0 0 0	1	5
	4	1002	-1	0 0 0	1	6
	5	1001	0	0 0 0	3	15
	6	1003	0	0 0 0	4	15
	7	1001	0	0 0 0	6	15
	8	2002	0	0 0 0	13	15
	9	1002	0	0 0 0	3	16
	10	1001	-1	0 0 0	3	5
	11	1002	-1	0 0 0	3	7
	12	1003	-1	0 0 0	3	11
	. . .	. . .	. . .	. . .	. . .	. . .
	25	2002	-1	0 0 0	12	13
	26	2002	-1	0 0 0	12	14

After all the f-faces have been identified, the next step is to prompt the user for the surface finish attributes for each surface. The surface finish is one of the important factors in selecting the manufacturing process.

Surface Finish Attributes and Tolerances of Hole Diameter Acquisition

The machined parts are assumed to be made from aluminum (356 alloy) through the sand casting. The surface quality of a sand casting part generally varies from 250 to 650 micro-inches without using a special facing sand [82]. In this research a default value 125 micro-inches (AA) is assumed for all the finished surfaces of a part. In PADL-1 the default values of the tolerances of the hole diameter are +0.001 inch and -0.001 inch.

The PADL-1 system does not include the surface finish prompting procedure. An interactive program was developed to ask the designer to enter the surface finish required for the surfaces which require a surface finish other than 125 micro-inches. This procedure also asks for the tolerances of hole diameters when the surfaces are cylindrical.

The designer who uses this interactive program is assumed to be familiar with the PADL-1 system and the PADL-1 coordinate system. To identify a particular face, the user is asked to enter the position of a corner point of the face (if it is a plane face), or the center points and the radius (if it is a cylindrical face), and the type of the face (X, Y, or Z). Then, the designer is asked to enter the surface finish requirements. A flow chart shown in Figure 21 illustrates the procedure for entering the surface finish and the tolerances of the hole diameters. For example, Table IX is a list of steps for updating the surface finish requirements of the two cylindrical surfaces shown in Figure 17.

When this procedure is complete, the contents of the corresponding records in the ROUGHNESS TOLERANCE file are updated. Table X-A is a list of the initial contents of the ROUGHNESS TOLERANCE file for the part displayed in Figure 17. Table X-B shows the contents after updating the surface finish requirements. For example, the surface finish of surface 13 and 14 have been changed to 16 and 63 AA respectively. The size tolerances of the surface 14 has been changed to ± 0.005 inch. Then, FREXPP extracts the form features from the information stored in the updated b-face FACE file and the EDGE file automatically.

Classification of Form Features

Features that can be identified by FREXPP are classified into two groups: cylindrical features and non-cylindrical features. Features of these two groups are further classified as "basic" and "secondary" features.

Basic Features

Basic features are those features that can be recognized directly through the part boundary information. Table XI displays the features categorized in basic features

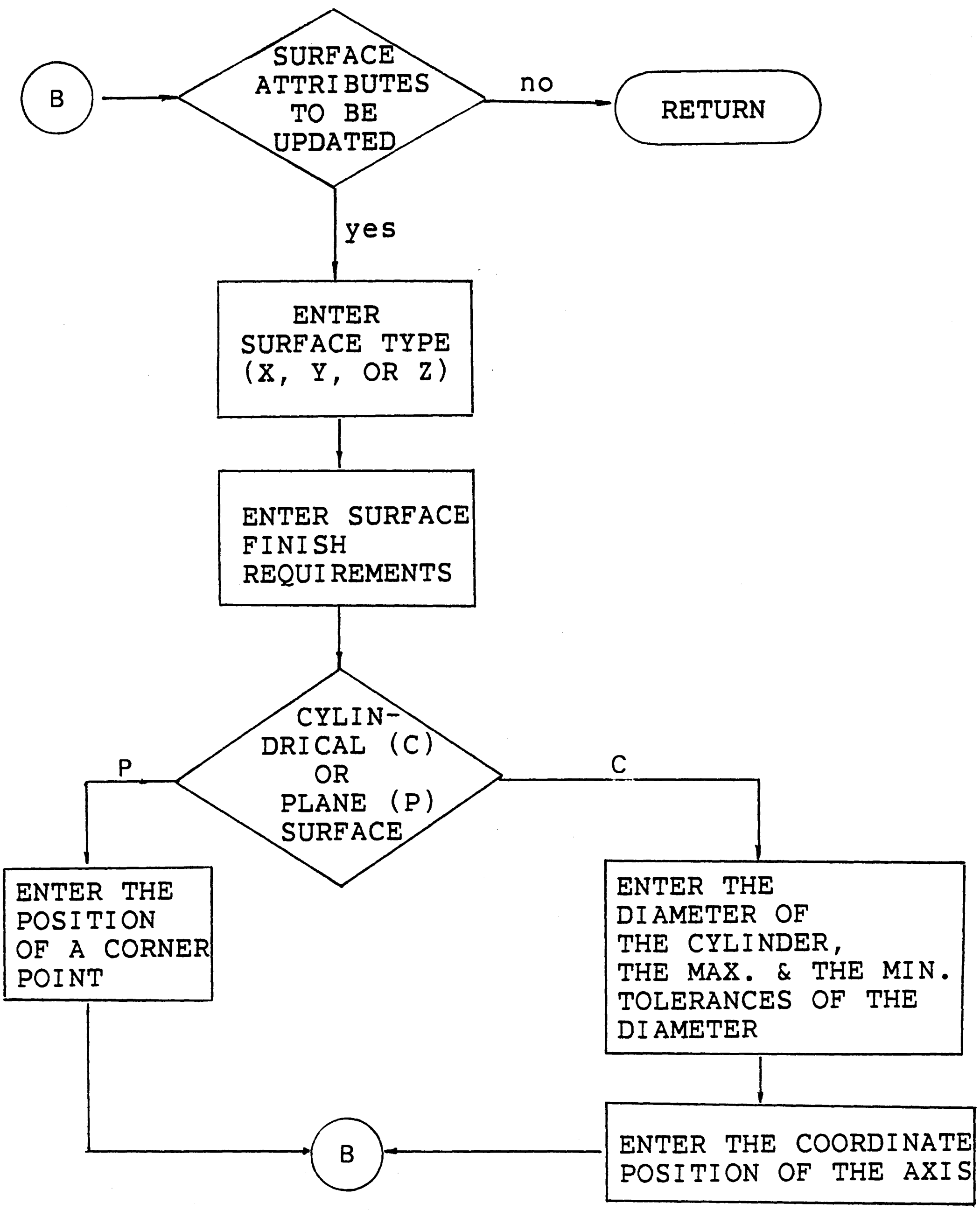


Figure 21. Procedure for Entering Surface Finish and Tolerances

TABLE IX

EXAMPLE OF SURFACE FINISH AND TOLERANCES UPDATING SEQUENCE

```
IS THERE ANY SURFACE ATTRIBUTES TO BE UPDATED?
Y
IS THIS A PLANE SURFACE(P) OR A CYLINDRICAL
SURFACE(C)?
C
ENTER THE SURFACE TYPE (X,Y, OR Z):
Y
ENTER THE REQUIRED SURFACE FINISH (IN MICRO-INCHES)
63
ENTER THE DIAMETER OF THE HOLE (IN INCHES),
THE MAXIMUM AND THE MINIMUM TOLERANCE OF THE HOLE
DIAMETER (IN 0.001 INCHES):
0.625,5.0,0.0
ENTER THE "X" COORDINATE POSITION OF THE AXIS:
2.5
ENTER THE "Z" COORDINATE POSITION OF THE AXIS:
1.5
IS THERE ANY OTHER SURFACE ATTRIBUTES TO BE UPDATED?
Y
IS THIS A PLANE SURFACE(P) OR A CYLINDRICAL
SURFACE(C)?
C
ENTER THE SURFACE TYPE (X,Y, OR Z):
Y
ENTER THE REQUIRED SURFACE FINISH (IN MICRO-INCHES)
16
ENTER THE DIAMETER OF THE HOLE (IN INCHES),
THE MAXIMUM AND THE MINIMUM TOLERANCE OF THE HOLE
DIAMETER (IN 0.001 INCHES):
1.0,2.0,0.0
ENTER THE "X" COORDINATE POSITION OF THE AXIS:
2.5
ENTER THE "Z" COORDINATE POSITION OF THE AXIS:
1.5
IS THERE ANY OTHER SURFACE ROUGHNESS TO BE ENTERED?
NO
```

TABLE X

CONTENTS OF THE ROUGHNESS TOLERANCE FILE

	RECORD(FACE) NUMBER	ROUGHNESS	MAXIMUM TOLERANCE	MINIMUM TOLERANCE
A. INITIAL STATE	1	125	0.00100	-0.00100
	2	125	0.00100	-0.00100
	3	125	0.00100	-0.00100
	4	125	0.00100	-0.00100
	5	125	0.00100	-0.00100
	6	125	0.00100	-0.00100
	7	125	0.00100	-0.00100
	8	125	0.00100	-0.00100
	9	125	0.00100	-0.00100
	10	125	0.00100	-0.00100
	11	125	0.00100	-0.00100
	12	125	0.00100	-0.00100
	13	125	0.00100	-0.00100
	14	125	0.00100	-0.00100
B. UPDATED FILE	1	125	0.00100	-0.00100
	2	125	0.00100	-0.00100
	3	125	0.00100	-0.00100
	4	125	0.00100	-0.00100
	5	125	0.00100	-0.00100
	6	125	0.00100	-0.00100
	7	125	0.00100	-0.00100
	8	125	0.00100	-0.00100
	9	125	0.00100	-0.00100
	10	125	0.00100	-0.00100
	11	125	0.00100	-0.00100
	12	125	0.00100	-0.00100
	13	16	0.00100	-0.00100
	14	63	0.00500	-0.00500
	15	125	0.00100	-0.00100
	16	125	0.00100	-0.00100

TABLE XI

BASIC FEATURES AND SECONDARY FEATURES

Features			
Cylindrical Features		Non-cylindrical Features	
Basic	Secondary	Basic	Secondary
HOLE-1 BOSS SINGLE- STEP BORE	COUNTER BORE BORE-2 BORE-3 BORE-4 BORE-5	BLOCK SLOT STEP POCKET-1 PLANE	PAD GROOVE HOLE-2 T-SLOT

and secondary features. Each basic feature has a unique pattern. A pattern is formed by a "key" surface and the surface normals of the "side" surfaces. A "key" surface is the surface through which the pattern of a form feature can be defined. The identification of key surfaces is discussed in the Key Surface Selection section. The "side" surfaces are the surfaces which share the edges with the key surface. Each surface has its coordinate position along the axis to which the surface is perpendicular. The coordinate position of a surface is called the surface axis offset. The larger the coordinate is, the higher the surface axis offset will be.

For example, in Figure 22 a POCKET-1 feature, surface V is the key surface, surfaces I, II, III and IV are the side surfaces and surfaces III and IV have the higher surface axis offset. The pattern of the rectangular pocket can be described as a key surface (surface V) which is surrounded by four side surfaces (surfaces I, II, III, and IV), and the surface normals of the same type surfaces (I and III are X-type surfaces, II and IV are Z-type surfaces) point toward one another. In Figure 23 a SINGLE-STEP BORE feature, disk face A (key surface) shares two edges with cylindrical surfaces B and C (side surfaces).

Secondary Features

A secondary feature cannot be directly identified through the boundary information. It is formed by the combinations of basic features or basic features with

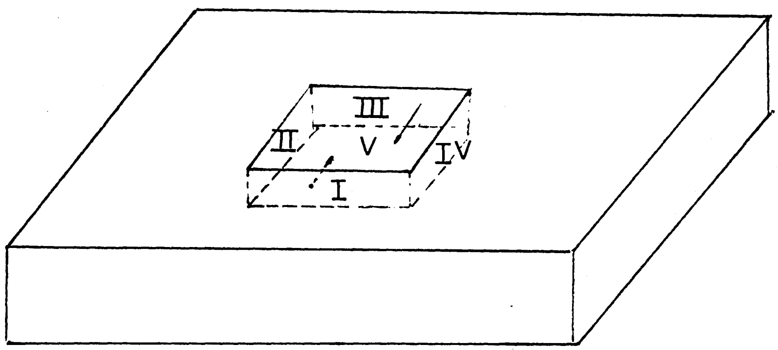


Figure 22. A POCKET-1 Form Feature

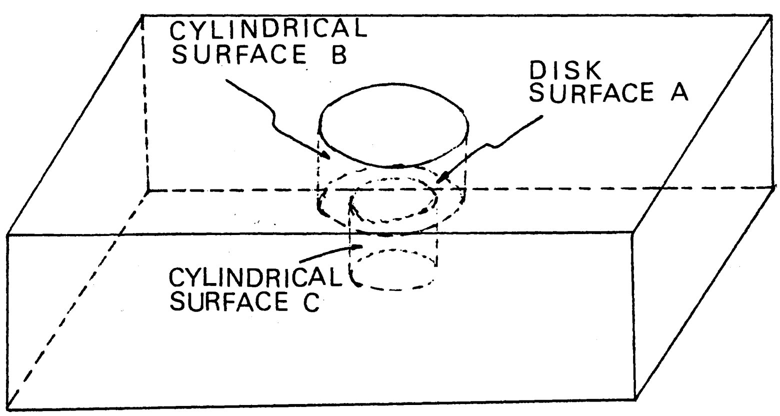


Figure 23. A SINGLE-STEP BORE Feature

surfaces; or the combinations of secondary features. The basic features that are used for constructing the secondary features are called "acting features". These features are SLOT, BLOCK, BLIND HOLE, and SINGLE-STEP HOLE.

For example, a rectangular through hole is defined as the combination of four consecutive rectangular slots. The two side surfaces of each slot are the key surfaces of the other two connected slots. In Figure 24, face II and face IV are the side surfaces of slot 1, and face I is the key surface. Face II and Face IV are the key surfaces of slot 2 and 4 respectively.

A BORE-2 feature is the result of the combination of two SINGLE-STEP BOREs. As proceeding along the axis of the BORE-2 feature, the diameter of the bore must be decreased from the outer-most cylindrical surface of feature. In Figure 25, the BORE-2 feature has a series of cylinders with the decreasing diameters from left to right. The purpose of separating features into the basic and secondary categories is for the design of the recognition sequence of the feature recognition procedure.

Form Feature Hierarchy

As described in the previous section, each basic feature has a key surface from which the pattern of a feature is formed. The secondary features are formed by the combination of basic features. Consequently, some basic features must be identified prior to the identification of the secondary features. Thus, basic features and secondary

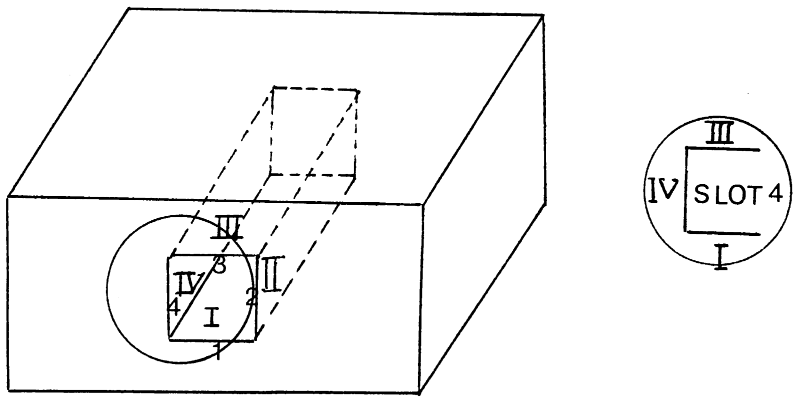


Figure 24. A HOLE-2 Feature

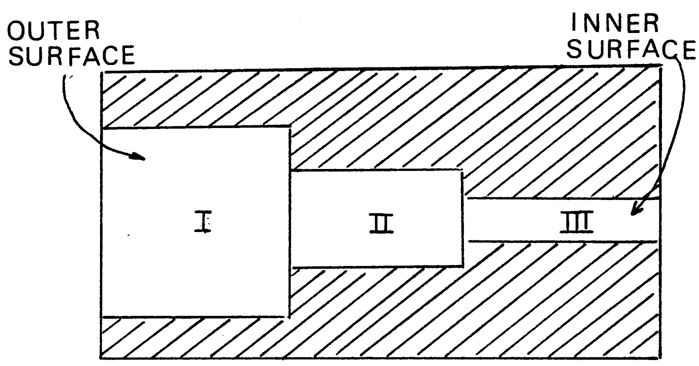


Figure 25. A BORE-2 Feature

features are organized in a hierarchical fashion. Features in the lowest level will be recognized first. Figure 26 shows the form feature hierarchy. The criteria to categorize the features into different levels are:

1. The basic features are ranked lower than the secondary features (level I is the lowest level).
2. The less likely a feature is a part of another feature, the lower the feature is ranked.

For example, a pocket is a basic feature and has no possibility of being a part of another feature; therefore, it is categorized in the lowest level of the form feature hierarchy. A rectangular blind slot can be found in a pocket. It is possible for it to be a part of another feature; therefore, it is in level II. Furthermore, a step can be found in a slot feature; consequently, it has a greater possibility of being a part of another feature than does a slot; therefore, it is in level III. A polygonal plane surface has been defined as a plane form feature. A polygonal plane feature is not formed by the key surface; therefore, it is categorized in the second level.

The hierarchy of the cylindrical type features is defined similarly to the block type features. A through hole can be a part of a SINGLE-STEP BORE; therefore, the level of a SINGLE-STEP BORE is lower than the level of a through hole. The SINGLE-STEP BORE in level I is the acting feature of counter bore, BORE-5, and double-step bore. The double-step bores in level II are the acting features of BORE-2 and BORE-3 in level III.

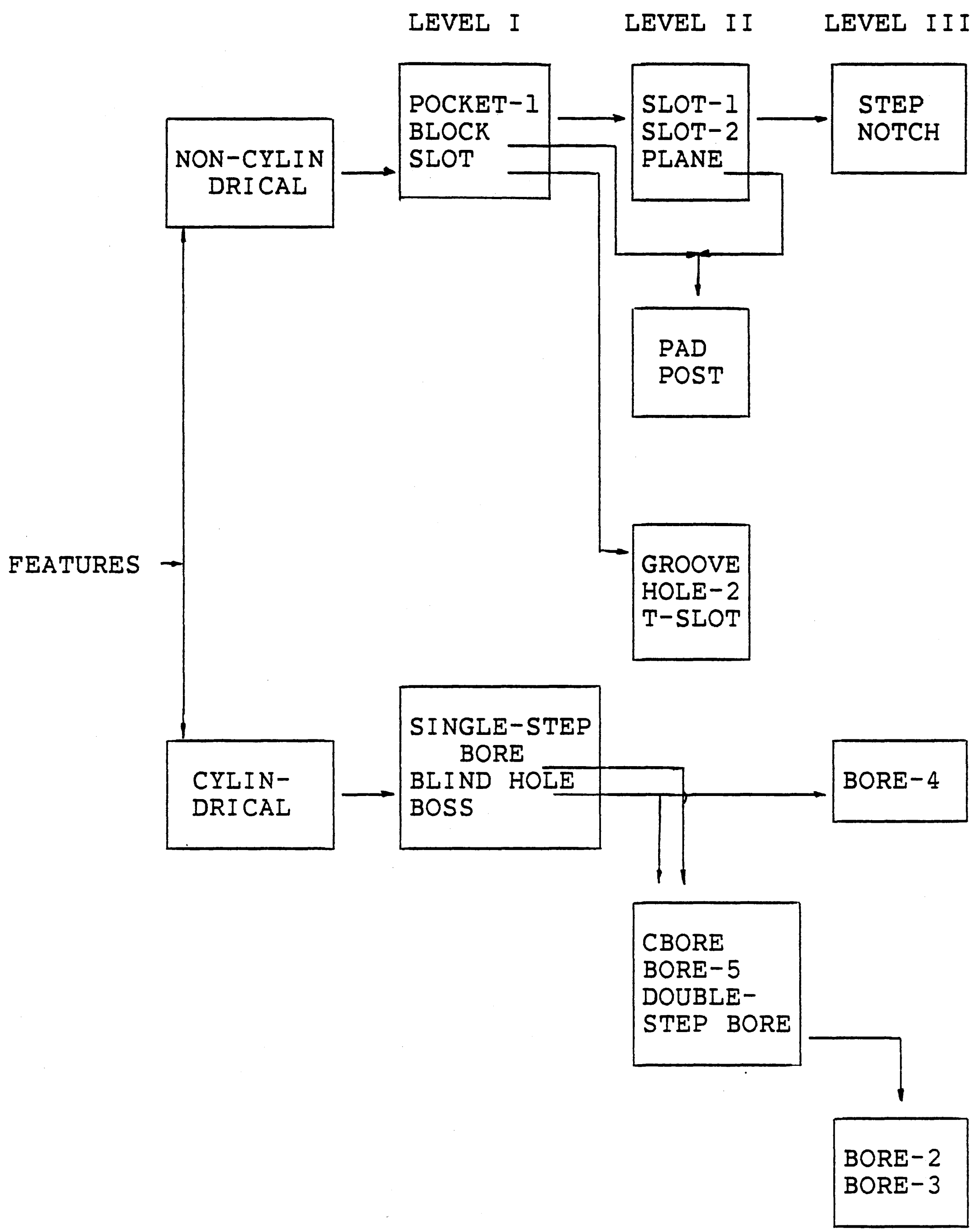


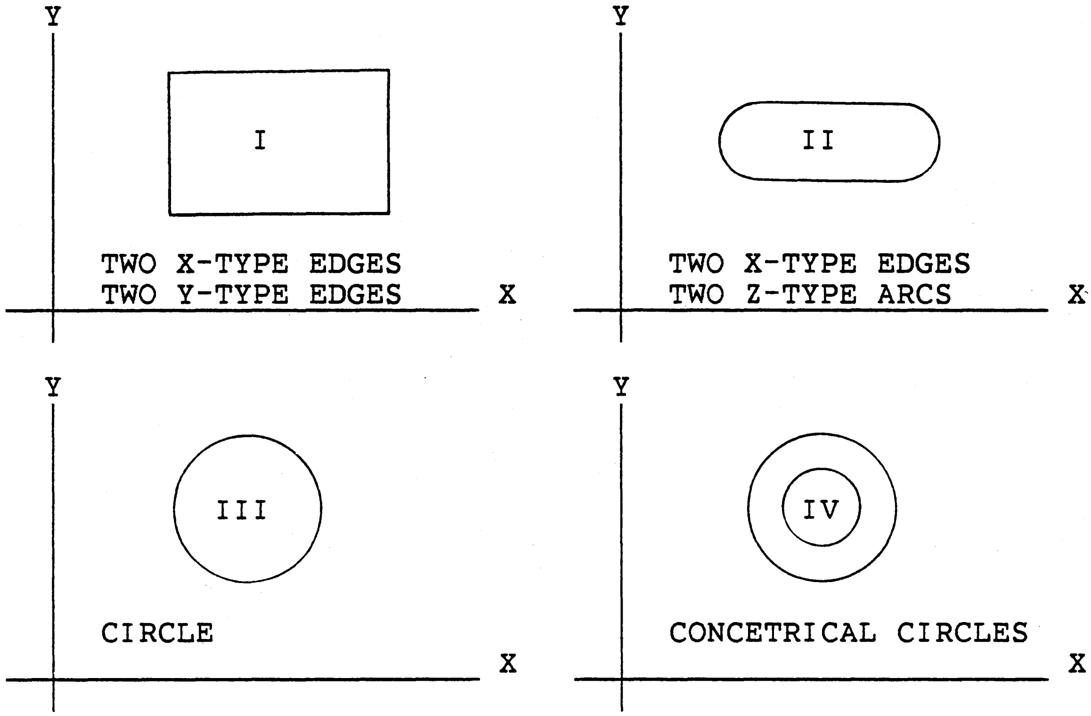
Figure 26. Form Feature Hierarchy

The form feature hierarchy defines the recognition sequence of features. Both the cylindrical and non-cylindrical basic features are classified into two levels. Features in each level are mutually exclusive. The secondary features are formed by the acting features (basic features which are used to form the secondary features) or secondary features. Secondary features are categorized a level higher than their acting features in the form feature hierarchy. For example, SLOT in level I is the acting features for constructing features GROOVE, HOLE-2, and T-SLOT in level II.

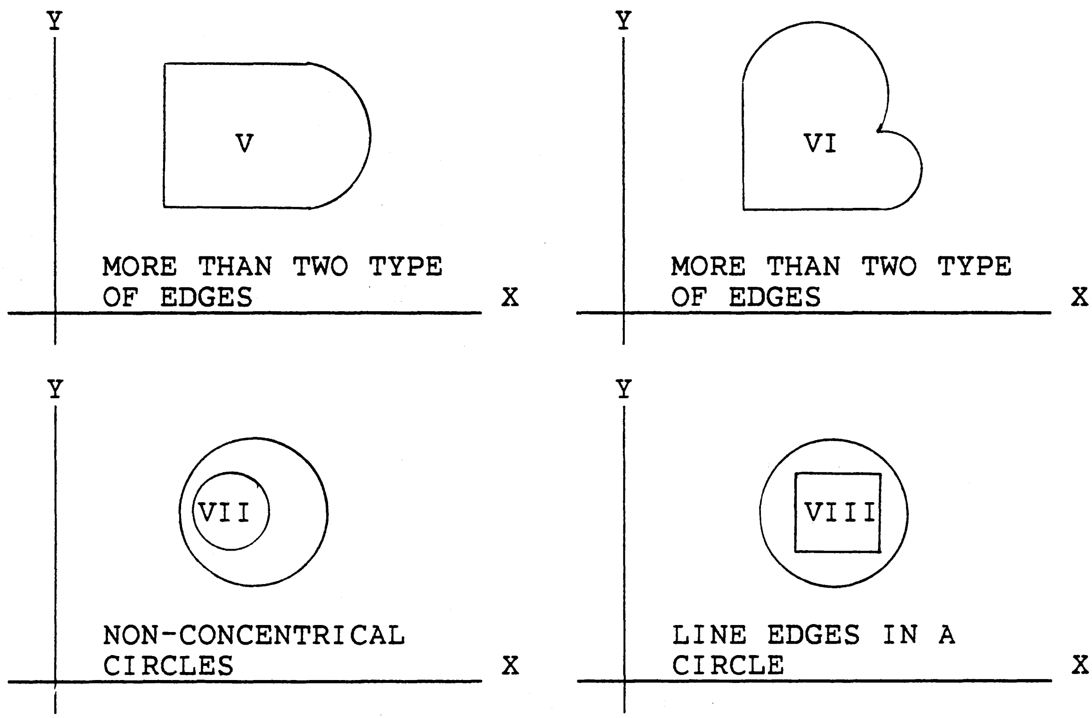
Key Surface Selection

Selecting the key surface is the primary step for recognizing the form features. Key surfaces for non-cylindrical features, such as POCKET, SLOT, BLOCK, and STEP are rectangular surfaces; for a SLOT with round ends is an elliptical surface; and for a NOTCH is a cylindrical surface. Key surfaces for cylindrical features, such as BOSS, BLIND HOLE, and SINGLE-STEP BORE, are disk surfaces; and for a through hole (HOLE-1) is a cylindrical surfaces.

The rule for selecting an eligible key surface for a non-cylindrical feature is that the surface contains only two pairs of edge-types. For example, in Figure 27, surface I contains two X-type LINE edges and two Y-type edges; surface II contains two X-type LINE edges and two Y-type ARC edges. Surface V and VI are ineligible key surfaces because there are more than two type edges.



Eligible Key Surfaces I, II, III, and VI



Ineligible Key Surfaces V, VI, VII, and VIII

Figure 27. Eligible and Ineligible Key Surfaces

The rule for selecting an eligible disk key surface is that the edge(s) of the surface must be concentric circle(s) and the number of edges are 2 or 1. For example, the disk surface of a blind hole is bounded by one circle, such as surface III in Figure 27; the disk surface of a step bore are bounded by two circles, such as surface IV in Figure 27. Surface VII is not a key surface because the two ARC edges are not concentric; surface VIII is not an eligible key surface because the edges are not all circles.

The rule for selecting a cylindrical key surface is that the cylindrical surface is not a side surface of any other cylindrical features. All of the surfaces are ordered according to the number of edges (the least number is ordered first). The sequence of selecting the eligible key surface is based on this ordered sequence.

Procedure for Cylindrical Form Features Recognition

Form feature recognition procedure is based on pattern matching. The cylindrical features are recognized prior to the non-cylindrical features. They are categorized into one of three levels in the form feature hierarchy. The recognition sequence of the cylindrical features is the basic features first and the secondary features next; the low level feature first and the high level feature next.

Cylindrical Basic Features Recognition

The recognition strategy of the basic features is to examine if the pattern formed by an eligible key surface matches the pattern of a certain feature. The control strategy of the recognition procedure is:

1. An unmatched key surface will be examined against all features of the same level before proceeding to the next eligible key surface.
2. All eligible key surfaces will be examined against the features in one level before proceeding to the next level.

The following rules are built for identifying features.

Basic Features of the First Level. The key surfaces for the features of this level are disks. Rules are defined for BOSS, BLIND HOLE, and SINGLE-STEP BORE.

1. If there is a single edge for the key surface and the surface normal of the associated cylindrical surface points away from its axis, then the feature is identified as a BOSS.
2. If there is a single edge for the key surface and the surface normal of the associated cylindrical surface points toward its axis, then the feature is identified as a blind hole.
3. If there are two edges for the key surface, the axes of the two related cylindrical surfaces are the same, and the surface normals of both cylindrical surfaces point toward the axes, then the feature is identified as a SINGLE-STEP BORE.

Basic features of the Second Level. HOLE-1 is the only feature currently classified in this level. The key surface for a through hole is a cylindrical surface. The rule for identifying a through hole is:

4. The cylindrical surface is bounded only by arc edges.

All the identified features are stored in the FEATURE file. The data structure of the FEATURE file is discussed in Appendix B.

Cylindrical Secondary Features Recognition

The secondary features are identified by matching the feature patterns with the patterns that are built by the acting features and their side surfaces. The feature number and the side surfaces of features can be retrieved from the FEATURE file.

Secondary Features of the Second Level. The basic features for constructing the secondary form features of this level are SINGLE-STEP BOREs and BLIND HOLEs. The side surfaces of a SINGLE-STEP bore are ordered by their diameters (high to low) and can be retrieved from the FEATURE file. The cylindrical surface with the large diameter is called the "outer" cylinder. The surface with the small diameter is called the "inner" cylinder. The following rules are developed for identifying the features: CBORE (counter bore), BORE-5, and DOUBLE-STEP BORE.

1. If both the outer cylinder and the inner cylinder

are the side surfaces of only one feature, then the SINGLE-STEP BORE is a counter bore type feature.

2. If the outer cylinder is the side surface of only one feature, and the inner cylinder is the side surface of a SINGLE-STEP BORE and a blind hole, then the combined feature is identified as a BORE-5 type of feature.

3. If the outer cylinder surface is the side surface of only one feature and the inner cylinder is the side surface of two SINGLE-STEP BOREs, or vice versa, then the combined feature is identified as a double-step feature.

The combination of a SINGLE-STEP BORE with any other feature is not a recognized feature in this system. A BLIND HOLE is the basic feature of a BORE-4 or a POCK-2 (circular cavity) type of feature. The condition for distinguishing these two features is based on the ratio of the diameter and the depth of the blind hole. If the ratio is greater than 10, the feature is identified as a circular cavity; otherwise, the blind hole is a BORE-4 type of feature.

Secondary Features of the Third Level. The DOUBLE-STEP BOREs are the acting features of BORE-2 or BORE-3 type features. The cylindrical surface with the largest diameter of a DOUBLE-STEP BORE is called the outer cylinder. The cylindrical surface with the smallest diameter is called the inner cylinder. The rules for recognizing the BORE-2 and BORE-3 features are:

1. If the outer and the inner cylinder of a DOUBLE-STEP BORE are the side surfaces of only one feature, then this feature is identified as a BORE-2 type feature.
2. If the outer cylinders of two DOUBLE-STEP BOREs are the side surfaces of only one feature respectively and the inner cylinders of these two features are the same surfaces, then the combined feature is identified as a BORE-3 feature.

The combinations of a DOUBLE-STEP BORE and any other feature are non-recognizable. After all the possible cylindrical features have been identified, the system starts recognizing the non-cylindrical features.

Procedure for Non-cylindrical Features Recognition

The non-cylindrical form features are categorized into one of three levels in the form feature hierarchy. The recognition strategy and procedure control strategy are the same as defined for cylindrical feature recognition procedure.

Non-cylindrical Basic Features Recognition

Basic features of the First Level. The features which are classified in the first level are POCKET-1 (rectangular cavity), SLOT (an acting feature), and BLOCK (an acting feature). The key surfaces of these three features are rectangular surfaces. The following rules are used to identify such features:

1. If the surface normals of both paired side surfaces "point toward" one another, these surfaces form a POCKET-1 feature.
2. If the surface normals of each paired side surfaces "point away" from one another, these surfaces form a BLOCK feature.
3. If the surface normals of a paired side surfaces "point toward" one another, and the other paired surface normals "point away" from each other, these surfaces form a SLOT feature.

During the recognition of the basic features, surfaces that are related to the recognized low level features are marked and not available for using in the high level features because features are mutually exclusive.

Basic Features of the Second Level. The features which are classified in the second level are: SLOT-1 (through slots with rounded ends), SLOT-2 (rectangular blind slots), and PLANE. The key surface for a SLOT-1 or SLOT-2 features is a rectangular surface. The rules for identifying these two features are:

4. If the key surface is bounded by a pair of plane surfaces and a pair of cylindrical surfaces, the surface normals of the paired plane surfaces "point toward" one another, the surface normals of the cylindrical surfaces point to their axes, and the

boundary surfaces of the paired cylindrical surfaces are the same, then these surfaces form a SLOT-1 feature.

5. If the surface normals of one paired surface "points toward" one another, the other paired surface normals "point toward" the same direction, then these surfaces form a SLOT-2 feature.

A surface whose outer boundary loop has five or more edges is considered as a PLANE feature.

<u>Basic Features of the Third Level</u>. STEP and NOTCH are the features that are classified in the third level of the form feature hierarchy. The key surface of a STEP feature is a rectangular surface and the key surface of a NOTCH feature is a cylindrical surface. The rules for identifying these two features are:

6. If the surface normals of one paired surfaces "point away" from one another, the other paired surface normals "point toward" the same direction, then these surfaces form a STEP feature.
7. If the shared surfaces of the two LINE edges of a cylindrical key surface have the same surface axis offset, and the surface normals of the shared ARC edges point away from each other, then this cylindrical surface forms a NOTCH feature.

Non-cylindrical Secondary
Features Recognition

The non-cylindrical secondary features that can be recognized by FREXPP are classified in the second level. The secondary features, PAD and POST, are recognized as BLOCKs on a PLANE feature. The rules for identifying PAD and POST are stated as follows:

1. A BLOCK is a PAD, if (i) two or more side surfaces of the BLOCK intersect with the plane surface, (ii) the height between the key surface of the block and the plane surface is less than 1 inch and is less than the width of the key surface, and (iii) the key surface of the BLOCK contains no internal feature.
2. A BLOCK is a POST, if (i) two or more side surfaces of the BLOCK intersect with the plane surface, (ii) the height between the key surface of the block and the plane surface is larger than 1 inch, and (iii) the key surface of the BLOCK contains no internal feature.

The secondary features HOLE-2, GROOVE and T-SLOT are derived from the SLOT features. To identify these features, all the acting SLOT features are retrieved from the FEATURE file and the relationship among the slots are studied. The following rules have been set for identifying HOLE-2, GROOVE, and T-SLOT:

3. For a given rectangular slot, if the key surface

of the slot is not a side surface of any other feature, the rectangular slot is a GROOVE.

4. For any given four rectangular slots, if the key surface of each slot is the side surface of two other slots, these four slots form a HOLE-2 feature (rectangular through hole).

5. For a given rectangular slot A, if (i) the side surfaces of slot A are the key surfaces of slots B and C, (ii) one of the side surfaces of slot B and C shares the key surface of slot A, and (iii) the other side surfaces of B and C have the same axis offsets and are the side surface of two blocks, then these five features -- three slots and two blocks -- form a T-slot feature.

The features which can be also recognized in this system are: blind slot with rounded ends, extrusion with rounded ends, single-step boss, non-concentric step bores or bosses, and a hole in a boss or a boss in a blind hole. All of the unidentified plane surfaces are designated as PLANE features, and the cylindrical features are designated as HOLE-1 or BOSS features according to the surface normals of the cylindrical surfaces.

During the process of feature recognition, the records of two files, the FEATURE file and the INTERNAL FEATURE file, are created and updated. The FEATURE file stores all the features recognized in this procedure. Information stored in each record of the FEATURE file includes the feature number, the feature type number, and the names of key surfaces and the side surfaces of the feature. When a feature contains internal features, the INTERNAL FEATURE file is used to store the external feature number, the key surface name of the external feature, and one surface name of each inner loop. This information will be retrieved later for relating the external and internal features together. The data structure of these two files are stated in Appendix B. Table XII displays the contents of the FEATURE file of the part designed in Figure 18, page 85. Table XIII-A displays contents of the first stage of the INTERNAL file. Besides storing information in the FEATURE file, the following internal files are also generated:

1. Key Surface-Feature Number file. In this file, each key surface is associated with a feature number. A number zero indicates that this feature is not a key surface.
2. Feature Number-Feature Type file. This file provides the feature type number when the feature number is given.
3. Feature Number-Key Surface file. The key surface name is provided by this file, if the feature number is given.
4. Side Surface-Key Surface file. Side surfaces are the surfaces which build the feature with the key surface. This file relates a side surface of a feature to its key surface. A surface could be a side surface of different tentative features. A link list is used when more than one feature is

TABLE XII

CONTENTS OF THE FEATURE FILE

FEATURE NUMBER	FEATURE TYPE NUMBER	NUMBER OF SURFACE	KEY SURFACE NUMBER	SIDE SURFACE NUMBER
1	7 (CBORE)	3	12	13 14
2	11 (PAD)	5	9	11 15 7 16
3	18 (PLANE)	5	6	5 15 1 16
4	18 (PLANE)	5	1	6 3 15 5
5	18 (PLANE)	5	5	6 3 1 16
6	18 (PLANE)	1	3	
7	18 (PLANE)	1	16	
8	18 (PLANE)	1	15	

TABLE XIII

CONTENTS OF THE INTERNAL FILE

A. FIRST STAGE	RECORD NUMBER	EXTERNAL FEATURE NUMBER	NUMBER OF INTERNAL LOOPS	KEY SURFACE NUMBER	SIDE SURFACE NUMBER
	1	5	1	5	14
	2	6	1	3	7
	3	8	1	15	3

B. SECOND STAGE	RECORD NUMBER	EXTERNAL FEATURE NUMBER	NUMBER OF INTERNAL FEATURES	KEY SURFACE NUMBER	INNER FEATURE NUMBER
	1	5	1	5	1
	2	6	1	3	2
	3	8	1	15	1

associated with a surface.

These files provide the information among surfaces and features for relating the external and internal features.

Organizing the Features

During the recognition procedure, all the individual features are recognized. However, the relationships among features are not established yet because the external features may be recognized prior to the recognition of their internal features. In this section, the procedure for connecting the related external and internal features is specified.

Each record of the INTERNAL FEATURE file contains the information of a external feature name and surface names. Each surface is related to a feature and is identified through the following rules:

1. If a given surface is a key surface of a feature (this is identified through the Key Face-Feature Number relation), then, this related feature is identified as an internal feature.
2. If a given surface is not a key surface of a feature, the system will then find the key surface through the Side Surface-Key Surface relation. A feature is then related to the given surface through the Key Face-Feature Number relation.

If the related feature is a block, this feature is tested to see if it is a rectangular post or a pad.

When a given surface cannot be related to any feature, this "surface" is assumed to be a flat feature if it is a plane surface; or it is assumed to be a hole feature if it is a cylindrical surface. When all the surfaces in an INTERNAL FEATURE record have been examined, the contents of this record is updated with the external and related internal feature numbers. Table XIII-B displays the contents of the updated INTERNAL file. During this step the identified internal features are marked. The unmarked features are classified as external features. The final results of the feature relation are stored in the INTERNAL EXTERNAL relation file (INOFRL). The data structure of this file is stated in Appendix B. Table XIV displays the final recognition result of the part designed in Figure 18, page 85.

Summary

In this chapter, the form feature recognition procedure of the FREXPP system has been described. Features, according to the shape of the building surfaces, are classified as cylindrical and non-cylindrical. A feature is further classified as a basic feature or a secondary feature according to the relationship among surfaces. A basic feature can be identified through a given surface and the information associated with the given surface. A secondary feature is identified through the known basic features or the known secondary features. The sequence of the features to be recognized is ordered in a form feature hierarchy.

Figure 28 displays the flow chart of the feature

TABLE XIV

A FINAL RESULT OF THE RECOGNIZED FEATURES

FEATURE # 1 HAS THE FEATURE TYPE 7 AND
IS AN INTERNAL FEATURE OF 5 AND 8

FEATURE # 2 HAS THE FEATURE TYPE 11 AND
IS AN INTERNAL FEATURE OF 6

FEATURE # 3 IS AN EXTERNAL FEATURE HAVING THE TYPE CODE 18

FEATURE # 4 IS AN EXTERNAL FEATURE HAVING THE TYPE CODE 18

FEATURE # 5 HAVING THE FEATURE TYPE CODE 18
IS AN EXTERNAL FEATURE CONTAINING THE INTERNAL FEATURE 1

FEATURE # 6 HAVING THE FEATURE TYPE CODE 18
IS AN EXTERNAL FEATURE CONTAINING THE INTERNAL FEATURE 2

FEATURE # 7 IS AN EXTERNAL FEATURE HAVING THE TYPE CODE 18

FEATURE # 8 HAVING THE FEATURE TYPE CODE 18
IS AN EXTERNAL FEATURE CONTAINING THE INTERNAL FEATURE 1

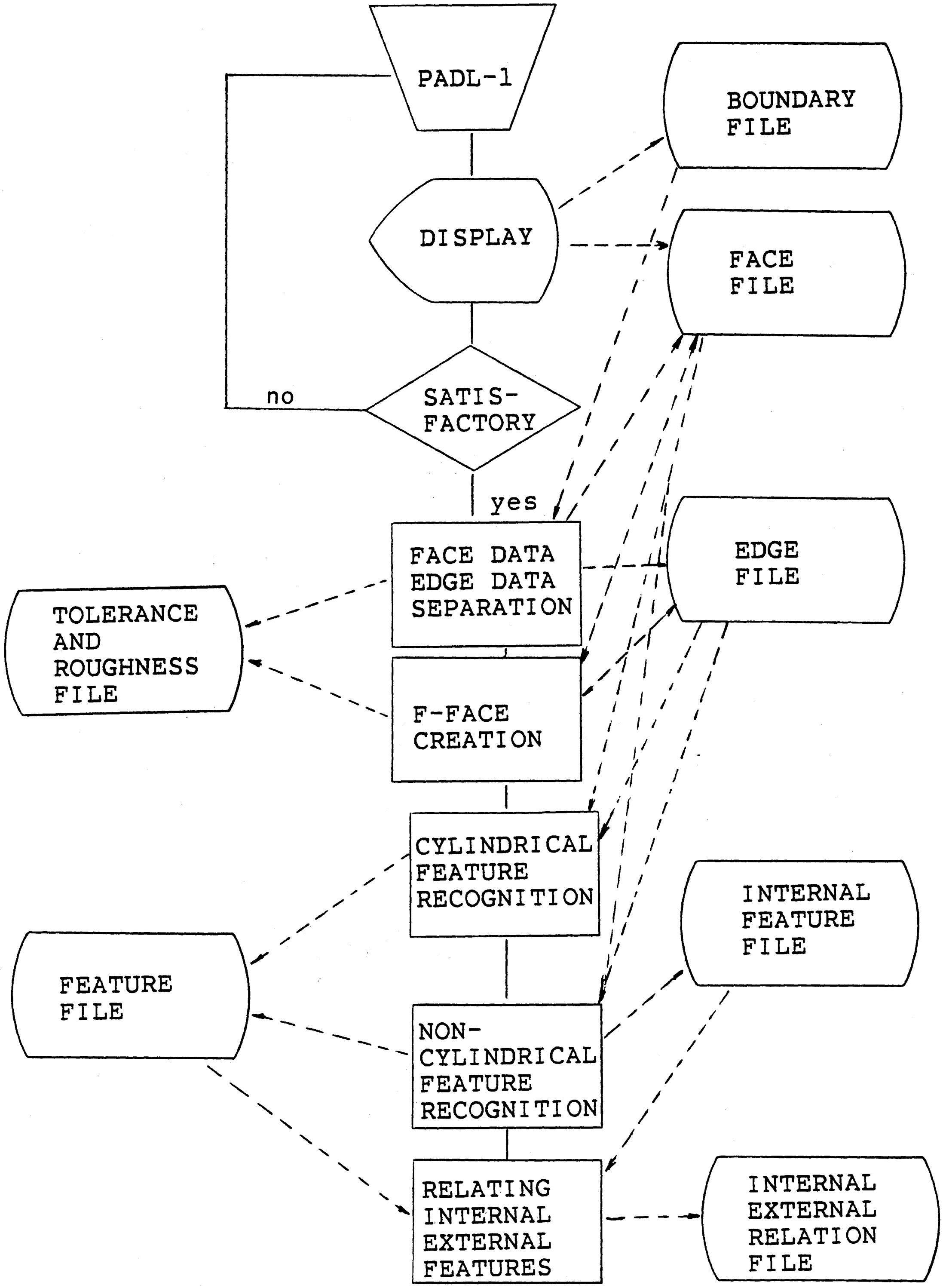


Figure 28. Form Feature Recognition Flow Chart

recognition procedure. It starts by analyzing the boundary representation data. The surface data and the edge data of a part are separated from the boundary representation and stored in different files. The surfaces of form features are then generated from the FACE file and the EDGE file. FREXPP extracts the cylindrical form features first. As all the possible cylindrical features have been recognized, FREXPP starts to extract the non-cylindrical features. The sequence of feature recognition follows the level classified in the form feature hierarchy. The recognization is based on the rules defined for each feature.

During the execution of the feature recognition procedure, a feature file (FEATURE) is used to store all the recognized features and an internal feature file (INTERF) is used to store features containing internal features. A feature is called as an internal feature, if it is bounded by the edges of another surface. Otherwise, a feature is called an external feature. All the cylindrical features, pockets, slots and pads are internal features. When all the features of a part are recognized, the recognition procedure starts to establish the relationship among features. The results are stored in the INTERNAL EXTERNAL relation file (INOFRL). This file will be retrieved later in the expert process planning system.

CHAPTER V

PROCESS PLANNING SYSTEM

Introduction

The metal processing can be categorized into five general areas [21].

1. Basic Process Operations
2. Principal Process Operations
3. Major Operations
4. Auxiliary Process Operations
5. Supporting Operations.

"Basic process operations" are those which produce the initial material shape or form for specific products. Materials, such as sand castings, forgings, bar stock and strip stock are produced from this class of operation.

"Principal process operations" are the manufacturing methods that form or shape the materials into the desired form. They are cutting, forming, casting and molding, and assembling.

"Major operations" are those operations performed within the principal process operations. For example, where forming is the principal process operation, forging, rolling, drawing, piercing, extruding, spinning, and welding are the major operations. Where cutting is the principal process, turning, drilling, milling, broaching and many others are major operations.

"Auxiliary process operations" are those operations that assure continuity and completion of the principal process operations. These operations generally change the appearance and characteristics of the workpiece. Some of the important auxiliary process operations are straightening, clearing, finishing and shot peening.

"Supporting operations" are employed to ensure the successful completion of the product. The major supporting operations are receiving, material handling, quality control, inspection, packing and shipping.

A complete manufacturing sequence for producing a product requires fitting all the operations together. The process planner is primarily concerned with planning the major operations. In this research, it is assumed that the part material comes out of sand casting. The major operation to be considered is cutting. The expert system was designed to select the manufacturing process for machining a given feature.

Sequencing the Features

Surfaces of machining parts are classified as "critical" surfaces and "non-critical" surfaces. Critical surfaces are defined as the surfaces on the part which are best qualified for locating and measuring the part on each of its operations [21]. They are identified through close tolerances, surface finish, and surfaces designated as references for dimensioning. There are two types of critical surfaces, process and product.

"Process critical" surfaces are those surfaces which serve as reference surfaces for the location system. "Product critical" surfaces are those surfaces which are specified by tolerances, surface finish and other geometric attributes. The "process critical" surfaces of a part must be machined prior to the operation of the product critical surfaces. A surface can be identified as a process critical surface and a product critical surface. A feature which is associated with the critical surfaces is called a critical feature.

Reference Surfaces

Since reference surface data are not input to the PADL-1 system, three surfaces are assumed to be the reference surfaces for a part in this system. They are:

1. Reference surface A. It is the outer-most Y type plane surface having a negative surface normal and is not an internal feature.

2. Reference surface B. It is the outer-most X type plane surface having a negative surface normal and is not an internal feature.

3. Reference surface C. It is the outer-most Z type plane surface having a negative surface normal and is not an internal feature.

Figure 29 shows an example of the three reference surfaces of a part. Surface 8 is the outer-most X type

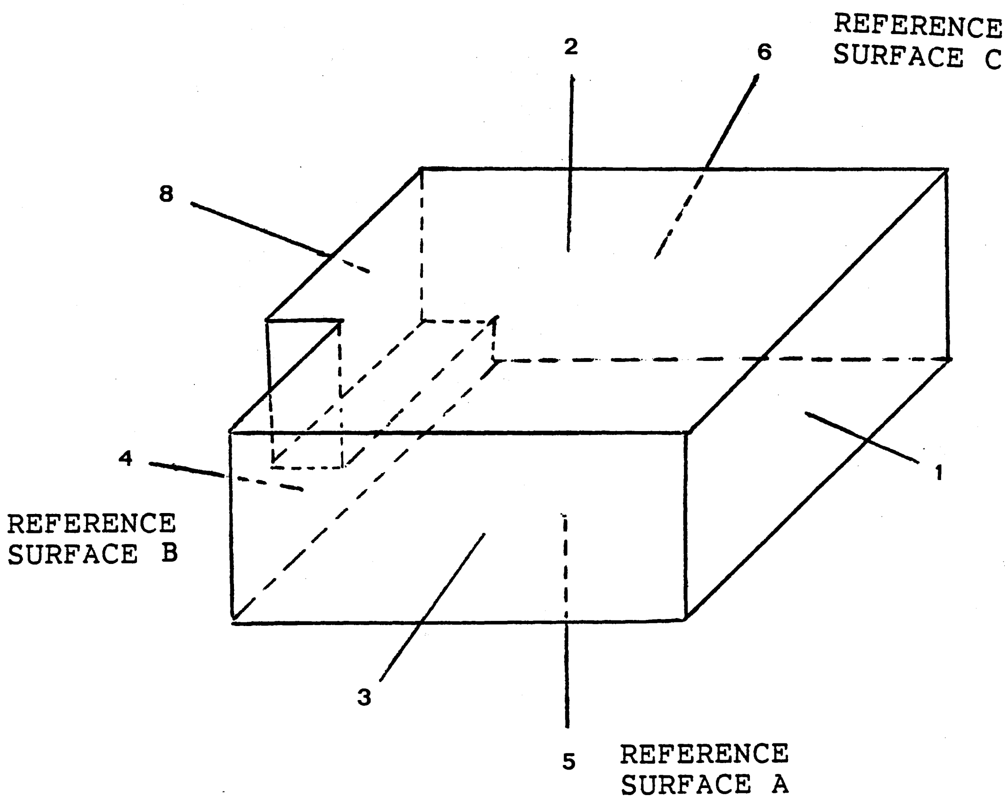


Figure 29. Reference Surfaces

surface, but it is not an external feature. The three reference surfaces are surface 5, 4, and 6.

Feature Sequencing Strategy

The rules of the strategy for sequencing the recognized features are:

- The reference surfaces are ordered prior to other surfaces of the same type.
- The external features are ordered prior to the internal features.
- The plane features are ordered prior to the non-plane features.
- The non-cylindrical features are ordered prior to the cylindrical features.

The following is a list of the feature ordering steps used in the FREXPP system:

1. Select the reference surface A. The feature whose key surface is selected as the reference surface A is ordered.

2. Order the plane features. The plane surfaces whose surface normal directions are opposite to the surface normal direction of reference surface A are ordered.

3. Repeat steps 1 and 2 when the reference surface is B or C.

4. Order the external non-plane features. The surface normal directions of the key surfaces are opposite to the surface normal direction of reference surface C.

5. Order the plane features. The plane surfaces whose

surface normal directions are the same as the surface normal direction of reference surface C are ordered.

6. Order the external non-plane features. The surface normal directions of the key surfaces are the same as the surface normal direction of reference surface C.

7. Repeat steps 4, 5, and 6 when the reference surface is A or B.

8. Order the internal non-cylindrical features. The ordering sequence for these non-cylindrical features are based on the face-type code (in the order of 98, 102, 97, 99, 101, and 103; Table III page 69 displaying a cross reference listing of the face-type code) of the surfaces that share the edges with the internal features. Features, such as SLOT-1, POCKET-1, HOLE-2 and BOSS are selected at this step.

9. Order the internal cylindrical features. The ordering sequence for the cylindrical features are based on the type code (in the order of 103, 101, 97, 99, 98, and 102) of the surfaces that share the edges with the internal features.

When more than two features are selected at each step, the selected plane features and the key surfaces of non-cylindrical features are ordered according to the position of the surfaces. The ordering rules are:

1. The surfaces, which have the surface normal directions that are opposite to the surface normal direction of a reference surface, are ordered from low surface axis offset to the high surface axis offset.

2. The surfaces, which have the same surface normal directions that a reference surface has, are ordered from high axis offset to low axis offsets.

These rules assure that the outer-most features will be machined first. The ordered features are stored in the SEQUENCE-FEATURE file. The data structure of the SEQUENCE-FEATURE file is illustrated in Appendix B. Table XV shows an example of the contents of this file. The record numbers of this file correspond to the sequence number. This file is combined with the FACE file, EDGE file, FEATURE file, and FEATURE-RELATION file to form the general data base for the expert process planning model.

Building the Expert Process Planning Model

The expert process planning model developed in this research selects the manufacturing process for a given feature. FREXPP was developed by using the expert system building tool -- EXPERT [77]. EXPERT is best suited for designing consultation type expert systems; such a system can be used to solve classification type problems. This type of problem is characterized by a predetermined list of potential conclusions from which the program may choose.

Using EXPERT to create a particular expert system is similar to writing a computer program using a special computer language. The EXPERT system has its own syntax. Any standard editor may be used to create and list the file of the expert decision model. Before running a newly

TABLE XV

CONTENTS OF SEQUENCE-FEATURE FILE

RECORD (SEQUENCE) NUMBER	FEATURE NUMBER	KEY SURFACE NUMBER	FACE-TYPE CODE
1	5	5	98
2	8	15	102
3	7	16	99
4	1	1	101
5	3	6	97
6	6	3	103
7	2	9	103
8	1	12	102

created decision model, the edited file has to be compiled by a decision model compiler named XP. The compiled module is then executed by the program EXPERT.

When using the EXPERT system, the major work is to represent the problem solution logic. The result is an expert decision model. An expert decision model consists of three major sections: hypotheses, findings and rules that describe logical relationships among findings and hypotheses.

Representations of Hypotheses

The control strategies used in expert systems are forward chaining and backward chaining. Hypothese are the goals to be inferred by using these strategies. Forward chaining works from facts to hypotheses. It starts with a collection of facets and tries all available rules over and over until a hypothesis is reached. Backward chaining works from hypotheses to facts. This strategy finds rules that demonstrates the hypotheses and then verifies the facts that enable the rule to work.

Hypotheses in the EXPERT system are structured in a taxonomic classification scheme. Two major components, "Taxonomy" and "Process", are under the hypotheses section. "Taxonomy" contains the mnemonics that describe the results of the findings. "Process" contains the mnemonics that describe the suggestions for the findings. Mnemonics are unique names in a decision model and is limited to four characters. Examples of the representations of the hypotheses are shown below, along with their English interpretations.

```
**Hypotheses

*Taxonomy
HOLE  Hole type problem
BORE  Bore type problem
DRIL  Drill type problem
CYLN  Cylindrical type feature
PRHO  Previous hole does not exist

*Process
MILL  Mill
BBOR  Bore
DDRL  Drill
```

where, ** indicates one of the major sections in the EXPERT system and * indicates a subsection.

Representations of Findings

Findings are observations that are important in reaching conclusions. They are responses to questions which can be reported in the form of yes/no, or as a numerical result. Four types of questions: multiple choice, checklist, numerical, and yes/no, are available for obtaining information. The checklist question differs from a multiple choice question in that the choices are not mutually exclusive and more than one may be true [77]. In the designed process planning decision model, only two types of questions, multiple choice and numerical, are used for requests. Information to be obtained for the expert process planning system is: the type of features, diameter of the holes, tolerances of the hole diameter and the roughness of the surfaces. The representation of these findings is shown as follows:

**Findings

*Numerical
DIAM Diameter of the hole
DEPT Depth of the hole
XTOL Maximum length of the hole diameter
STOL Minimum length of the hole diameter
ROUF Roughness of the surface

*Multiple Choice
Type of feature
FHOL Hole
FBOR Bore
FNCL Non-cylindrical features

*Functional findings
DTOL = XTOL - STOL
DIDE = DIAM / DETH

where, ** indicates one of the major sections in the EXPERT system and * indicates a sub-section.

Functional findings are simple mathematical formulas used to calculate values from other numerical findings. Functional findings simplify the user's input work, because calculation is not required.

Representations of Rules

Rules are the representations of the experts' decision logic. The decision rules are expressed by the mnemonics. These mnemonics are defined in the hypotheses and findings. Rules relate the observations and the conclusions. Three types of rules are used in the EXPERT system for describing the relationship among findings and hypotheses. They are:

FF -- finding to finding rules
FH -- finding to hypotheses rules
HH -- hypothesis to hypothesis rules

The major purpose of FF rules is to control the sequence of the questions to be asked. This type of rule is not implemented in the expert system developed, because the possible question sequence is unknown.

FH rules are the most important rules for constructing the expert system decision model. They are logical combinations of findings which indicate confidence in the confirmation or denial of the hypotheses.

For example, if the feature is cylindrical, the surface finish requirement is between 32 AA and 63 AA, the diameter is between 1 inch and 2 inches, then the hole should be reamed with 0.6 confidence. When using the EXPERT syntax, the rule can be expressed as follows:

```
F(CYLN,T) & F(ROUF,32:63) & F(DIAM,1:2)-> H(REAM,0.6)
```

HH rules relate hypotheses to other hypotheses. An HH rule is called a table which is stated in two parts: an *IF part and a *THEN part. In the *THEN part, several rules may be described and evaluated. The *IF part sets the context for when the set of rules in the *THEN part should be evaluated. HH rules are evaluated in the order of appearance in the model file. They are usually used for relating the hypotheses in *Taxonomy to the hypotheses in *Treatments and generate the final results for a particular case. For Example,

```
*HH RULES

*IF
[1: F(HOLE,T), F(BORE,T)] & H(PRHO,0)

*THEN
H(BORE,0.01:1.0)->H(BBOR,0.70)
H(DRIL,0.01:1.0)->H(DDRL,0.75)
*END
```

where, ** indicates one of the major sections in the EXPERT system and * indicates a subsection.

This HH rule is read as: If the feature type is a hole or bore, and the pre-hole is not required, then it is concluded that if the bore type problem has 0.01 to 1.0 confidence level, then bore the feature with 0.70 confidence level, if the drill type problem has 0.01 to 1.0 confidence level, then drill the feature with 0.75 confidence level.

Confidence level measures are assigned on a scale of -1 to 1, with 1 being complete confirmation and -1 being complete denial. The confidence level of each rule is subject to the expert who designed it. They may not be universally agreed upon, but they can be modified to satisfy the requirements of different companies without much difficulty.

The complete process planning decision model is listed in Appendix E. The expert knowledge for machining a single hole is primarily derived from the TOM knowledge base. The primary machining process for non-cylindrical features is "milling". This assumption simplifies the procedure of a selecting process for generating plane surfaces and features with plane surfaces.

Questioning Strategy

The rule interpreter of the EXPERT system applies the forward chaining strategy to control the reasoning task. Forward chaining strategy starts with a collection of facts and tries all available rules over and over, until either a goal state is reached or no applicable rules are found.

When running EXPERT, the questions are determined heuristically. Information in response to these question is entered by the user. The questioning strategy that EXPERT currently employs to select questions follows the following criteria:

1. Least costly question: Because questions may involve different degrees of risk or cost, the EXPERT system allows the designer to affix a cost/value to each type of findings. For example, a check list question having a value 3 is expressed as *Checklist/cost=3. Questions of lesser cost are considered by the questioning strategy before those of higher cost. Cost value is not an exact measure of magnitude of cost or risk, but rather a relative ordering on the questions. The cost argument is optional and default value is 1 in the EXPERT system for the questions whose cost argument are not assigned.

2. Highest weighted hypothesis: The use may assign a weight to each hypothesis to indicate the frequency of the occurrence relative to other hypotheses.

3. Hypotheses which are related to some finding results: The FH rules have the higher priority than the HH rules.

4. Confidence Level increase: Findings which can potentially increase the maximum absolute value of the confidence level of a hypothesis will be asked first. For example,

```
F(ROUF,32:125) & F(DIAM,1:2)->H(REAM,0.6)
F(FBOR,F)->H(BORE,-1)
```

The finding which is associated with the second rule will be asked first, because this rule increases the absolute value of the confidence level of a hypothesis higher than the first rule does.

Modifying the EXPERT System

Although the questioning strategy is heuristic, the variables which store the findings are fixed. Each finding mnemonic in the *Numerical section contributes a question, and all finding mnemonics in the *Multiple Choice section belong to one question. A multi-dimensional array FIND is used to store all the finding information. The first mnemonic in the *Finding section is linked to the first dimension of the array FIND. All the other mnemonics are sequentially linked to the array FIND.

The EXPERT system weights the rules and selects a rule to be asked according to the above questioning strategy. The finding mnemonics in the selected rule are evaluated and linked to their dimensional numbers in the array FIND. Then, the proper question in English is printed on the screen of a terminal and the EXPERT system waits for the answer from the user. However, the objective of this research is to build a system that links the CAD and CAM without the intervention of human beings. In order to make the EXPERT system able to fetch the information from a data file automatically, several modifications have been made in the EXPERT system.

1. The I/O control unit has been shifted from the

interactive mode to the disc access mode.

2. A direct access file (FEINPT file) is provided for storing the facts (information about a feature). The record number of this file corresponds to the question numbers in the EXPERT system. Facts about a feature which are stored in the FEINPT file are:

1. The surface roughness
2. The diameter of a hole
3. The depth of a hole
4. The maximum tolerance on the diameter
5. The minimum tolerance on the diameter
6. The feature type number

Each piece of these information is stored in an individual record. Records 2 through 5 are designed for the hole and bore types of features.

A computer program cannot provide the right information unless the computer memory address that stores the information has been specified. The requirement for designing the **Finding section is that the *Numerical sub-section must be arranged ahead of the *Multiple Choice section. The finding mnemonics in the *Numerical sub-section must be arranged in the sequence of the roughness of the surface, the diameter of the hole, the depth of the hole, the maximum tolerance of the hole, the minimum tolerance of the hole, and the feature type number. The finding mnemonics in the *Multiple Choice sub-section are FHOL (through hole), FBOR (hole bottom flat), and FNCL (Non-cylindrical features). All of the cylindrical features can be decomposed to either a through hole or a hole with flat bottom.

Executing the Process Planning Expert System

The process planning expert system is a specific decision-making program for applying the process planning model to different form features. The program begins by reading the process planning model (knowledge base) file name. This file name was entered through the file-name describing program (FILECR) before running the process planning expert system. The information of the decision model was compiled by the XP program prior to the use by the expert system. The XP system indicates any errors found in the decision model and prints the errors upon discovery, allowing the user to locate approximately where the error occurred.

The expert system, then, gets the features to be processed one at a time from the sequenced feature file. The feature name will be printed on the screen and stored in the process plan file. The facts about a feature are assembled from the feature file, face information file, and roughness and tolerance file. The assembled data are stored in the direct access file (FEINPT). The expert system then goes into questioning mode and asks questions that are heuristically generated from the decision model. The answers to these questions are obtained from the FEINPT file. At the conclusion of questioning, the system prints a summary of findings and ordered list of suggested manufacturing processes. The expert system stores the top list of the suggested manufacturing processes in the process plan file (PROCEF).

After printing the summary and conclusions, the expert system goes into command mode, where the user may enter commands to perform various tasks. The list of commands is shown in Table XVI. The user can always find out which commands are available and their formats by typing a "?". The "NEW" command asks the system to process the next feature. The system stops running when all the features have been processed or can be stopped by using the "QUIT" command.

Table XVII shows an example of how the EXPERT system responds to the questioning commands. Before handling the next feature, the system stores the first suggested process into the EXPERT PROCESS file. The data structure of the EXPERT PROCESS file is explained in Appendix B. The system stops automatically as all the features have been processed. An output formater (PLAN) may be used to produce a hard copy of the generated process plan at any time. Table XVIII shows the final process plan of the part displayed in Figure 3 page 12. Table XIX lists the process plan designed for the part displayed in Figure 6 page 16.

Summary

In this chapter, the feature sequencing strategy and the method for generating a process planning decision model

TABLE XVI

COMMANDS FOR USE IN COMMAND/
QUESTION MODES

ASK	Returns EXPERT to questioning mode; Will go on to print the summary and interpretation
ASKU	Asks all unknown/unasked questions
DX	To print interpretations
FIND	To print a complete list of findings, with mnemonic and status of each question
FIND(mne)	Prints this finding and its status
FIND(a:b)	Prints findings in sequential order from mnemonic a through mnemonic b
FIX mne	To revise or set the value of finding mne
FIX n	To change the value of question n
HELP or "?"	To print list of available commands
HYPO	Explanation of reasoning for highest ranked conclusion
HYPO(mne)	To print explanation of hypothesis mne
HYPO(a:b)	Explanation for mnenonic a of hypothesis through mnemonic b of hypothesis
NEW	To go on to another feature
QUIT	To stop the program (also Q)
RULE	To print all FH decision rules in English
RULE(n)	To print rule n
RULE(n:m)	To print this range of rules
SUM	To print a summary of findings
TABLE	Prints all HH rule tables
TABLE(n)	Prints HH rule table n
TABLE(n,m)	Prints table n, rule m

TABLE XVII

SAMPLE OUTPUTS OF USING QUESTIONING COMMANDS IN FREXPP SYSTEM

```
TURN OVER PART TO Y+ FACE -- FEATURE TYPE NUMBER IS 7

SUMMARY

    Roughness of the surface 63
    Diameter of the hole (in inch) 0.62
    Maximum length of the hole diameter(in 0,001") 630
    Minimum length of the hole diameter(in 0,001") 620

    Type of feature
        A Through Hole

INTERPRETIVE ANALYSIS

  Feature Analysis:
    0.82   DRILLING TYPE PROBLEM

  Process Suggestions:
    0.73   Drill

Command Mode: FIX x, HYPO, NEW, ASK, QUIT,... ? for HELP
:HYPO

BORING TYPE PROBLEM was set by rule 4 in this manner:
  Roughness of the surface  (63)
  Diameter of the hole  (0.62)
  <(XTOL-STOL)>  (10)

  ---->  BORING TYPE PROBLEM

  Would you like further HYPO information?   *YES

  Direct confidence weight:                      0.800
  Final Weight                                   0.822

  Weights implied by the taxonomy:
      Forward (from predecessors) positive:    0.001
      Forward (from predecessors) negative:    0.000
      Inverse (from successors):               0.000

  Evidence for this hypothesis can be found in:
    1 Associative rules directly
    0 Associative rules implied by the Taxonomy

  Rule setting the hypothesis in full:
  FH-Rule  4:
```

(Continued)

```
  Y  Roughness of the surface
          ROUF must have a value between 63 and 250;
          its value is 63
  Y  Diameter of the hole
          DIAM must have a value between 0.01 and 1.2;
          its value is 0.62.
  Y  <(XTOL-STOL)>
          DTOL must have a value greater than 8;
          its value is 10

  Y -->  DRILLING TYPE PROBLEM
         DRIL may be set to 0.8 by this rule

  Rules Not Setting Hypothesis: DRILLING TYPE PROBLEM
  FH-Rule  3:

  Y  Roughness of the surface
          ROUF must have a value between 63 and 250;
          its value is 63
  Y  Diameter of the hole
          DIAM must have a value between 0.01 and 1.2;
          its value is 0.62.
  Y  <(XTOL-STOL)>
          DTOL must have a value greater than 8;
          its value is 10

  Y -->  DRILLING TYPE PROBLEM
         DRIL may be set to 0.85 by this rule

Command Mode: FIX x, HYPO, NEW, ASK, QUIT,... ? for HELP
:NEW
```

<u>SUMMARY</u>

```
    Roughness of the surface 16
    Diameter of the hole (in inch) 1.00
    Maximum length of the hole diameter(in 0.001") 1001
    Minimum length of the hole diameter(in 0.001")  999

    Type of feature
        A Non-through Hole
```

<u>INTERPRETIVE ANALYSIS</u>

```
  Feature Analysis:
    0.82   BORING TYPE PROBLEM

  Process Suggestions:
    0.73   BORE

Command Mode: FIX x, HYPO, NEW, ASK, QUIT,... ? for HELP
:QUIT
```

TABLE XVIII

PROCESS PLAN FOR PART A

SET PART TO X-Z FACE (Bottom)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO X-Z FACE (Top)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO Y-Z FACE (Left End)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO Y-Z FACE (Right End)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO X-Y FACE (Back)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO X-Z FACE (Front)		
Rough Mill, Finish Mill	PLANE	FEATURE
Rough Mill, Finish Mill	PAD-1	FEATURE
SET PART TO X-Z FACE (Top)		
DRILL	CBORE	FEATURE
THE CYLINDER DIAMETER IS 0.625 INCH		
BORE	CBORE	FEATURE
THE CYLINDER DIAMETER IS 1.000 INCH		

* PART A is displayed in Figure 3 page 13.

TABLE XIX

PROCESS PLAN for PART B

SET PART TO X-Z FACE (Bottom)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO X-Z FACE (Top)		
Rough Mill, Finish Mill	PLANE	FEATURE
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO Y-Z FACE (Left End)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO Y-Z FACE (Right End)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO X-Y FACE (Back)		
Rough Mill, Finish Mill	PLANE	FEATURE
SET PART TO X-Y FACE (Front)		
Rough Mill, Finish Mill	PLANE	FEATURE
Rough Mill, Finish Mill	PAD-1	FEATURE
SET PART TO X-Z FACE (Top)		
Rough Mill, Finish Mill	STEP	FEATURE
Rough Mill, Finish Mill	STEP	FEATURE
Rough Mill, Finish Mill	GROVE	FEATURE
Rough Mill, Finish Mill	SLOT-1	FEATURE
BORE	CBORE	FEATURE
THE CYLINDER DIAMETER IS	3.0	INCH
BORE	CBORE	FEATURE
THE CYLINDER DIAMETER IS	3.5	INCH
DRILL	CBORE	FEATURE
THE CYLINDER DIAMETER IS	0.8	INCH
DRILL	CBORE	FEATURE
THE CYLINDER DIAMETER IS	0.8	INCH

* PART B is displayed in Figure 6 page 15.

are described. The feature sequencing strategy basically was designed to process plane features first, then in the sequence of external features, internal non-cylindrical features, and internal cylindrical features. The implementation of this strategy is through several FORTRAN programs.

The process decision model contains the rules for selecting the manufacturing process for a given feature. The EXPERT consultation system is used to design the expert process planning system. Modifications to the EXPERT system have been made so that the input data can be automatically read in from a file. Figure 30 depicts the flow chart of the expert process planning system.

This system gets the information from the feature relation file and the FACE file to create a feature sequence file (FINALF). The system then starts to process features in FINALF one at a time. Information describing a feature is retrieved from the ROUGHNESS TOLERANCE file, FEATURE file, FACE file, and EDGE file and is placed in the feature information file (FEINPT). The sequence that the EXPERT system gets the information from the FEINPT file is based on the decision logic stored in the knowledge base.

When all the observations are obtained, the system summarizes the observations and makes suggestions. The user is allowed to enter any modification at this stage. The highest suggestion is then stored in the process file (PROCEF). The system stops processing as all the features have been studied. A computer program (PLAN) can be used to generate the hard copy of a process plan at any time.

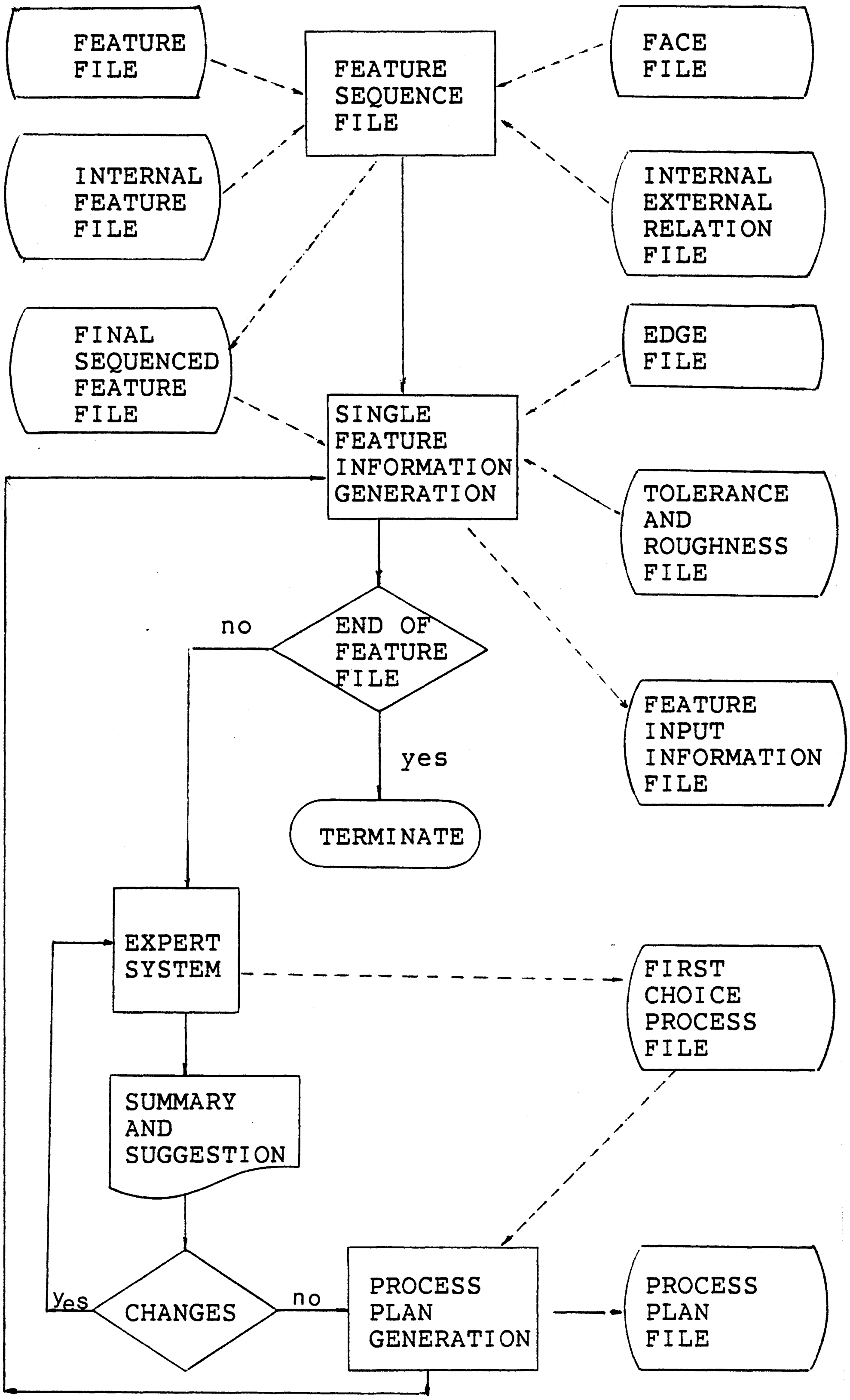


Figure 30. Process Planning System Flow Chart

CHAPTER VI

SUMMARY AND CONCLUSIONS

CAD and CAM are the most advanced technologies developed in today's manufacturing industry. A computer integrated manufacturing system, which integrates CAD, CAM and computerized manufacturing management systems, such as material requirement planning system, capacity planning system, and etc., is the ultimate goal in manufacturing automation. Such an integrated system requires no human intervention to link the design and manufacturing functions. Once an object is designed through a CAD system, the design data is passed to the CAM system for manufacturing. The purpose of this research has been to:

1. Develop an elementary expert system that extracts the form features from the part design data of a solid modeling system.

2. Provide an interactive procedure that prompts the user for the surface finish attributes of each machining surface and the tolerances of the hole diameters.

3. Provide a procedure that orders the recognized form features.

4. Provide a procedure that transfers the ordered form features, surface attributes and tolerances of hole diameters to the input data file of the expert process

planning system in a workable data format.

5. Represent, organize and store the knowledge of manufacturing in the knowledge base.

6. Build a process planning expert system by modifying the EXPERT system.

The boundary representation data is created by the PADL-1 solid modeling system. The reason that PADL-1 is chosen is that it is publicly available and it has the dimensioning capability to express the dimensions on the engineering drawings. The reason that a solid modeling system is chosen is that this type of systems requires less input data and has more flexible ways to describe a part than any other types of systems. Finally, the reason to use the boundary representation data is that this representation is standardized and has the potential to be linked to sculptured surface data.

The parts considered in this research are limited to those that can be constructed by PADL-1. In addition, this class of parts is reduced to those:

1. Consisting of primitive blocks and the form features described in Appendix D.
2. That do not have a form feature intersecting the intersection of two primitives, such as parts illustrated in Figures 4 and 5.
3. Made from cast aluminum (356 alloy).

FREXPP (Feature Recognition and EXpert Process Planning) is a prototype automated process planning system. The unique characteristics of this system are:

1. FREXPP can automatically extract the form features of a part from the boundary representation data of the PADL-1 solid geometric modeling system.
2. FREXPP automatically generates process plans from the recognized features.
3. FREXPP is an interactive process planning system. The user can change the parameters of a feature, such as feature type, surface finish, and tolerances so that various process plans may be generated and a desired plan can be selected.

The main objective of this research is to develop an approach through which the design data of a solid modeling system can be automatically transferred to a CAM system. FREXPP demonstrates that the computer-aided process planning can be integrated with computer-aided design.

The automation of feature recognition from the solid representation eliminates most of manual input required by current computer-aided process planning systems. An additional advantage of the expert system approach is that it should be possible to transplant this prototype system to different companies, because the expert system will permit process planners to modify the manufacturing decision logic contained in the system without affecting the system structure.

Process planning is a multi-dimensional problem. The basic components are part type, material and manufacturing process. Each of the components has several sub-components, and each sub-component has several sub-sub-components. For example, milling is a sub-component of the manufacturing process, and face milling is a sub-component of milling; aluminum is one type of material, alloy 356 casting is a sub-type of aluminum; a part can be classified as a rotational or a non-rotational part. A non-rotational part may have several features.

These three basic components are interrelated. Material type will affect the selection of machine tool, speed, and feed rate. A certain type of machine can generate certain types of features. In this research, we have limited not only the part to certain features, but also the manufacturing processes to milling, center drilling, drilling, boring, and reaming and the material to aluminum 356 alloy. The process plans generated by the system are rough process plans (a routing file). It is believed that once the routing file is built, the detailed process plan can be established. For future research, the following items are recommended:

1. Form features. The types of the form features for the future extension can be classified in two categories: (i) form features with cylindrical surfaces or plane surfaces that are angular to the three coordinate planes, and (ii) form features with the surfaces constructed from primitives other than cylindricals or planes.

An advanced solid modeling system is required to fulfill this extension. PADL-2 with the rotational capability and more primitives (cones, wedges) is a potential candidate.

2. Form feature recognition procedure. The form feature recognition procedure in this research is implemented by the conventional programming technique. However, the recognition strategies are composed of several rules. This procedure can be remodeled through the expert system approach.

3. Part manufacturing information prompting system. Process planning is a complicated problem. Several stages of decisions have to be made before obtaining a final process plan. To make decisions requires proper information. Currently, the developed geometric modeling systems do not include the capability for prompting all of the manufacturing information. An information prompting system that is assisted by a graphical system is suggested. The proposed system would display each recognized feature with special shading and prompts for the information of each feature, such as critical feature, surface attributes, dimension and geometric tolerance information.

4. Process planning knowledge base expansion. In this research, expert knowledge has been used for selecting the manufacturing process for certain types of features, material, and manufacturing processes. For future study, the knowledge base may be expanded for form feature sequencing, machine type selection, manufacturing process selection for different material, and fixture selection.

5. The expert system. The EXPERT consultation system is used to develop the manufacturing decision model. EXPERT is based on the FORTRAN computer language. The expert

knowledge needs to be transformed into a knowledge base by a knowledge engineer or a person who is familiar with the EXPERT terminology. For a "user friendly" system, a list type language, such as LISP or PROLOG, would be a better language to use.

In this research, the form features recognized by FREXPP are clearly defined. The recognition procedure does not recognize compound features (the combination of two or more defined features). To develop a recognition procedure using the expert system approach for compound features would be a challenge for future study.

BIBLIOGRAPHY

1. Alting, L. Manufacturing Engineering Processes. New York: Marcel Dekker Inc., 1982.

2. Barr, A., E. A. Feigenbaum. The Handbook of Artificial Intelligence Volume I. Palo Alto, California: HeirisTech Press, 1981.

3. Bishop, A. B., R. A. Miller. "IE's Expertise is well Suited to Role of Integrating CAD and CAM." Industrial Engineering, 13, 11 (November, 1981), pp. 126-129.

4. Bjorke, D. "Investigation of Heuristic Approaches to Process Planning." (Unpub. paper presented to CAM-I, Inc., June, 1982.) Arlington, Texas: CAM-I, Inc., 1982.

5. Boothroyd, G. Fundamental of Metal Machining. London: Edward Arnold Ltd., 1965.

6. Bradley, J. File and Data Base Techniques. New York: CBS College Publishing Co., 1981.

7. Brown, C. M., A. A. G. Requicha, H. B. Voelcker. "Geometric Modeling Systems for Mechanical Design and Manufacturing." ACM Annual Conf. Proc. (December, 1978), pp. 770-777.

8. Bullers, I., S. Y. Nof, A. B. Winston. "Artificial Intelligence in Manufacturing Planning and Control." AIIE Transactions, 12, 4 (December, 1980), pp. 351-363.

9. CAM-I's Automated Process Planning Users Manual. Arlington, Texas: CAM-I, Inc., 1976.

10. Chang, T. C., R. A. Wysk. "An Integrated CAD/Automated Process Planning System." AIIE Transactions, 13, 3 (September, 1981), PP. 223--233.

11. Chang, T. C. "Computer-Aided Process Planning Today and in the Future." 1983 Fall Industrial Engineering Conference Proceedings, (October, 1983), pp. 349-354.

12. Chern, B. "A New System for Describing Simple Mechanical Parts." Proceeding of the Seminar on CAM-I Geometric Modeling and Automated Process Planning Projects. (February, 1979), pp. 11-29.

13. Childs, J. J. Numerical Control Part Programming. New York: Industrial Press Inc., 1973.

14. COMPACT II Programming Manual. Ann Arbor, Michigan: Manufacturing Data System, Inc., 1978.

15. Functional Specification for An Experimental Planning System XPS-1. Computer-Aided Manufacturing-International, Inc., Arlington, Texas, 1980.

16. "Computerized Process Planning." Manufacturing Horizons. (May/June, 1982), pp. 8-10.

17. Davis, R., D. B. Lenant. Knowledge-Based System in Artificial Intelligence. New York: McGraw-Hill Book Co., Inc., 1982.

18. Descotte, Y., J. Latmobe. "GARI: A Problem Solver that Plans How to Machine Mechanical Parts." Seventh International Joint Conference on Artificial Intelligence. Vancouver British Columbia, Canada, (August 1981), pp. 766-772.

19. Duda, R. O., J. G. Gaschnig. "Knowledge-based Expert Systems Come of Age." BYTE, 6, 9 (September, 1981), pp. 238-281.

20. Dunn, M. S. Computerized Production Process Planning for Machined Cylindrical Parts. East Hartford, Connecticut: UTRC, Final Report R81-945220-23, 1981.

21. Eary, Donald F., Gerald E. Johnson. Process Engineering for Manufacturing, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1962.

22. El-Midany, T. T., B. J. Davies. "AUTOCAP-A Dialogue System for Planning the Sequence of Operations for Turning Components. International Journal Machine Machine Tool Design and Research. Vol. 21 No. 3/4, 1981, pp. 175-191.

23. El-Midany, T. T., A. M. El-Tamimi. "Computer Aided Production Planning (CAPP) Systems and Approaches." IFAC Conference, Purdue university, 1982.

24. Eskicioglu, H., B. J. Davies. "An Interactive Process Planning System for Prisomatic Parts (ICAPP)." International Journal Machine Tool Design and Research. Vol. 21, No. 3/4, 1981, pp. 193-206.

25. Feigenbaum, E. A. "The Art of Artificial intelligence Themes and Case Studies of Knowledge Engineering." National Computer Conference Proceeding, (1978), pp. 227-240.

26. Feigenbaum, E. A., J. Feldman. Computer and Thought. New York: McGraw-Hill Book Co., Inc., 1963.

27. Fuchs, I. H. "Integrated CAD/CAM System-structures, Range of Application, Experiences." Proc. 4th Annual Conf. Computer Graphics on CAD/CAM Systems MIT, (1982), pp. 255-288.

28. Green, C., B. Raphael. "The Use of Theorem-Proving Techniques in Question-Answering Systems." ACM National Conference Proceedings, (1968), PP. 169-181.

29. Green, C. "Theorem Proving by Resolution as a basis for Question-Answering Systems." Machine Intelligence., 4 (1969), pp. 183-205.

30. Groover, M. P. Automation, Production Systems, and Computer-Aided Manufacturing. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1980.

31. Halevi, G. The Role of Computers in Manufacturing Processes. New York: John Wiley & Sons, Inc., 1980.

32. Hartquist, E. E., R. B. Tilove, H. B. Voelcker. "Representations in the PADL-1.0/n Processor: Boundary Representations and the BFILE/1 System." Production Automation Project, University of Rochester, Rochester, New York, 1980.

33. Hayes-Roth, Frederick, D. A. Waterman, D. B. Lenat. Building Expert Systems. Reading, Massachusetts: Addison-Wesley Publishing Company, Inc., 1983.

34. Hegland, D. E. "CAD/CAM Integration Key to the Automated Factory." Production Engineering, (August, 1981), pp. 31-35.

35. Horvath, M. "Semi-Generative Process Planning for Part Manufacturing." Advanced Manufacturing Technology. Ed. P. I. Blake. New York: North Holland, Inc., 1980, pp. 131-140.

36. Houtzeel, A. "Computer-Assisted Process Planning Minimizes Design and Manufacturing Costs." Industrial Engineering, 13, 11 (November, 1981), pp. 60-64.

37. Houtzeel, A. "The Many Faces of Group Technology." American Machinist, (January, 1979), pp. 115-120.

38. Hyde, M. O. Computers That Thinks. Hillside, New Jersey: Enslow Publishers, 1982.

39. Illustrated Glossary of Workpiece Form Features. (Doc. #R-80-ppp-02.1), CAM-I, Arlington, Texas, 1980.

40. "Implementing CIM." American Machinist, (August 1979), pp. 115-120.

41. Iwata, K., Y. Kakino, F. Ohba, N. Sugimura. "Development of Non-part Family Type Computer Aided Production Planning System CIMS/PRO." Advanced Manufacturing Technology. Ed. P. I. Blake. New York: North Holland, Inc., 1980, pp. 157-184.

42. Jackson, P. C. Jr. Introduction to Artificial Intelligence. New York: Petrocell Books, Inc., 1974.

43. Kakino, Y., F. Ohba., T. Moriwaki, K. Iwata. "A New Method of Parts Description for Computer-Aided Production Planning." Advances in Computer-Aided Manufacture. Editor D. McPherson. New York: North Holland, Inc., 1977, pp. 197-209.

44. Kleer, J. "Qualitative and Quantitative Reasoning in Classical Mechanics." Artificial Intelligence: An MIT Perspective Volume I. Eds. P. H. Winston, R. H. Brown. Cambridge, Massachusetts: The MIT Press, 1979, pp. 11-32.

45. Krouse, J. K. Computer-Aided Design and Computer-Aided Manufacturing. New York: Marcel Dekker Inc., 1982.

46. Lenat, D. B. "The Ubiquity of Discovery." National Computer Conference Proceedings, (1978), pp. 241-255.

47. Leslie, W. H. P. Numerical Control User's Handbook. New York: McGraw-Hill Book Co., Inc., 1970.

48. Lindsay, R. K., B. G. Buchanan, E. A. Feigenbaum, J. Lederberg. Applications of Artificial Intelligence for Organic Chemistry: The DENDRAL Project. New York: McGraw-Hill Book Co., Inc., 1980.

49. Link, C. H. "CAPP - CAM-I Automated Process Planning System." Proceedings of the 1976 CAM-I NC Conf., 1976.

50. Martin, W.A., R.J. Fateman. "The MACSYMA System." Proceedings of the Second Symposium on Symbolic and Algebraic Manipulation, (1971), pp. 59-75.

51. Matsushima, K., N. Okada, T. Sata. "Development of a Process planning System for Pressure Vessels." Annals of the CIRP, 28, 1 (1979), pp. 351-354.

52. Matsushima, K., N. Okada, T. Sata. "The Integration of CAD and CAM by Application of Artificial Intelligence Techniques." CAM-I Special Projects, 1983.

53. McDermott, J. "R1: An Expert in the Computer System Domain." AAAI, 1, (Jananuary, 1980), pp. 269-271.

54. Nau, Dana S. "Expert Computer System." Computer, (February, 1983), pp. 63-85.

55. Nilsson, N. J. Problem-Solving Method in Artificial Intelligence. New York: McGraw-Hill, Inc., 1971.

56. Nilsson, N. J. Principles of Artificial Intelligence. Palo Alto, California: Tioga Publishing Co., 1980.

57. Okino, N., Y. Kakazu, H. Kubo, N. Hashmoto. "Geometry Data-Base for Multi-Parts in CAD/CAM Systems, TIPS/GDB." Advanced Manufacturing Technology. Ed. P. I. Blake. New York: North Holland, Inc., 1980, pp. 71-86.

58. Okino, N., Y. Kakazu, H. Kubo. "TIPS-1: Technical Information Processing System for Computer-Aided Drawing and Manufacturing." Computer Languages for Numerical Control. Ed. J. Havany. New York: North Holland, Inc., 1973, pp. 141-150.

59. Ross, T., R. G. Munck. "Geometric Modeling in Computer-Aided Manufacturing." Proc. of the Seminar on CAM-I Geometric Modeling and Automated Process Planning Projects, (February, 1977), PP. 90-99.

60. Scheck, D. E. "Feasibility of Automated Process Planning." (Unpub. Ph.D. dissertation, Purdue University, 1966.)

61. Senior Staff of the Production Automation Project. The PADL-1.0/n Processor: Overview and System Documentation. University of Rochester, Rochester, New York, 1977.

62. Shank, R. C. "The Current State of AI: One Man's Opinion." The AI Magazine, (Winter/Spring, 1983), pp. 3-8.

63. Shank, R. C., C. K. Riesbeck. Inside Computer Understanding. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Publishers, 1981.

64. Shortliffe, E., R. Davis, B. Buchanan. "Production Rules as a Representation for a Knowledge-based Consultation Program." Artificial Intelligence, 8 (1977), pp. 15-45.

65. Spur, G., H. M. Anger, W. Kunzendorf, G. Stuckmann. "A Dialogue System for Computer Aided Manufacturing Process Planning. 19th International Machine Tool Design and Research Conference, 1978.

66. Stark, R. H. The PADL-1 Primer. Las Cruces, New Mexico: New Mexico State University, Technical Report NMSU-TR-80-CS-03, 1980.

67. Stefik, M. J. "Planning with Constraints (MOLGEN: Part I)." Artificial Intelligence, 16 (1981), pp. 111-140.

68. Stefik, M. J. "Planning with Constraints (MOLGEN: Part II)." Artificial Intelligence, 16 (1981), pp. 141-170.

69. Tempelhof, K. H. "A System of Computer-Aided Process Planning for Machine Parts." Advanced Manufacturing Technology. Ed. P. I. Blake. New York: North Holland, Inc., 1980, pp. 141-150.

70. Thornhill, R. B. Engineering Graphics and Numerical Control. New York: McGraw-Hill Book Co., Inc., 1967.

71. Tilove, R. B. "Representations in the PADL-1.0/n Processor: Simple Geometric Entities" Production Automation Project, University of Rochester, Rochester, New York, 1977.

72. Tipnis, V. A., S. A. Vogel, C. E. Lamb. "Computer-Aided Process Planning System for Aircraft." Advanced Manufacturing Technology. Ed. P. I. Blake. New York: North Holland, Inc., 1980, pp. 151-170.

73. Trucks, H. E. Designing for Economical Production. Dearborn, Michigan: Society of Manufacturing Engineers, 1976.

74. Tulkoff J. "Automated Process Planning-Using Group Technology to Improve Productivity." (Unpublished Paper Presented at Productivity in Aerospace Industries International Conference February 1982). Marietta, Georgia: Lockheed-Georgia Co., 1982.

75. Voelcker, H. B., W. Hunt. "The Role of Solid Modeling in Machining Process Modeling and NC Verification." Proc. 1981 International Congress of the SAE, February, 1981.

76. Weill, R., G. Spur, W. Eversheim. "Survey of Computer Process Planning Systems." CIRP Annals, 1982.

77. Weiss, S. M., K. B. Kern, C. A. Kulikowski, M. Uschold. "A Guide to the Use of the EXPERT Consultation System." New Brunswick, New Jersey: Technical Report CBM-TR-94 1979.

78. Wesley M. A., T. Lozano-Perez, L. I. Lieberman, M. A. Lavin, D. D. Grossman. "A Geometric Modeling System for Automated Mechanical Assembly." IBM J. Research Develop, 24, 1 (January, 1980), pp. 64-74.

79. Winston, P. H. Artificial Intelligence. Reading, Massachusetts: Addison-Wesley Publishing Co., Inc., 1977.

80. Winston, P. H., B. K. P. Horn. LISP. Reading, Massachusetts: Addison-Wesley Publishing Co., Inc., 1981.

81. Wysk, R. A. "Automated Process Planning and Selection." (Ph.D. dissertation, Purdue University, 1978.)

82. Yankee, Herbert W. Manufacturing Processes. Englewood Cliffs, New Jersey: Prentice-Hall, Inc., 1979.

83. Zimmers, E. W., L. J. Plebani. "Using a Turnkey Interactive Graphics System in Computer-Aided Manufacturing." Industrial Engineering, 13, 11 (November, 1981), pp. 98-104.

APPENDIX A

HALFSPACES, SURFACE NORMAL, SURFACE TYPE CODES

Introduction

This appendix is an introduction to the basic concept of the surface normal and the surface code defined in the PADL-1 system. Surface codes implies the surface normal directions. Surface normal is related to the concept of halfspaces. The next section presents a brief introduction to the halfspaces.

Halfspaces

Halfspaces are the results of dividing a three Euclidean space, E3, by an unbounded surface. The two halfspaces are distinguished as the positive halfspace (H+) and the negative halfspace (H-). The dividing surface is called the halfspace boundary (H). In the PADL-1 system, H is either an unbounded plane surface or unbounded cylindrical surface. "Positive" halfspace and "negative" halfspace may be designated by any agreed convention in a designed system. In the PADL-1 system, the exterior of a solid has been chosen as the "positive" side of any H in which a surface lies. H+ consists of all points on the positive side and H- consists of all points on the negative side. Points in H belong to both H+ and H-.

Surface Normal and Surface Type Codes

At any point on a surface of a solid, two opposite normal vectors can be constructed. They are perpendicular to the tangent line evaluated at the point, one directs away from H+, the other one directs away from H-. The one that directs away from H- (points toward exterior of the solid) is called the "surface normal" in the PADL-1 system. A surface normal can be used to distinguish the interior and the exterior of a solid.

"Surface normal direction" is the direction which a surface normal points to in a three dimensional coordinate system. There are six surface normal directions for plane surfaces in the PADL-1 system. Figure 31 shows the surface normal directions of plane surfaces on a block. The signs '+' and '-' indicate the direction relating to one of the PADL coordinate axes. An integer number is assigned to each face according to the surface normal direction of each face. The coding strategy for plane surfaces is to add a number 1, 2, or 3 to 100 or to substract a number 1, 2, or 3 from 100. A number which is less than 100 implies that the surface normal points to the negative side of one of the coordinate axes. The numbers shown in Figure 31 are designed for plane surfaces. These numbers are called the surface type codes.

The surface normal directions of disks are defined as the ones for plane surfaces, since disks are plane surfaces. The coding strategy for disk surfaces is to add or to substract a number 1, 2, or 3 to or from 200. A number less than 200 implies that the surface normal points to the negative side of one of the coordinate axes.

The positive surface normal direction of a cylindrical surface is defined as the surface normal that points away from the axis of the cylinder. The negative surface normal

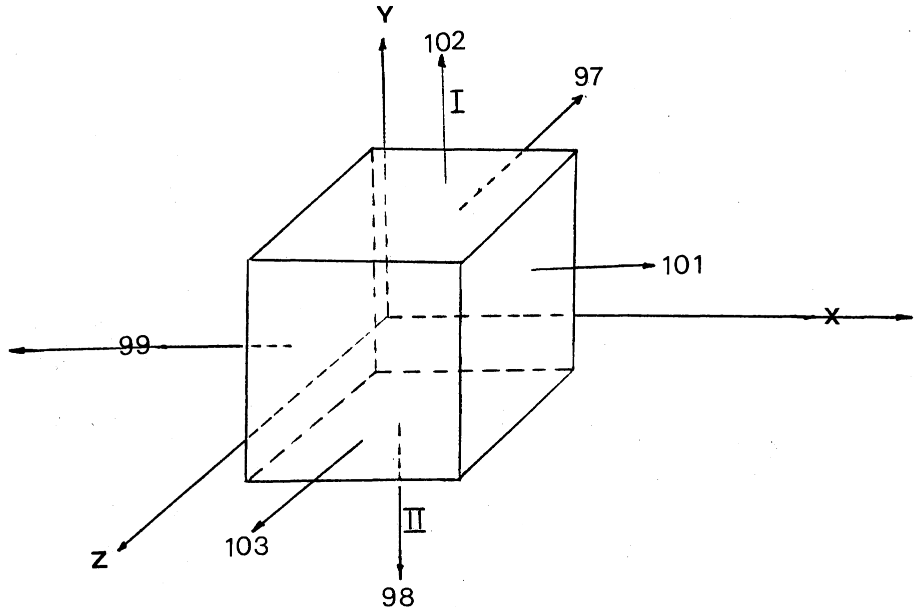


Figure 31. Surface Normals on Plane Surfaces

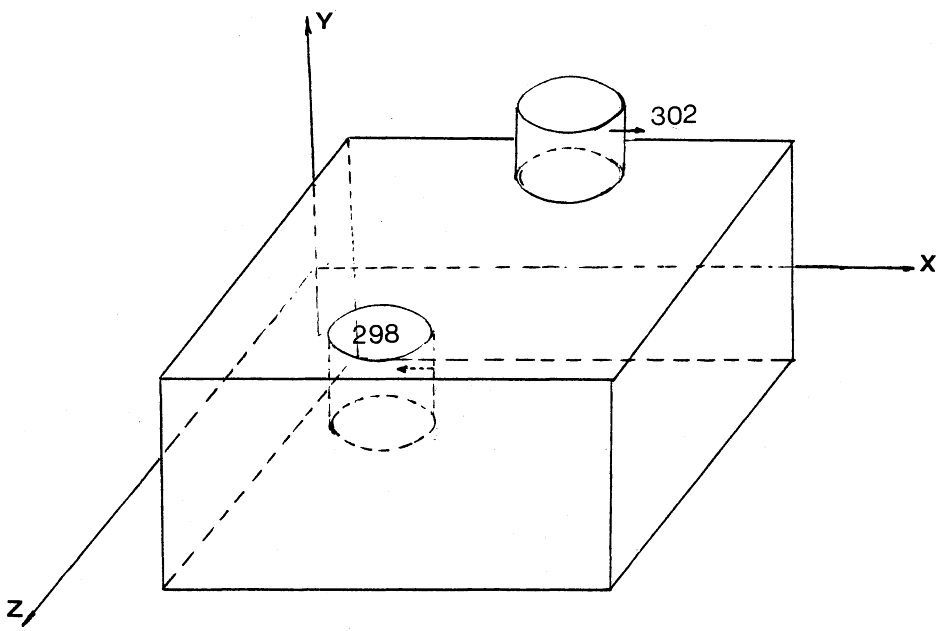


Figure 32. Surface Normals on Y-type Cylinders

direction of a cylindrical surface is defined as the surface normal that points toward the axis of the cylinder. Figure 32 shows an example of the Y-type cylinder surface and the surface normal directions. The coding strategy for plane surfaces is to add or to substract a number 1, 2, or 3 to or from 300. A number less than 300 implies that the surface normal points to one of the coordinate axes. Table II (page 68) lists all the surface type numbers for all of the possible faces.

APPENDIX B

FILE DESCRIPTIONS

Introduction

This appendix describes the data structures and the contents of each file used in FREXPP. These files are DIRECTORY file (DIRECT), FACE file (FACE), EDGE file (EDGE), FEATURE file (FEATURE), INTERNAL FEATURE file (INTERF), INTERNAL-EXTERNAL RELATION file (INOFRL), form feature recognition FINAL RESULT file (FINALF), single feature INFORMATION file (FEINPT), SINGLE FEATURE PROCESS file (SINGLF) and the PROCESS PLAN file (PROCEF).

Directory File

The directory file (DIRECT) contains three records. The length of each record is 200 bytes. The first record is the main directory record, and it has 8 fields. The first 7 fields are 4 bytes long (I4 format) and the last field is designed for future expansion. The contents of these fields are described as follows:

1. An integer number indicates the number of FACE records in the FACE file.

2. An integer number indicates the number of EDGE records in the EDGE file.

3. An integer number indicates the available feature number.

4. An integer number indicates the available field number of the third record of this file.

5. An integer number indicates the current available record number for the INTERNAL FEATURE file.

6. An integer number indicates the current available record number for the feature recognition result file (FINALF).

Record 2 and Record 3 contain 100 fields. The length of each field is 2 bytes. These two records are used to link the same type of features together. The field numbers of the second record corresponds to the feature type numbers. The content of each field is the field number of the third record.

Fifty pair fields are designed in the third record. The first field is an information field which contains the feature numbers. The second one is a link field which contains a field number. The same type of features are linked through the link field. An integer number 0 in a field of any record indicates that no information is related to it.

FACE File

The FACE file is a direct access file which contains the surface information of a part. The record length of this file is 100 bytes. Each record contains the following information:

1. Face name (I3). Each surface has a name (an integer number). In the FACE file, the record number implies a surface name. A negative number indicates that this surface is represented in other FACE records.

2. Face type code (I5). A face type code is represented by a 3 digit number which distinguishes a plane

surface from a cylindrical surface. It also distinguishs an X-type surface from a Y-type surface or a Z-type surface. Besides, a face type code implies the surface normal direction (positive or negative side along an axis) of a surface on a part. The strategy of assigning a face type code in the PADL-1 system was discussed in Appendix A.

3. Number of edges (I3). An integer number denotes the number of edges which encloses a surface.

4. Alternate name (I3). An integer number indicates the original p-face name of the surface.

5. F-face indicating flag (I3). 0 stands for a non-f-face.

6. A two byte dummy variable (A2).

7. Configurational positional information (3(I1,F9.5)). A 3 tuple of real numbers that carries the configurational and positional parameters of the surface. If the surface is a plane or disk, the first number of this field represents the surface axis offset. It is the distance between the original and the surface. The other two numbers are zero. If the face is cylindrical, the first two numbers indicate the position of the center axis, the third number indicates the length of the radius. Each of the three numbers is led by an integer number in I1 format.

8. B-face link field (I3). This field contains a b-face name which is a part of the surface represented by this b-face.

9. Overflow record number (I3). An integer number indicates the record number in the FACE RECORD OVERFLOW

file. A FACE record was designed to store 12 edge names. When the number of edges is larger than 12, the OVERFLOW records in the FACE RECORD OVERFLOW file is used. The total number of edges of a surface is limited to 60 in this research.

10. Edge fields (12I3). A maximum of 12 edge names can be stored in these fields.

11. B-face name (I3). An integer number indicates a b-face name with which the surface is associated.

The FACE file provides the surface type and configuration information for the feature recognition procedure.

EDGE File

An edge is the intersection of two surfaces. Edges, in the PADL-1 system, are classified into three types: LINES, ARCS and CEDGES. LINES are the collections of regular straight lines. A straight line is defined by two points. ARCS are the collection of circular arcs. An arc is defined by its radius, center point, and two end points. The two end points are expressed by a minimum angle and a maximum angle in terms of degrees. CEDGES are the collections of the intersections of two cylindrical surfaces and defined by two sets of arcs. The EDGE file is a direct access file. The number of records in this file is equal to the number of edges for a part. Each record is 120 bytes in length and contains the following information:

1. Edge name (I3). Each edge has been assigned a name

(an integer number). The naming strategy used in this file is different from the one in the PADL-1 system. In the PADL-1 system, an edge name is defined as "the j th edge of face i". By this way, an edge is named twice under two different names. To avoid redundant information, an integer number is sequentially assigned to an edge as the sequence of an edge appeared in a face record. The edge name corresponds to the record number in the EDGE file.

2. Edge type code (I6). An edge type code is used to distinguish the X axis based edges from the Y or Z axis based edges. LINES type edge codes are 1001, 1002, 1003 for X, Y, and Z type lines respectively. ARCS type edge codes are 2001, 2002, and 2003 for X, Y, and Z type arcs respectively.

3. Edge updating flag (I3). 0 denotes that the edge information has been modified.

4. Dummy variables (3I2). This variable contains 6 bytes which can be used for the future expansion.

5. Surface names (2I3). A pair of integers which specify the faces that share the edge.

6. Overflow record number (I6). An integer number that indicates the record number in the EDGE RECORD OVERFLOW file.

7. Configurational and positional information (6(I2.F9.5)). A 6-tuple of real number that carries the configurational and positional parameters of the edge. Each tuple is led by an integer number. If the edge is a line, the coordinates of the two end points are expressed by two

3-tuple of real numbers. Each of the 3-tuples stores the coordinates of a point in the order of X, Y and Z directions. If the edge is an arc, the first three data points represent the center point of an arc; the second three data points represent the radius of the arc; and the last two data points represent the minimum angle of the arc. If the edge is a CEDGE, the data of one arc is stored in the EDGE record and the data of another arc is stored in the EDGE RECORD OVERFLOW file. The EDGE RECORD OVERFLOW file is designed and implimented in this research. However, the feature of any two crossed cylinders is not discussed in this research. The EDGE file provides the coordinates of edges and face names related to an edge for the feature recognition procedure.

FEATURE File

The dataset name of this file is FEATURE. This file stores all the recognized features. The record size of this file is 60 bytes. Each has 30 fields. Each field is 2 bytes long with the I2 format. The record numbers correspond to the feature numbers.

1. The first field stores the feature number.
2. The second field stores the feature type number.
3. The third field stores an integer number that indicates the number of fields are used, starting from field number 4.
4. The fourth field stores the key face name of the feature.

5. The remaining fields store the side surface name. The FEATURE file is updated whenever a new feature is recognized.

INTERNAL FEATURE File

INTERF is the dataset name of the INTERNAL FEATURE file. This file stores the features that contain internal features. The record length of this file is 60 bytes. Each record is equally divided into 30 fields and each field is 2 bytes long with the I2 format.

1. The first field stores a feature number.
2. The second field stores the number of internal features.
3. The third field stores the key face name of the feature.
4. Fields 4 through 30 stores the internal feature numbers.

The records of this file will be referenced by the INTERNAL-EXTERNAL RELATION file.

INTERNAL-EXTERNAL RELATION File

The INOFRL file specifies the internal and external feature relations. The record numbers imply the feature numbers. Each record of this file has three fields which are:

1. The first field is a feature indicator. This field has an "I6" format. The content of this field could be 0 or 1. A number "0" represents an external feature and a number

"1" indicates an internal feature.

2. The second field has an "I7" format. If the feature is an external feature having an internal feature, this field contains an INTERNAL FEATURE record number. If the feature is an external one having no internal features, a "0" number is assigned to this field. If the feature is an internal feature, the associated external feature number is assigned to this field.

3. The third field has an "I7" format. It contains an external feature number if the feature is an internal one of two external features or the feature is an internal feature containing internal features.

FINAL RESULT File

FINALF is the dataset name of this file. This file stores the sequenced features. Each record is 20 bytes long. The content of each record is explained as follows:

1. The first field stores a feature number.
2. The second field stores the key surface number of the feature.
3. The third field stores the key surface type code.

The format of a record is (I6,I7,I7). This file is an input file to the process planning procedure.

INFORMATION File

FEINPT is the dataset name for the INFORMATION file. Each record of this file is 160 bytes long and has the format (160A1). The first record contains the surface finish data. The second record stores the length of a hole diameter. The third record stores the depth of a hole. The fourth record stores the maximum length of the diameter. The fifth record stores the minimum length of the diameter. The sixth record stores the feature type number. This file prepares the observations for the expert process planning system.

SINGLE FEATURE PROCESS File

This file stores the first suggested manufacturing process from the expert system. The first record has the (5I4)format. It stores the feature sequence number, the feature number, the feature type code, the surface number, and an integer number 1 or 0, where 1 stands for turning the part and 0 for not turning the part. The second record stores the confidence level and the first selected manufacturing process. The format of the second record is (72A1). The dataset name of this file is SINGLF.

PROCESS PLAN File

PROCEF is the dataset name of the file. It stores the final result of the process planning. The content of each record is described as follows:

1. The sequence number,
2. The feature number,
3. The feature type number,
4. The surface number,
5. An integer number, 1 for turn over the part and

0 for not turn over the part,

6. The confidence level, and

7. The selected manufacturing process.

The record layout is (5I4,7A1,45A1).

APPENDIX C

SYSTEM REQUIREMENTS AND COMMANDS OF FREXPP

In this appendix the storage requirements and the system commands of the FREXPP system are discussed. FREXPP was designed to execute on the VAX 11/780 under VMOS (Virtuual Memory Operating System). A Tektronix 4010 storage tube terminal or one compatible to this type of terminal is required to display the graphics.

FREXPP contains three major sections, part design, feature recognition and process plan generation. PADL-1 was adopted to implement the part design procedure. The storage requirement for the PADL-1 executable codes is 667 blocks. One block contains 520 bytes on a VAX system. The feature recognition procedure is implemented by a set of FORTRAN programs. The storage requirement for the executable codes of this procedure is 127 blocks. The process plan sub-procedure is based on the EXPERT consultation system. The storage requirement for the EXPERT executable codes is 304 blocks plus 286 blocks for the knowledge compilation program. The storage for the data area requires 250 blocks.

The command for executing the FREXPP system is @FREXPP or FREXPP.COM. FREXPP.COM is a command file containing a sequence of commands that are used to run the PADL-1 system, the feature recognition system, and the expert process planning system. "@" is the Execute Procedure character in the VAX 11/780 system. Table XX shows a list of commands required to run this system. @FREXPP means that the commands in the FREXPP.COM file will be executed by the VAX operating system.

TABLE XX

THE FREXPP SYSTEM EXECUTION COMMANDS

FREXPP.COM

```
00100$ @PADL
00200$ @RECOG
00300$ @PROCESS
```

PADL.COM

```
00100$ON ERROR THEN GOTO EXIT
00200$ON CONTROLLY THEN GOTO EXIT
00300$ SET WORK/LIM=400
00400$ SET TERM/UPPER
00500$ ASSIGN/USER_MODE SYS$OUTPUT: FOR000
00600$ ASSIGN/USER_MODE MESAGE.DAT  FOR002
00700$ ASSIGN/USER_MODE SYS$OUTPUT: FOR003
00800$ ASSIGN/USER_MODE SYS$OUTPUT: FOR004
00900$ ASSIGN/USER_MODE SYS$OUTPUT: FOR006
01000$ ASSIGN/USER_MODE SYS$OUTPUT: FOR007
01100$ ASSIGN/USER_MODE SYS$OUTPUT: FOR099
01200$ ASSIGN/USER_MODE 'F$LOGICAL("TT") SYS$INPUT
01300$ RUN/NODEBUG PADL
01400$EXIT:
01500$ SET TERM/LOWER
01600$ SET WORK/LIM=150
```

RECOG.COM

```
00100$ ASSIGN/USER_MODE 'F$LOGICAL("TT") SYS$INPUT
00200$ RUN RECOG
```

PROCESS.COM

```
00100$ ASSIGN/USER_MODE 'F$LOGICAL("TT") SYS$INPUT
00200$ RUN EXPERT
```

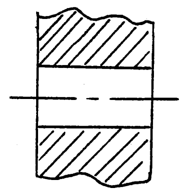
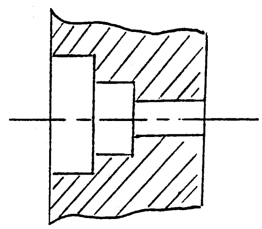
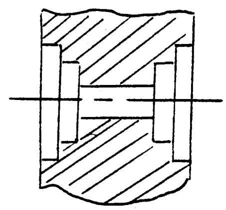
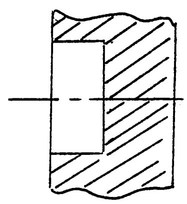
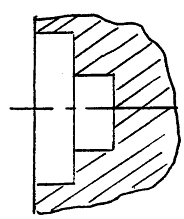
These three sub-procedures are executed sequentially and loaded into the core memory when the preceeding procedure is complete. The first command @PADL triggers the PADL-1 solid modeling system. The time to display the part in Figure 3, page 12, is approximately 30 seconds (real time) using a 1200 baud rate terminal and is approximately 53 seconds for displaying the part in Figure 6, page 15. The running time depends on the complexity of a part.

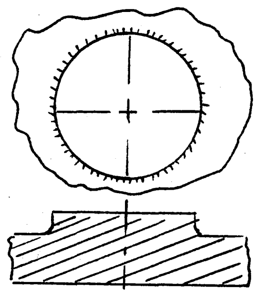
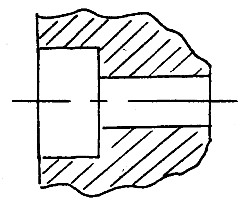
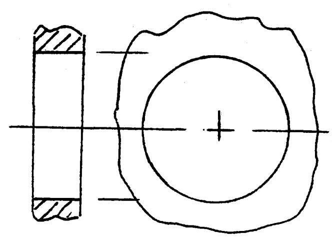
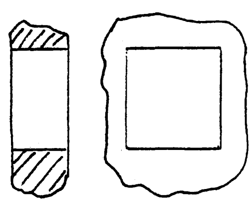
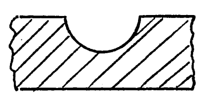
The Second command @RECOG triggers the feature recognition procedure. The time for creating the FACE file and EDGE file is approximately 7 seconds of real time for both example parts. It takes less than a second to recognize a feature once the f-faces have been decided. The third command executes the expert process planning system. The time for making a suggested manufacturing process is less than 2 seconds.

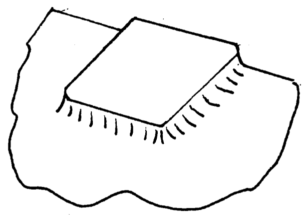
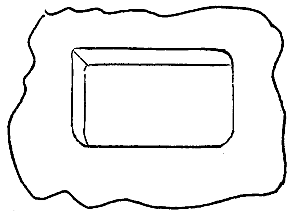
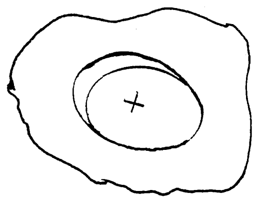
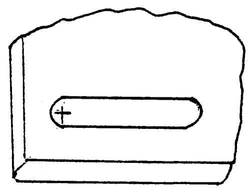
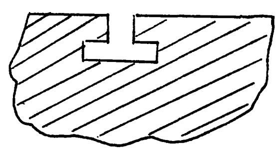
FREXPP is an interactive system. The response time is fast. The longest waiting interval is the time needed to copy data from core memory to disk. FREXPP is not suited for the micro-computer because both the PADL-1 system and EXPERT system require large data storage.

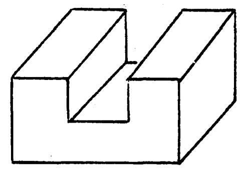
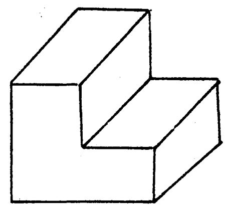
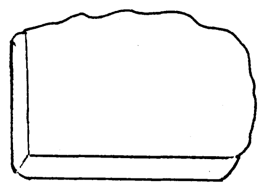
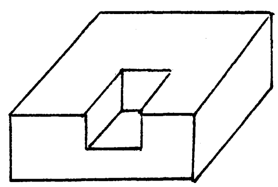
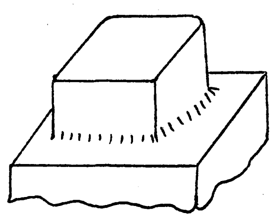
APPENDIX D

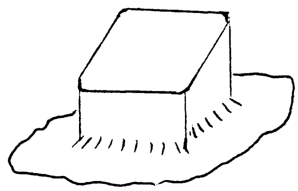
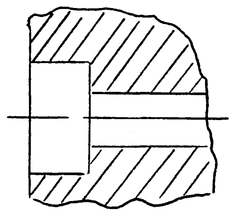
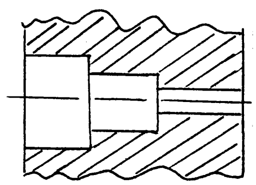
FORM FEATURES ILLUSTRATION

FEATURE NUMBER	DEFINITION	ILLUSTRATION
1	BORE-1. Precision, single diameter thru-going hole, normal to surface.	
2	BORE-2. Precision, multi-diameter, concentric, thru-going hole, stepped to one end, normal to surface.	
3	BORE-3. Precision, multi-diameter, concentric, thru-going hole, steped to both ends, normal to surface.	
4	BORE-4. Precision, single diameter, blind hole, normal to surface.	
5	BORE-5. Precision, multi-diameter, concentric, blind hole, stepped to one end, normal to surface.	

6	BOSS. A short, cylindrical projection from the surface or a fabricated workpiece.	
7	CBORE. Counter Bore. Cylindrical enlargement on the end of a hole.	
8	HOLE-1. A through, round, single-diameter opening, perpendicular to the surface of the workpiece.	
9	HOLE-2. A through, opening of square shape, normal to the surface of the workpiece.	
10	NOTCH. A U-shaped recess in the surface of a workpiece.	

11	PAD. A slight projection, rectangular in shape, on the work piece surface.	
12	POCKET-1. A shallow square or rectangular shaped cavity in the surface of a workpiece.	
13	POCKET-2. A shallow circular shapped cavity in the surface of a workpiece.	
14	SLOT-1. A long narrow thru-going opening in workpiece.	
15	TSLOT. A long narrow Channel with a T-shaped cross-section.	

16	SLOT. A rectangular recess cross the surface of a workpiece. A SLOT is also called a GROOVE.	
17	STEP. The formation of two perpendicular plane surface with a 90 degree angle.	
18	PLANE. A flat surface.	
21	BLIND SLOT. A rectangular recess along the edge of of a workpiece without crossing the surface. A blind slot is also called SLOT-2.	
23	BLOCK. A projection, rectangular in shape, on the workpiece surface.	

25	POST. A high projection rectangular in shape, on the workpiece surface.	
50	SINGLE-STEP BORE. TWO-diameter, concentric, thru-going hole stepped to one end, normal to surface.	
55	DOUBLE-STEP BORE. Multi-diameter, concentric, thru-going hole stepped to one end, normal to surface.	

APPENDIX E

LISTING OF PROCESS DECISION MODEL

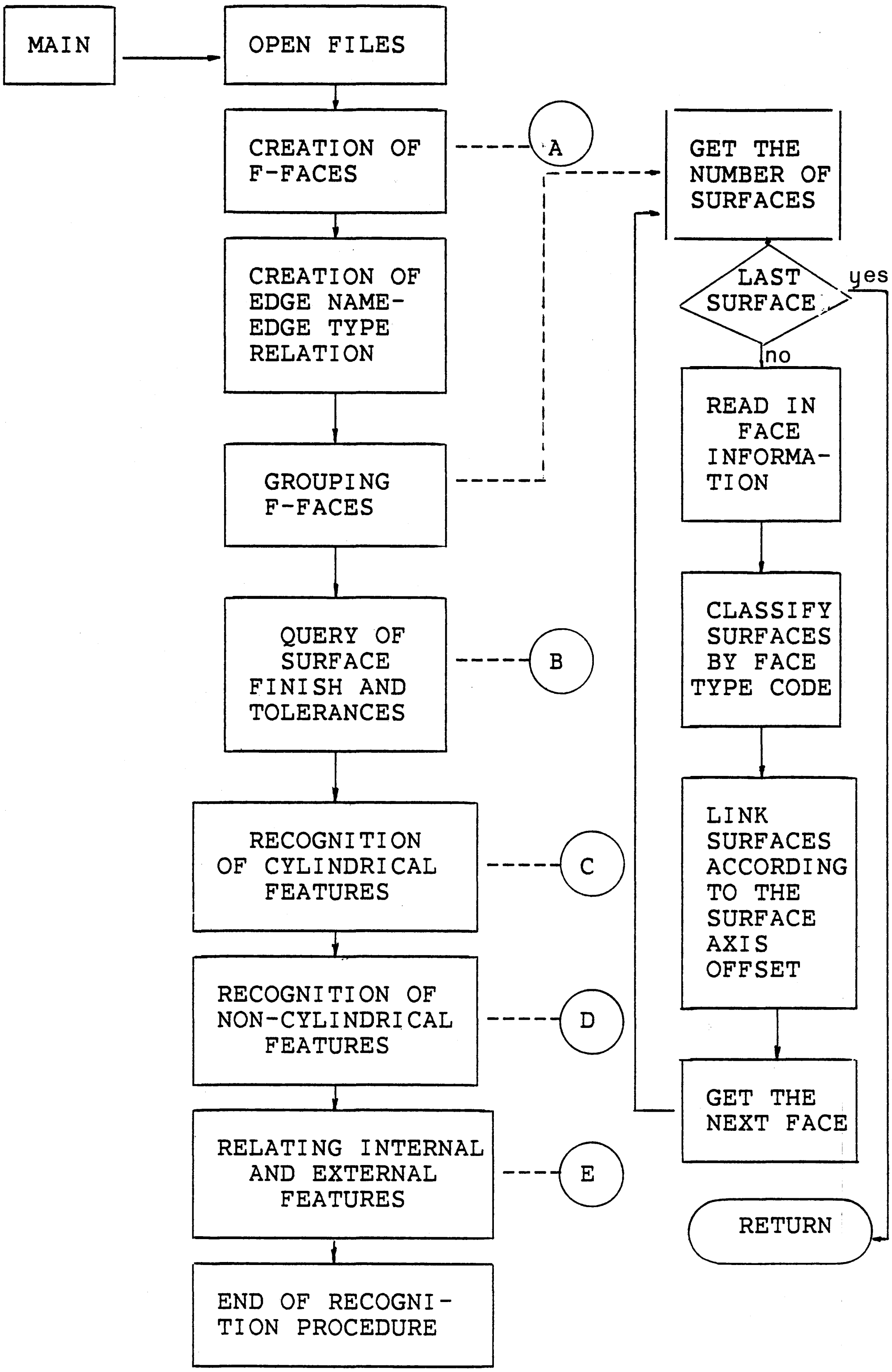
```
0100**HYPOTHESES
0200*TAXONOMY
0300BORR BORING TYPE PROBLEM
0400DRIL DRILLING TYPE PROBLEM
0500MILL MILLING TYPE PROBLEM
0600REAM REAMING TYPE PROBLEM
0700PRHO PREVIOUS HOLE DOES NOT EXIST
0800MMLL NON-CYLINDRICAL FEATURE PROCESSING PROBLEM
0900
1000*PROCESS
1100DRLL DRILL
1200BOOR BORE
1300REIM REAM
1400CDRL CENTER DRILL, DRILL
1500CBOR CENTER DRILL, DRILL, AND BORE
1600CREM CENTER DRILL, DRILL, AND REAM
1700CMIL CENTER DRILL, DRILL, AND MILL
1800MMIL Rough Mill, Finish Mill
1900
2000**FINDINGS
2100*Numerical
2200ROUF Roughness of the surface
2300DIAM Diameter of the hole
2400DETH Depth of the hole
2500XTOL Maximum length of the hole diameter(in 0.001")
2600STOL Minimum length of the hole diameter(in 0.001")
2700
2800*Multiple choice
2900 Type of Feature
3000 HOLE A Through Hole
3100 BORE A Non-through Hole
3200 NONC A Non-cylindrical feature
3300
3400*FUNCTIONAL FINDINGS
3500DTOL=XTOL-STOL
3600DIDE=DIAM/DETH
3700
3800**RULES
3900*FH RULES
4000F(NONC,T)->H(MMLL,1)
4100F(DIAM,0:0.1875)->H(PRHO,.95)
4200F(ROUF,125:250) &F(DIAM,1.2:2.5) &F(DTOL,8.0:*)+
4300->H(DRIL,.85)
4400F(ROUF,63:250) &F(DIAM,0.008:1.2) &F(DTOL,8.0:*)+
4500->H(DRIL,.8)
4600F(ROUF,32:250) &F(DIAM,2.:*) &F(DTOL,4.:*) &F(DIDE,4:*)+
4700&F(BORE,T)-> H(MILL,.95)
4800F(ROUF,32:250) &F(DIAM,2.:*) &F(DTOL,8.:*) &F(DIDE,4:*)+
4900&F(HOLE,T)-> H(MILL,.95)
5000F(ROUF,16:32) &F(DIAM,1.0:*) &F(DTOL,1.6:*) &F(BORE,T)+
5100-> H(BORR,0.8)
5200F(DIAM,1.0:*) &F(DTOL,1.6:4) &F(HOLE,T)-> H(BORR,0.8)
5300F(ROUF,32:125) &F(DIAM,1.0:*) &F(DTOL,1.6:*) &F(BORE,T)+
5400-> H(BORR,0.7)
5500F(ROUF,16:32) &F(DIAM,0.08:1.0) &F(HOLE,T)-> H(REAM,0.8)
```

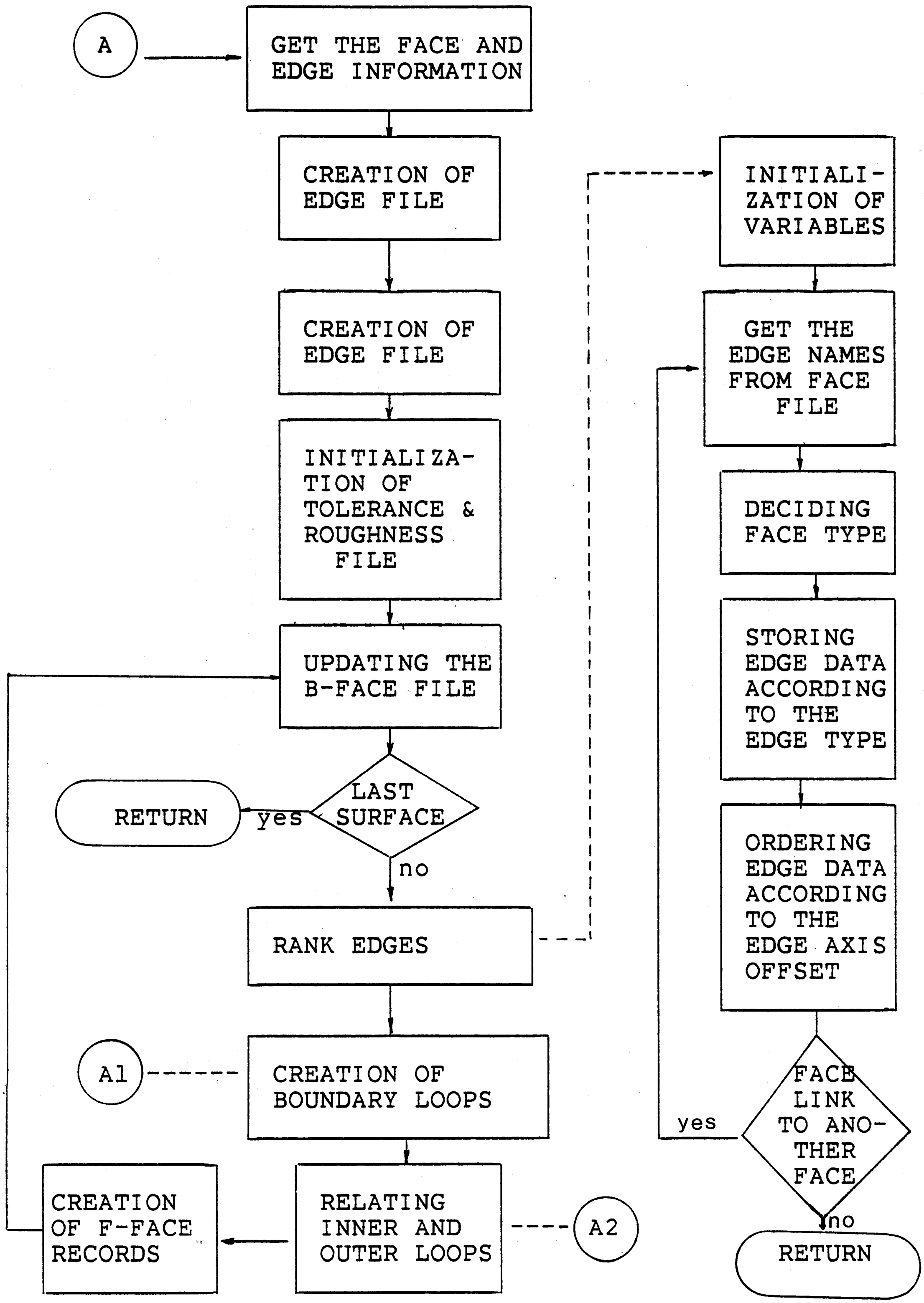
```
5600F(ROUF,16:125) &F(DIAM,0.12:1) &F(DTOL,0.:8) &F(HOLE,T)+
5700-> H(REAM,0.8)
5800F(ROUF,32:125) &F(DIAM,1:2)-> H(REAM,0.6)
5900
6000*HH RULES
6100*IF
6200[1:F(HOLE,T),F(BORE,T)] &H(PRHO,0:0.01)
6300*THEN
6400H(DRIL,.01:*)->H(DRLL,.70)
6500H(MILL,.01:*)->H(MLLL,.75)
6600H(BORR,.01:*)->H(BOOR,.70)
6700H(REAM,.01:*)->H(REIM,.70)
6800*END
6900
7000*IF
7100[1:F(HOLE,T),F(BORE,T)] &H(PRHO,0.5:*)
7200*THEN
7300H(DRIL,.01:*)->H(CDRL,.70)
7400H(MILL,.01:*)->H(CMIL,.75)
7500H(BORR,.01:*)->H(CBOR,.70)
7600H(REAM,.01:*)->H(CREM,.70)
7700*END
7800
7900*IF
8000F(NONC,T)
8100*THEN
8200H(MMLL,0.01:*)-> H(MMIL,.90)
8300*END
```

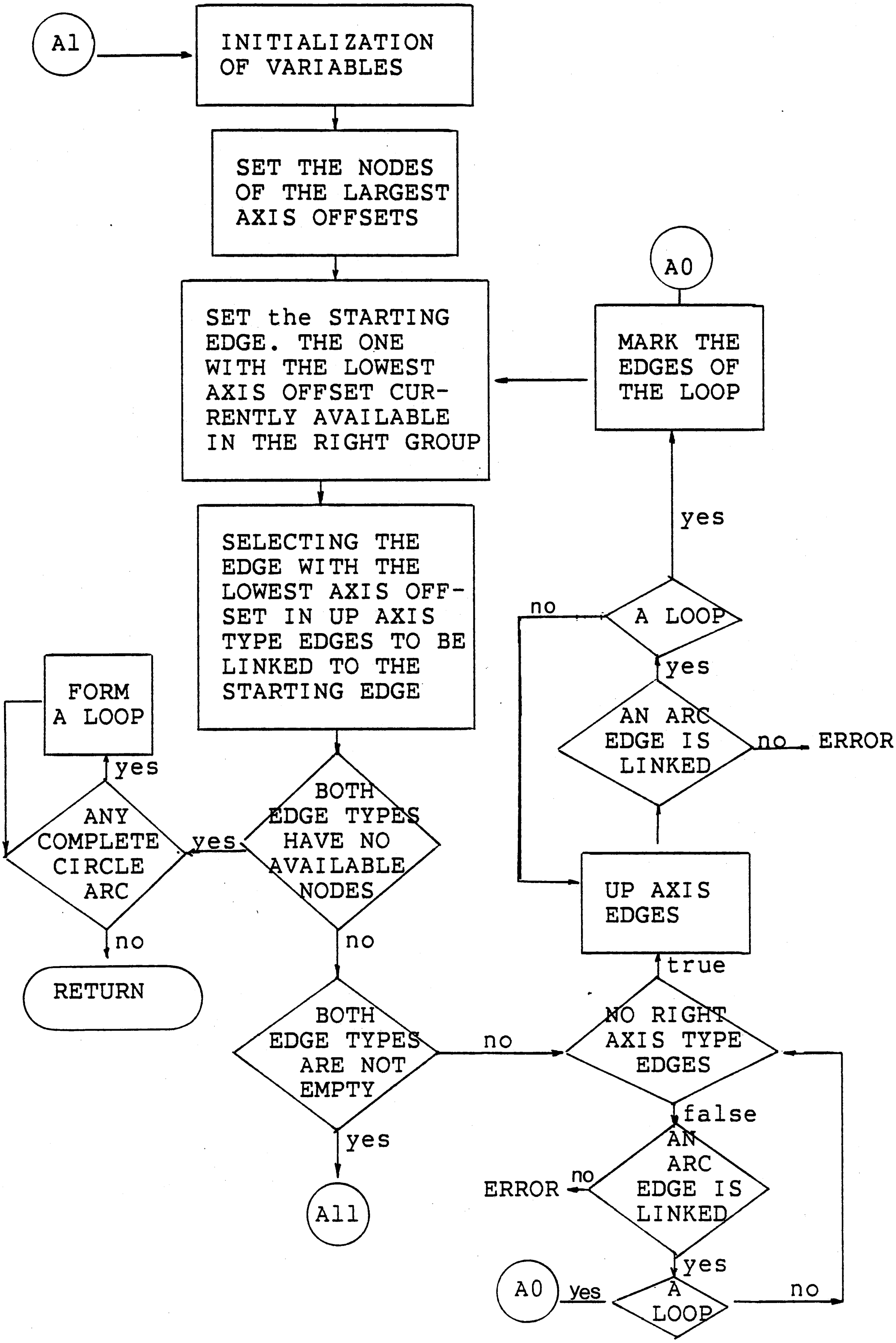
Notes:
- ** indicates one of the major sections in the EXPERT system.
- * indicates a sub-section.
- + stands for a continuation.

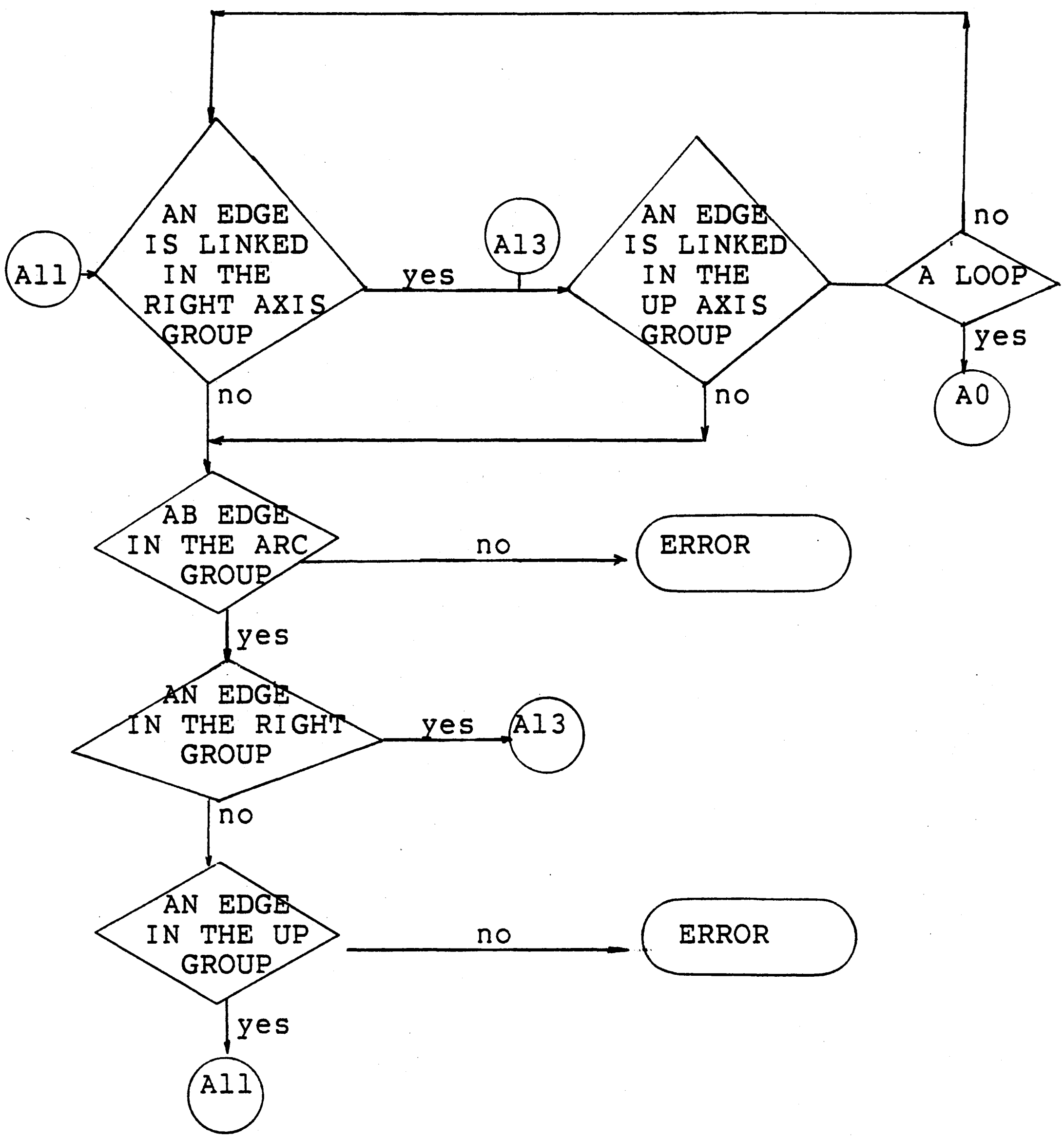
APPENDIX F

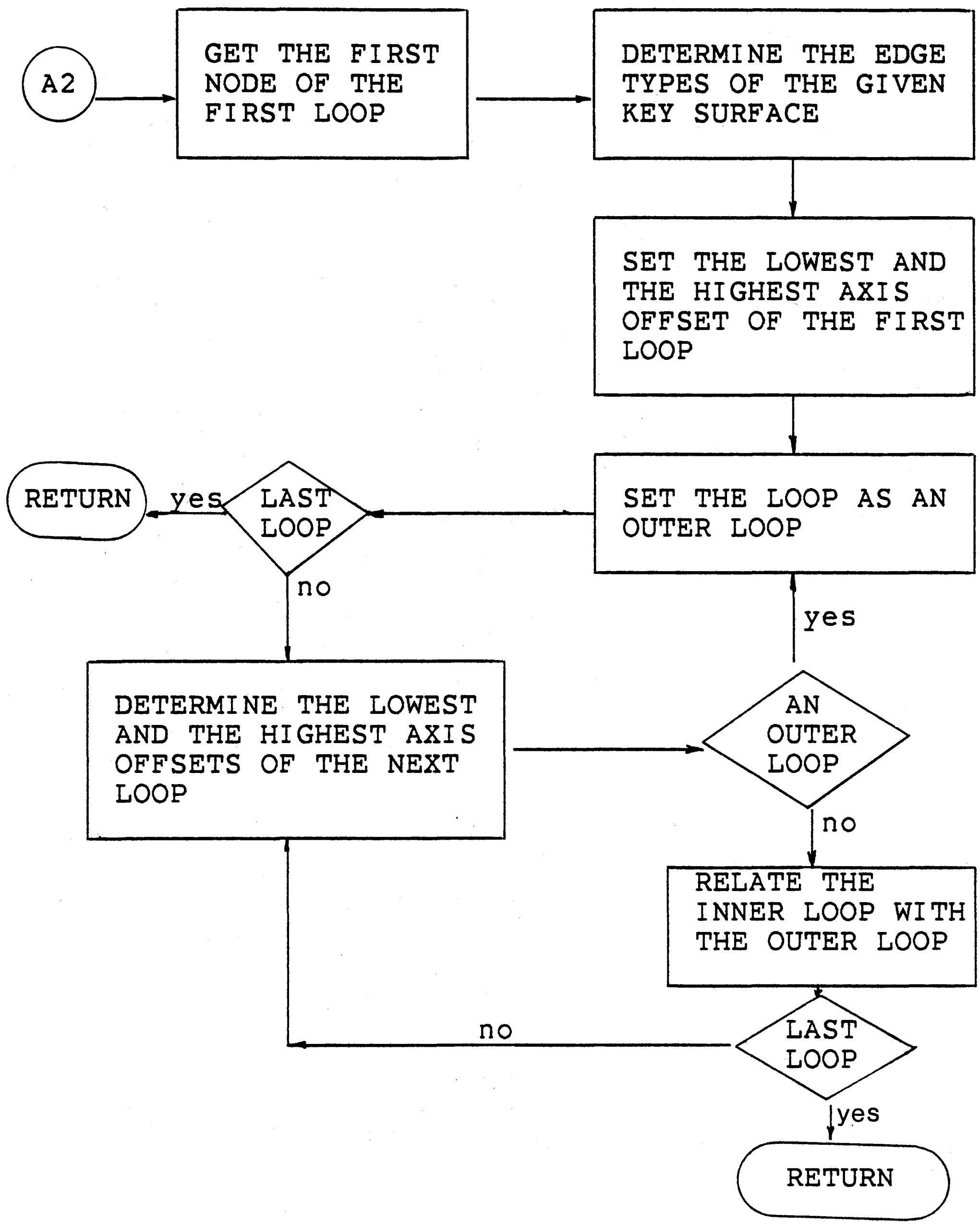
THE FLOW CHART OF THE FEATURE RECOGNITION PROGRAM

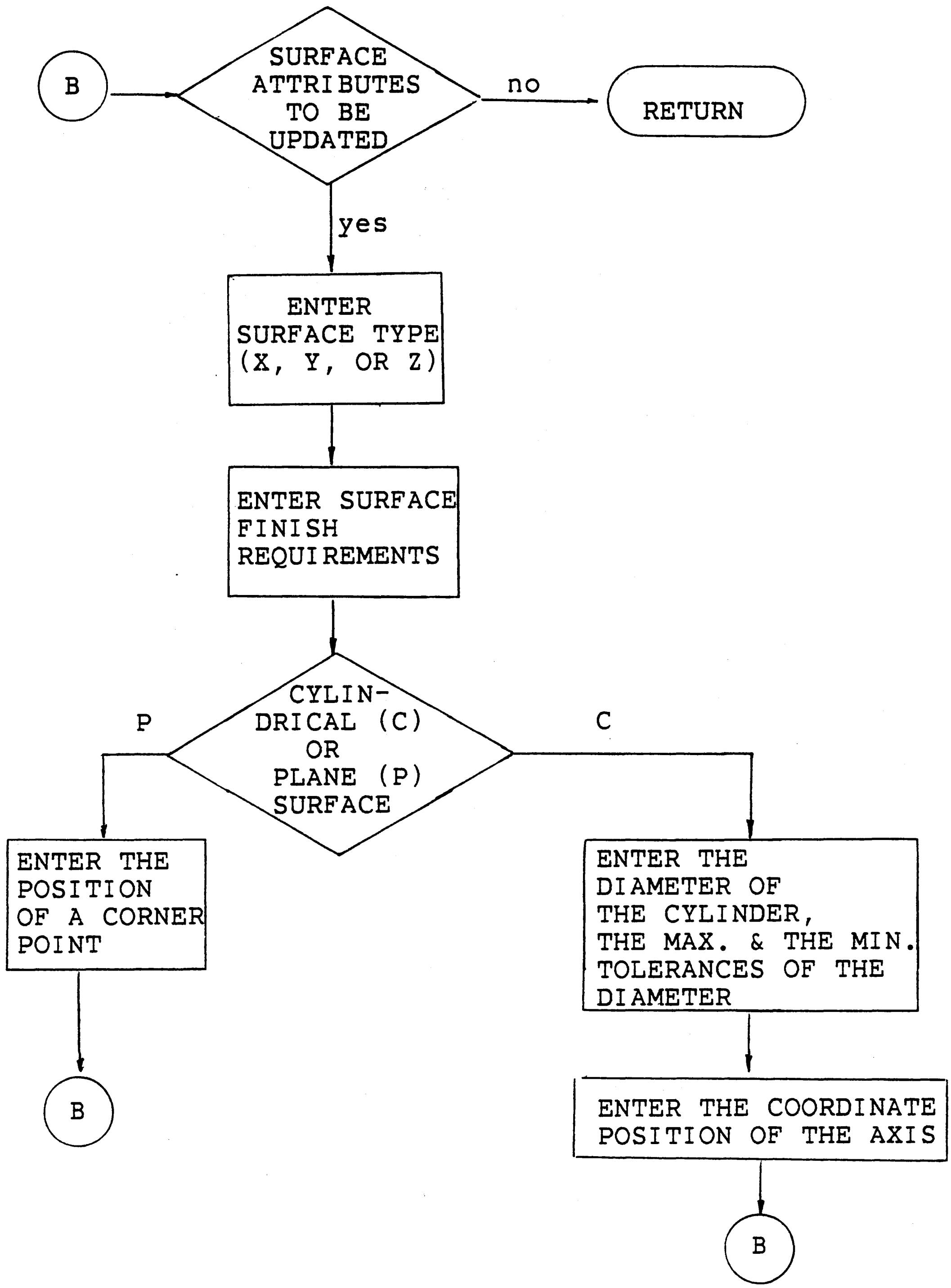

MAIN
OPEN FILES
CREATION OF F-FACES
A
CREATION OF EDGE NAME-EDGE TYPE RELATION
GROUPING F-FACES
QUERY OF SURFACE FINISH AND TOLERANCES
B
RECOGNITION OF CYLINDRICAL FEATURES
C
RECOGNITION OF NON-CYLINDRICAL FEATURES
D
RELATING INTERNAL AND EXTERNAL FEATURES
E
END OF RECOGNITION PROCEDURE
GET THE NUMBER OF SURFACES
LAST SURFACE
yes
no
READ IN FACE INFORMATION
CLASSIFY SURFACES BY FACE TYPE CODE
LINK SURFACES ACCORDING TO THE SURFACE AXIS OFFSET
GET THE NEXT FACE
RETURN

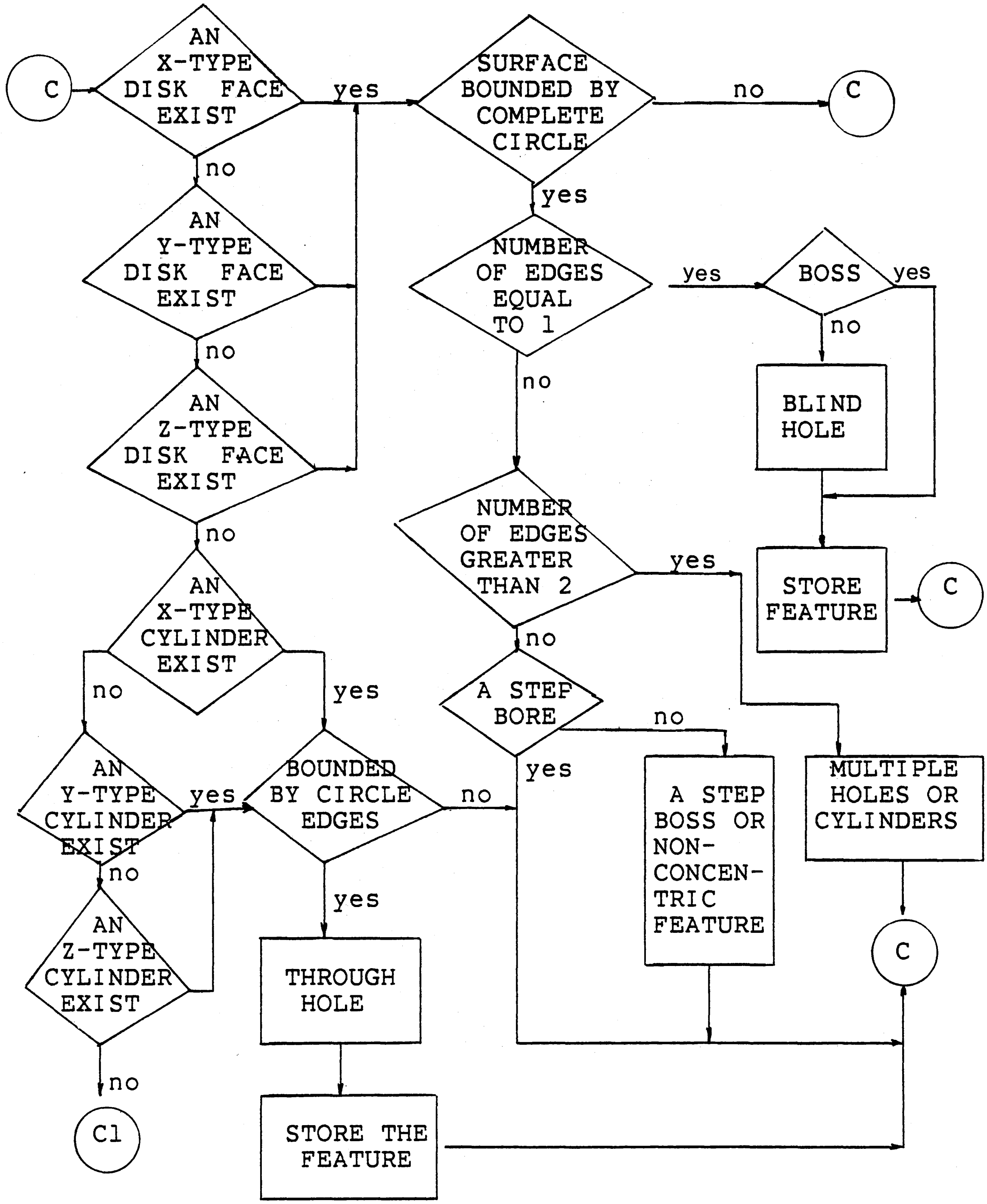

A
GET THE FACE AND EDGE INFORMATION
CREATION OF EDGE FILE
CREATION OF EDGE FILE
INITIALIZATION OF TOLERANCE & ROUGHNESS FILE
UPDATING THE B-FACE FILE
LAST SURFACE
yes
RETURN
no
RANK EDGES
A1
CREATION OF BOUNDARY LOOPS
RELATING INNER AND OUTER LOOPS
A2
CREATION OF F-FACE RECORDS
INITIALIZATION OF VARIABLES
GET THE EDGE NAMES FROM FACE FILE
DECIDING FACE TYPE
STORING EDGE DATA ACCORDING TO THE EDGE TYPE
ORDERING EDGE DATA ACCORDING TO THE EDGE AXIS OFFSET
FACE LINK TO ANOTHER FACE
yes
no
RETURN

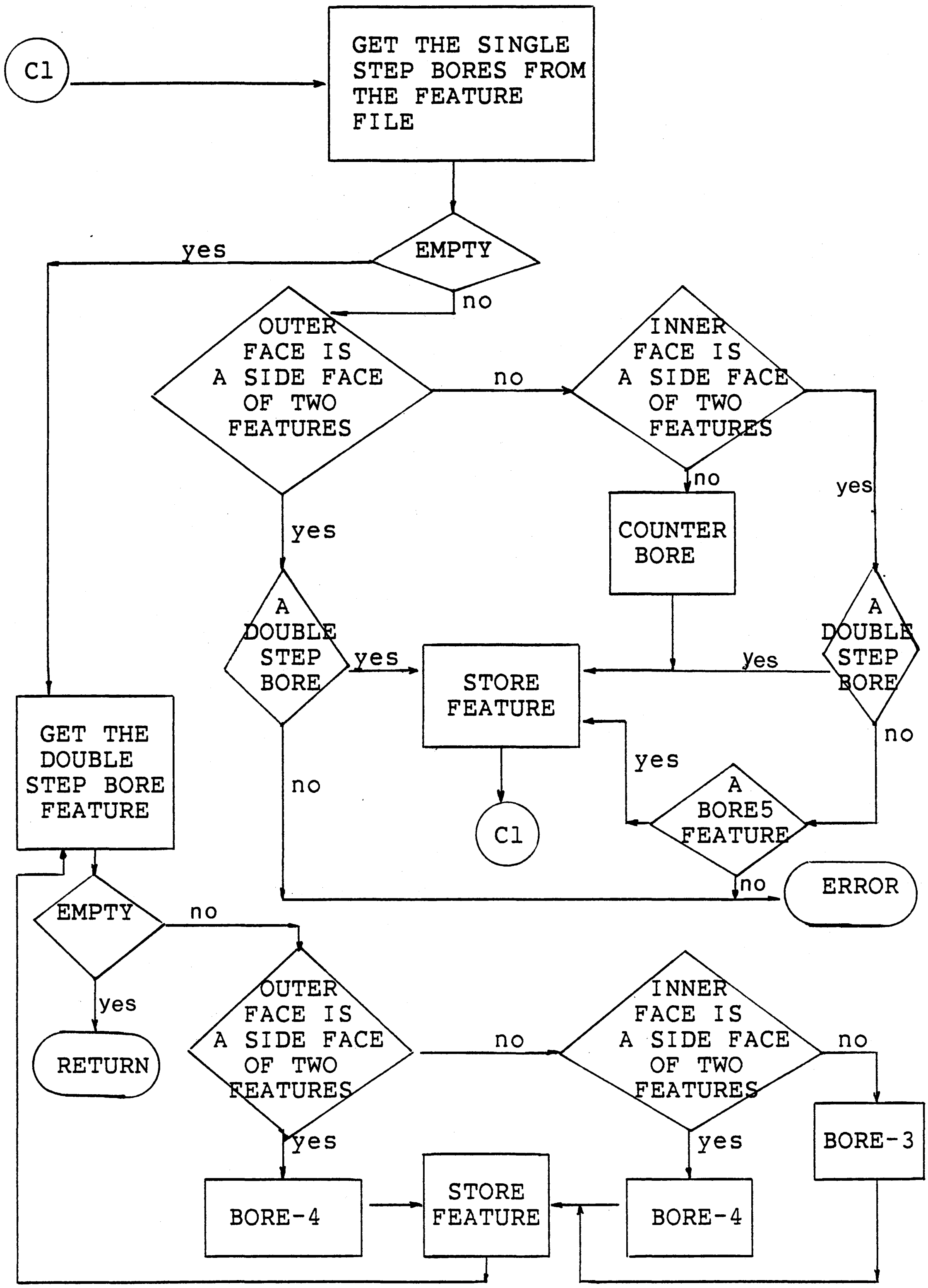

A1
INITIALIZATION OF VARIABLES
SET THE NODES OF THE LARGEST AXIS OFFSETS
A0
SET the STARTING EDGE. THE ONE WITH THE LOWEST AXIS OFFSET CUR- RENTLY AVAILABLE IN THE RIGHT GROUP
MARK THE EDGES OF THE LOOP
yes
SELECTING THE EDGE WITH THE LOWEST AXIS OFF- SET IN UP AXIS TYPE EDGES TO BE LINKED TO THE STARTING EDGE
no
A LOOP
yes
FORM A LOOP
AN ARC EDGE IS LINKED
no
ERROR
yes
ANY COMPLETE CIRCLE ARC
yes
BOTH EDGE TYPES HAVE NO AVAILABLE NODES
UP AXIS EDGES
no
no
RETURN
true
BOTH EDGE TYPES ARE NOT EMPTY
no
NO RIGHT AXIS TYPE EDGES
false
yes
AN ARC EDGE IS LINKED
ERROR
no
A11
yes
A0
yes
A LOOP
no

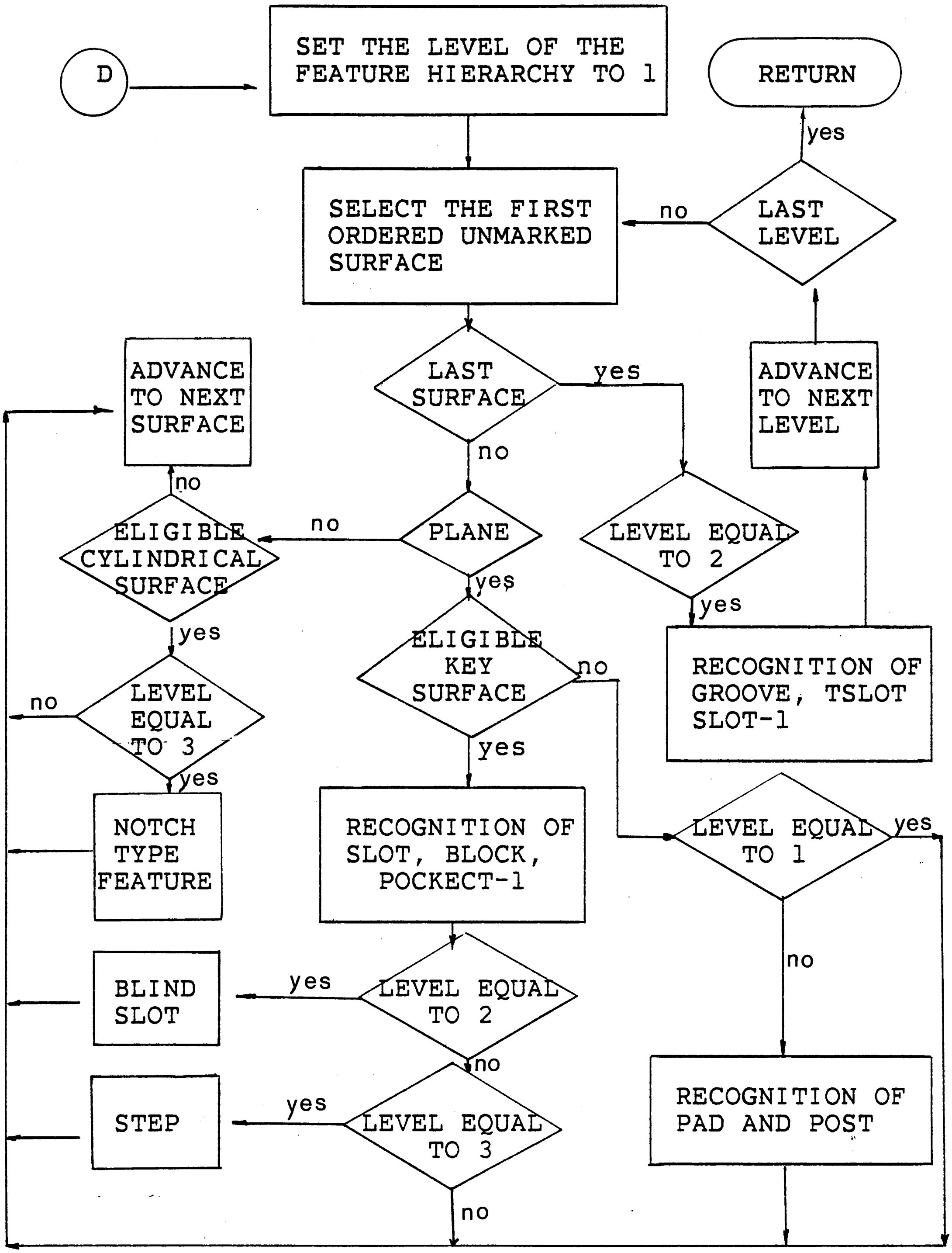

AN EDGE IS LINKED IN THE RIGHT AXIS GROUP
A11
yes
A13
AN EDGE IS LINKED IN THE UP AXIS GROUP
A LOOP
no
yes
A0
no
no
AB EDGE IN THE ARC GROUP
no
ERROR
yes
AN EDGE IN THE RIGHT GROUP
yes
A13
no
AN EDGE IN THE UP GROUP
no
ERROR
yes
A11

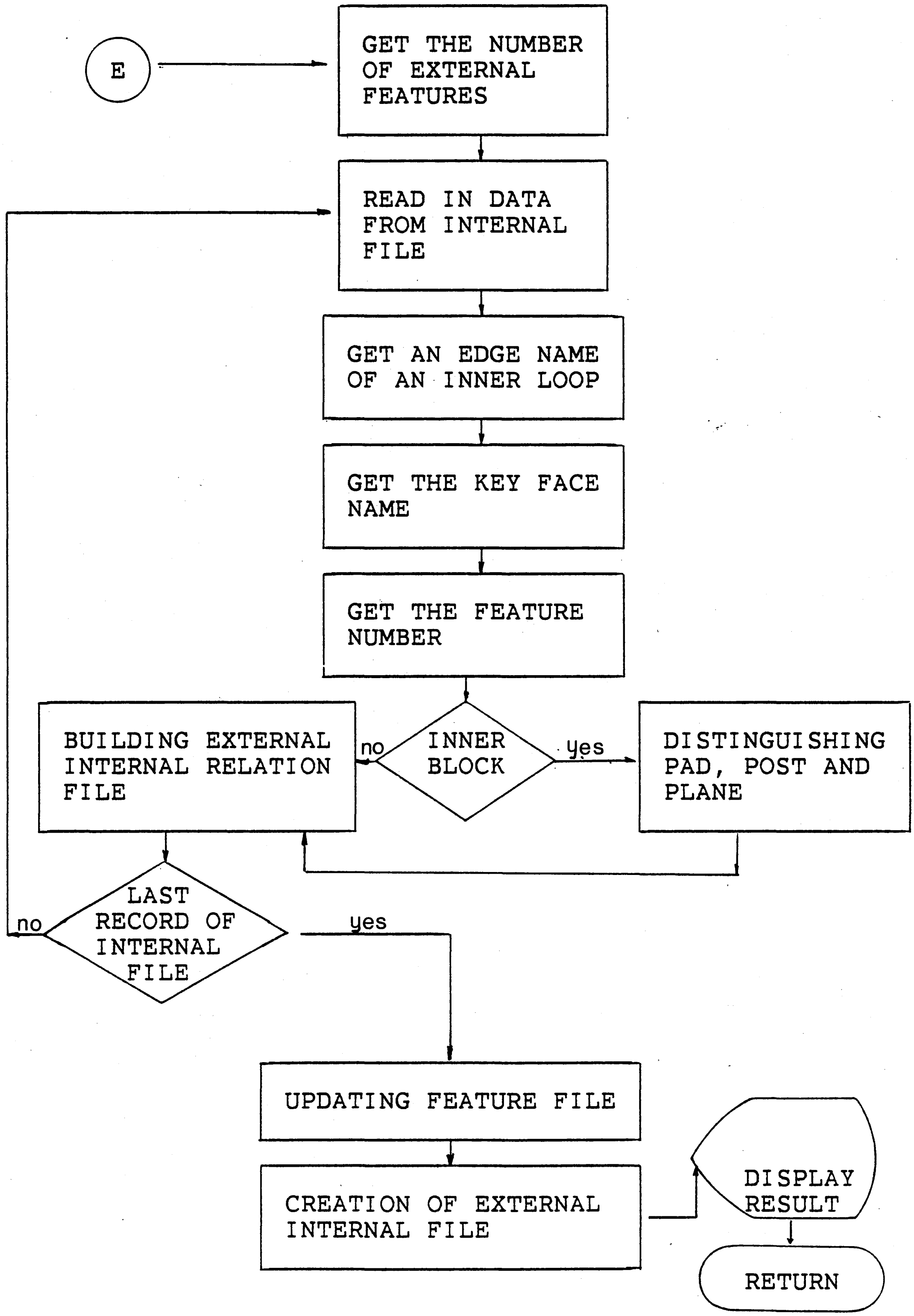

A2
GET THE FIRST NODE OF THE FIRST LOOP
DETERMINE THE EDGE TYPES OF THE GIVEN KEY SURFACE
SET THE LOWEST AND THE HIGHEST AXIS OFFSET OF THE FIRST LOOP
SET THE LOOP AS AN OUTER LOOP
LAST LOOP
yes
RETURN
no
DETERMINE THE LOWEST AND THE HIGHEST AXIS OFFSETS OF THE NEXT LOOP
AN OUTER LOOP
yes
no
RELATE THE INNER LOOP WITH THE OUTER LOOP
LAST LOOP
no
yes
RETURN


B
SURFACE ATTRIBUTES TO BE UPDATED
no
RETURN
yes
ENTER SURFACE TYPE (X, Y, OR Z)
ENTER SURFACE FINISH REQUIREMENTS
CYLINDRICAL (C) OR PLANE (P) SURFACE
P
C
ENTER THE POSITION OF A CORNER POINT
B
ENTER THE DIAMETER OF THE CYLINDER, THE MAX. & THE MIN. TOLERANCES OF THE DIAMETER
ENTER THE COORDINATE POSITION OF THE AXIS
B


C
AN X-TYPE DISK FACE EXIST
yes
SURFACE BOUNDED BY COMPLETE CIRCLE
no
C
no
AN Y-TYPE DISK FACE EXIST
yes
NUMBER OF EDGES EQUAL TO 1
yes
BOSS
yes
no
AN Z-TYPE DISK FACE EXIST
no
no
BLIND HOLE
NUMBER OF EDGES GREATER THAN 2
yes
STORE FEATURE
C
no
AN X-TYPE CYLINDER EXIST
no
yes
A STEP BORE
no
AN Y-TYPE CYLINDER EXIST
yes
BOUNDED BY CIRCLE EDGES
no
yes
A STEP BOSS OR NON-CONCEN-TRIC FEATURE
MULTIPLE HOLES OR CYLINDERS
no
AN Z-TYPE CYLINDER EXIST
yes
THROUGH HOLE
C
no
C1
STORE THE FEATURE


C1
GET THE SINGLE STEP BORES FROM THE FEATURE FILE
EMPTY
yes
no
OUTER FACE IS A SIDE FACE OF TWO FEATURES
no
INNER FACE IS A SIDE FACE OF TWO FEATURES
no
yes
COUNTER BORE
yes
A DOUBLE STEP BORE
yes
STORE FEATURE
yes
A DOUBLE STEP BORE
no
GET THE DOUBLE STEP BORE FEATURE
no
yes
A BORE5 FEATURE
C1
no
ERROR
EMPTY
no
yes
RETURN
OUTER FACE IS A SIDE FACE OF TWO FEATURES
no
INNER FACE IS A SIDE FACE OF TWO FEATURES
no
yes
yes
BORE-3
BORE-4
STORE FEATURE
BORE-4


D
SET THE LEVEL OF THE FEATURE HIERARCHY TO 1
RETURN
yes
SELECT THE FIRST ORDERED UNMARKED SURFACE
no
LAST LEVEL
ADVANCE TO NEXT SURFACE
LAST SURFACE
yes
ADVANCE TO NEXT LEVEL
no
no
ELIGIBLE CYLINDRICAL SURFACE
no
PLANE
LEVEL EQUAL TO 2
yes
yes
yes
ELIGIBLE KEY SURFACE
no
RECOGNITION OF GROOVE, TSLOT SLOT-1
no
LEVEL EQUAL TO 3
yes
yes
NOTCH TYPE FEATURE
RECOGNITION OF SLOT, BLOCK, POCKECT-1
LEVEL EQUAL TO 1
yes
no
BLIND SLOT
yes
LEVEL EQUAL TO 2
no
STEP
yes
LEVEL EQUAL TO 3
RECOGNITION OF PAD AND POST
no


E
GET THE NUMBER OF EXTERNAL FEATURES
READ IN DATA FROM INTERNAL FILE
GET AN EDGE NAME OF AN INNER LOOP
GET THE KEY FACE NAME
GET THE FEATURE NUMBER
INNER BLOCK
no
yes
BUILDING EXTERNAL INTERNAL RELATION FILE
DISTINGUISHING PAD, POST AND PLANE
LAST RECORD OF INTERNAL FILE
no
yes
UPDATING FEATURE FILE
CREATION OF EXTERNAL INTERNAL FILE
DISPLAY RESULT
RETURN

APPENDIX G

LISTING OF THE FEATURE RECOGNITION COMPUTER PROGRAM

A List of the Globle Variables in the Computer Program

CPOS :	The coordinates of the center of a cylinder along the RIGHT axis.
EEDGE :	Number of edges of a face.
EXAMF(I):	Flag to indicate if face I is available.
FEATURE :	Starting node name of a loop.
KL :	Key surface name.
LINK(I) :	Link field of node I.
MAINF(I):	Feature number of key face I.
MSFRL(I):	Node number which contains the key face of a side face I.
NETYP :	Edge type code.
NFEAT(N):	Feature type Number of the feature N.
NFTYP :	Face type code.
NMANF(N):	Key face name of feature N.
NNX :	Number of X-type plane and disk faces.
NNY :	Number of Y-type plane and disk faces.
NNZ :	Number of Z-type plane and disk faces.
NODE(I) :	Storage node I.
NSORT :	Faces are ranked according to the number of edges.
RANKE(I):	Ending position of the edge stored in node I.
RANKFL :	Variables for storing the nodes of a loop.
RANKL :	Sequenced edges according to the edge axis offset.
RANKN(I):	Edge name of node I.

(Continued)

RANKP(I)	:	Edge axis offset of edge RANKN(I).
RANKS(I)	:	Starting position of the edge stored in node I.
XCYL	:	X-type cylindrical surfaces.
XFACE	:	Ordered X-type plane and disk faces according to the surface axis offset.
XPOS	:	Surface axis offsets of the plane and disk faces, or the coordinates of the center of a cylinder along the RIGHT axis.
YCYL	:	Y-type cylindrical surfaces.
YFACE	:	Ordered Y-type plane and disk faces according to the surface axis offset.
ZCYL	:	Z-type cylindrical surfaces.
ZFACE	:	Ordered Z-type plane and disk faces according to the surface axis offset.

Program Calling Relation

CALLING PROGRAM	CALLED PROGRAM
MAIN	FOPEN, FCEEDG, READ20, FACESORT, ROUFINPT HOLBOR, BLOCGROV, INTERF
ASINF	GETNOD
BLOCAG	EDGERANK, EDGELINK, PAD, DABAS1, WRIT22
BLOCGROV	EDGERANK, EDGELINK, BLOCAG, DIRECT, DABASE, FACEN, WRIT22, GROVBRDG
DABASE	DABAS1
DABAS1	GETBLN, WRIT21, ASIGNF
EDGELINK	NXLINK, NYLINK,NZLINK
EDGERANK	READF, READE
FACESORT	READ20, READF, BUBLE
FCEEDG	WRITEF, WRITROUF, READF, EDGERANK, EDGELINK, EDGEUPDT, INEREL, READE,WRITEE
HOLBOR	FACEN, DABASE, READ20, READ21
GETBLN	READ20, WRIT20
GROVBRDG	READ20, READ21, DABAS1, CHEKGROV, DABASE
INEREL	MINMAX, INOUTF

(Continued)

INFLIN	GETNOD
INOUTF	FACEN
INTERF	READ20, READ22, READ21, WRIT21, INFLIN, WRIT22
READ20	WRIT20
ROUFINPT	ROUFP, ROUFR, WRITOUF, READF
WRIT22	READ20, WRIT20

```
C       ***********************************************************
C       *       THIS IS THE MAIN PROGRAM OF THE FEATURE RECOGNITION *
C       * PROCEDURE WHICH CALLS SUBROUTINES TO DO THE JOB OF        *
C       * F-FACE CREATION, PROMPTING FOR SURFACE FINISH ATTRIBUTES,*
C       * AND FEATURE RECOGNITION.  VARIABLES USED IN THIS MAIN     *
C       * PROGRAM ARE EXPLAINED AS FOLLOWS:                         *
C       * NFTYP : FACE TYPE CODE                                    *
C       * NETYP : EDGE TYPE CODE                                    *
C       * XPOS  : SUFACE AXIS OFFSET FOR THE PLANES AND DISKS, OR   *
C       *         THE COORDINATES OF THE CENTER OF A CYLINDER       *
C       *         ALONG THE RIGHT AXIS
C       * CPOS  : THE COORDINATES OF THE CENTER OF A CYLINDER       *
C       *         ALONG THE UP AXIS
C       * XFACE, YFACE, ZFACE : ORDERED X-TYPE, Y-TYPE AND Z-TYPE   *
C       *                       SURFACES                            *
C       * XCYL, YCYL, ZCYL : ORDERED X-TYPE, Y-TYPE, AND Z-TPE      *
C       *                    CYLINDRICAL SURFACES                   *
C       * NSORT : SURFACES ARE RANKED ACCORDING TO THE NUMBER OF    *
C       *         EDGES OF A SURFACE                                *
C       * EEDGE : NUMBER OF EDGES A SURFACE HAS.  SURFACES ARE      *
C       *         SEQUENCED BY SURFACE NAMS                         *
C       * NNX, NNY, NNZ : NUMBER OF X-TYPE, Y-TYPE, AND Z-TYPE      *
C       *                 PLANE AND DISC SURFACES.                  *
C       ***********************************************************
C
        DIMENSION NETYP(99),NFTYP(99),XPOS(99),CPOS(99)
        INTEGER XFACE(35),YFACE(35),ZFACE(35),NSORT(99),EEDGE(99)
        INTEGER XCYL(35),YCYL(35),ZCYL(35)
        COMMON /INOUT/IW,NFTYP,NETYP,XPOS,CPOS
        COMMON /INFOF/XFACE,YFACE,ZFACE,XCYL,YCYL,ZCYL
        DATA IW/12/,I2/2/
C
C       OPEN FILES
C
        CALL FOPEN
C
C       CREATION OF F-FACES
C
        CALL FCEEDG
C
C       GET THE NUMBER OF EDGES FROM DIRECTORY FILE
C
        CALL READ20(NN,I2)
C
C       CREATE EDGE NAME - EDGE TYPE RELATION
C
        DO 20 I=1,NN
        II=I
    20  READ(10,REC=II,FMT=30)IM,NETYP(I)
    30  FORMAT(I3,I6)
C
C       GROUP SURFACES ACCORDING TO THE FACE TYPE CODE
C
        CALL FACESORT(NNX,NNY,NNZ,KK,NSORT(1),EEDGE(1))
C
C       INPUT OF SURFACE ROUGHNESS AND TOLERANCE OF DIAMETER
C
        CALL ROUFINPT(NNX,NNY,NNZ,XPOS(1))
C
C       RECOGNITION OF CYLINDRICAL FEATURES
C
        CALL HOLBOR(NNX,NNY,NNZ)
C
C       RECOGNITION OF NON-CYLINDRICAL FEATURES
C
        CALL BLOCGROV(KK,NSORT(1),EEDGE(1))
C
C       RELATING INTERNAL FEATURE WITH EXTERNAL FEATURE
C
        CALL INTERF
C
        END
```

```
C
      SUBROUTINE FOPEN
C
C
C      MYDATA.DAT  : THE BOUNDARY FILE
C      FACE.DAT    : THE FACE FILE
C      EDGE.DAT    : THE EDGE FILE
C      FOVL.DAT    : THE FACE RECORD OVERFLOW FILE
C      EOVL.DAT    : THE EDGE RECORD OVERFLOW FILE
C      DIRC.DAT    : THE DIRECTORY FILE
C      FEATURE.DAT : THE FEATURE FILE
C      INTERF.DAT  : INTERNAL FEATURE FILE
C      INOFRL.DAT  : INTERNAL EXTERNAL FEATURE RELATION FILE
C      TOLROF.DAT  : TOLERENCE AND ROUGHNESS FILE
C
      OPEN (UNIT=8,FILE='MYDATA.DAT',STATUS='OLD')
      OPEN (UNIT=9,FILE='FACE.DAT',STATUS='OLD',ACCESS='DIRECT',
     1     RECL=100,FORM='FORMATTED')
      OPEN (UNIT=10,FILE='EDGE.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=120,FORM='FORMATTED')
      OPEN (UNIT=12,FILE='RESULT.DAT',STATUS='NEW')
      OPEN (UNIT=13,FILE='FOVL.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=100,FORM='FORMATTED')
      OPEN (UNIT=14,FILE='EOVL.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=80,FORM='FORMATTED')
      OPEN (UNIT=20,FILE='DIRC.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=200,FORM='FORMATTED')
      OPEN (UNIT=21,FILE='FEATUR.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=60,FORM='FORMATTED')
      OPEN (UNIT=22,FILE='INTERF.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=60,FORM='FORMATTED')
      OPEN (UNIT=24,FILE='INOFRL.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=20,FORM='FORMATTED')
      OPEN (UNIT=25,FILE='TOLROF.DAT',STATUS='NEW',ACCESS='DIRECT',
     1     RECL=30,FORM='FORMATTED')
      RETURN
      END
C
C***********************************************************************
C                                                                      *
C**   THIS PROGRAM READS IN DATA FROM THE BOUNDARY FILE                *
C**   MYDATA.DAT AND A TEMPORARY FILE FACE.DAT AND CREATE TWO          *
C**   DIRECT ACCESS FILE FACE.DAT AND EDGE.DAT                         *
C**   NF: NUMBER OF FACES OF A PART                                    *
C**   NE: NUMBER OF EDGES OF A PART                                    *
C**   SIDM(6): CENTRTROID POINT AND SIZE DIMENSION                     *
C**   NCEG: EDGE NUMBER                                                *
C**   NTEG(99): THE RECORD NUMBER OF THE STARTING EDGE OF A FACE       *
C                                                                      *
C***********************************************************************
      SUBROUTINE FCEEDG
      DIMENSION DATA(6),SIDM(6),NOEG(99),NTEG(99),DDTT(12),TTDD(12)
      DIMENSION NFACE(99),NEDGE(99),NSORT(99)
      DIMENSION RANKN(99),RANKFL(99),NFNAM(99),FEATURE(20),
     1          RANKP(99),RANKS(99),RANKE(99),EDGSML(9),EDGLAR(9),
     2          RANKR(99),RANKL(99),KONER(4),IDO(99),NFTYP(99),
     3          NETYP(99)
      INTEGER RANKN,RANKR,RANKL,RANKFL,FEATURE,EDGSML,EDGLAR
      DIMENSION NMIX(20),KFC(20),XPOS(99)
      INTEGER DIR(100),ROUF
      LOGICAL LIM1,LIM2,LIM3,LIF1,TRUE
      LOGICAL BDY,FEG,FBD,FLG,NDG,PLN
      COMMON /INOUT/IW,NFTYP,NETYP,XPOS
      DATA DIR/100*0/,I1/1/,NFACE/99*0/,NEDGE/99*0/
      DATA ROUF/125/,XTOL/0.001/,STOL/-0.001/
C
      NCEG=1
      IGN=0
      TRUE=.TRUE.
C
C     READ IN THE NUMBER OF FACES AND NUMBER OF EDGES
C
```

```
      READ(8,11)NF,NE
   11 FORMAT(2I3)
C
C     ICON : OVERFLOW RECORD NUMBER FOR EDGE FILE
C     NRC  : OVERFLOW RECORD NUMBER FOR FACE FILE
C     NFF  : A COUNTER TO COUNTER THE NUMBER OF
C            FACES THAT HAVE BEEN PROCESSED
C
      ICON=1
      NRC=1
      NFF=0
C
      READ(8,21)(SIDM(I),I=1,6)
   21 FORMAT(6F13.5)
C
   40 CONTINUE
      NFF=NFF+1
      IF (NFF .GT. NF)GO TO 300
C
C     READ IN THE FACE INFORMATION
C
      READ(8,41,END=300)IFC,ITY,INE,IP,INM,LIF1
      NSORT(IFC)=IFC
C     WRITE(6,41)IFC,ITY,INE
      NFC=IFC
   41 FORMAT(I3,I5,3I3,L2)
      NTEG(IFC)=NCEG
      READ(8,51)(I,DATA(I),I=1,3)
   51 FORMAT(3(I1,F13.5))
C
C*****READ IN EDGE DATA
C
      JJ=0
      DO 200 II=1,INE
      READ(8,61)IEG,IYP,INAM,LIM1,LIM2,LIM3,IF1,IF2
   61 FORMAT(I3,I6,I3,3L2,2I3)
      READ(8,71)(I,DDTT(I),I=1,6)
      IF (IYP .GT. 2500)READ(8,71)(I,DDTT(I),I=7,12)
   71 FORMAT(6(I2,F13.5))
C
C     CHECK IF EDGE II IS A BOUNDARY EDGE
C
      IF (LIM1 .NE. TRUE .AND. LIM2 .EQ. TRUE)GO TO 80
      IF (IF1 .LT. IFC)NSORT(IFC)=IF1
      READ(9,REC=IF1,FMT=61)IF1,ITY1,INE1
      READ(9,REC=IF2,FMT=61)IF2,ITY2,INE2
C
C     CHECK IF THE TWO CONNECTED FACES ARE PLANES OR DISKS
C
      IF(ITY1 .GT. 250 .OR. ITY2 .GT. 250)GO TO 80
      IF (IF1 .LT. IFC)GO TO 200
   75       IF(NFACE(NFC) .NE. 0)GO TO 77
               NFACE(NFC)=IF2
               GO TO 200
   77       NFC=NFACE(NFC)
      IF (NFC .NE. IF2)GO TO 75
   80 IF (IF1 .LT. IFC)GO TO 100
      JJ=JJ+1
      NOEG(JJ)=NCEG
      IGN=IGN+1
      ILIN=0
      IF(IYP.GT.2500)ILIN=ICON
C
C     CREATION OF EDGE FILES
C
      WRITE(10,REC=NCEG,FMT=81)IGN,IYP,INAM,L1,L2,L3,IF1,IF2,
     1ILIN,(J,DDTT(J),J=1,6)
      NETYP(NCEG)=IYP
   81 FORMAT(I3,I6,I3,3I2,2I3,I6,6(I2,F9.5))
      NCEG=NCEG+1
      IF (IYP .LT. 2500)GO TO 200
      WRITE(14,REC=ICON,FMT=97)(K,DDTT(K),K=7,12)
```

```
   97 FORMAT(6(I2,F9.5))
      ICON=ICON+1
      GO TO 200
  100 INC=NTEG(IF1)
C
C*****THIS EDGE HAS BEEN STORED ON DISC
C
  110 READ(10,REC=INC,FMT=81)ING,IYP,INAM,L1,L2,L3,IF1,IF2,
     1ICON,(J,TTDD(J),J=1,6)
      IF(IFC .EQ. IF2)GO TO 130
  120 INC=INC+1
      GO TO 110
  130 DO 140 I=1,6
      IF(DDTT(I) .NE. TTDD(I))GO TO 120
  140 CONTINUE
      JJ=JJ+1
      NOEG(JJ)=INC
C
  200 CONTINUE
      NST=NSORT(IFC)
      LIN=NFACE(IFC)
C
C     CREATEION OF B-FACE FILE
C
      CALL WRITEF(IFC,ITY,JJ,IP,INM,LIF1,DATA(1),LIN,ILIN,NOEG(1),
     1NST,NRC)
      NFTYP(IFC)=ITY
      XPOS(IFC)=DATA(1)
C
C     INITIALIZATION OF TOLERANCE - ROUGHNESS FILE
C
      CALL WRITROUF(IFC,ROUF,XTOL,STOL)
C
  220 IF (IFC .LE. NFC)GO TO 40
      IFC=NFC
C
      CALL READF(IFC,ITY,INE,IP,INM,LIF1,DATA(1),M1,ILIN,NOEG(1),
     1           INF)
      MF1=NFACE(NFC)
      NDF=ILIN
      CALL WRITEF(IFC,ITY,INE,IP,INM,LIF1,DATA(1),MF1,ILIN,NOEG(1),
     1INF,NDF)
      GO TO 40
C
C     UPDATING B-FACE FILE
C
  300 CONTINUE
      WRITE(6,*)'           START TO CREATE F-FACE'
C
      KK=NF
      DO 2000 LR=1,NF
      IF (NEDGE(LR) .GT. 0)GO TO 2000
      IIF=1
      IFC=LR
      KFC(IIF)=IFC
      CALL READF(IFC,ITY,INE,IP,INM,LIF1,DATA(1),M1,ILIN,NOEG(1),
     1INF)
      NEDG=INE
      NEDGE(LR)=1
C
C     CHECK IF THIS SURFACE IS A PLANE SURFACE
C
      IF (ITY .GT. 103)GO TO 2000
C
C     CHECK IF THIS B-FACE IS COMBINED WITH ANOTHER ONE
C     IT IS TRUE IF M1 IS GREATER THAN 0
C
 1200 IF (M1 .LT. 1)GO TO 1300
      NE=NEDG+1
      IIF=IIF+1
      KFC(IIF)=M1
      CALL READF(M1,ITY,INE,IP,INM,LIF1,DATA(1),LIN,LN2,NOEG(NE),
```

```
     1INF)
      NEDGE(M1)=1
      NEDG=NEDG+INE
      M1=LIN
      GO TO 1200
 1300 LLR=LR
C
C     ORGANIZE THE EDGES OF A B-FACE OR COMBINED B-FACES
C
      CALL EDGERANK(NX,NY,LLR,LLR,RANKN(1),RANKP(1),RANKS(1),
     1RANKE(1),RANKL(1),RANKFL(1),EDGSML(1),EDGLAR(1),IDO(1),
     1NFNAM(1),RANKR(1))
C
C     CONSTRUCTING OUTER AND INNER BOUNDARY LOOPS
C
      CALL EDGELINK(NX,NY,LOPIN,INRL,KONER(1),RANKN(1),RANKP(1),
     1RANKS(1),RANKE(1),RANKL(1),RANKFL(1),EDGSML(1),EDGLAR(1),
     2IDO(1),NFNAM(1),FEATURE(1),RANKR(1))
C
      IF (LOPIN .GT. 1)GO TO 1500
 1400 IF (IIF .LT. 2)GO TO 2000
      KK=KK+1
C
C     CREATION OF AN F-FACE
C
      CALL WRITEF(KK,ITY,NEDG,IP,INM,LIF1,DATA(1),IO,IL,NOEG(1),
     1INF,NRC)
      NFTYP(KK)=ITY
      XPOS(KK)=DATA(1)
      CALL WRITROUF(KK,ROUF,XTOL,STOL)
      CALL EDGEUPDT(KK,KFC(1),IIF,NOEG(1),NEDG)
      GO TO 2000
C
 1500 IGB=1
      IWR=0
      LLR=LR
C
C     RELATING THE OUTER AND INNER LOOPS
C
      CALL INEREL(LLR,IWR,LOPIN,IGB,NMIX(1),FEATURE(1),RANKN(1),
     1RANKP(1),RANKS(1),RANKE(1),RANKFL(1))
      IF (IGB .EQ. 1)GO TO 1400
      DO 1800 I=1,LOPIN
          K=0
          N=NMIX(I)
          IF (N .LT. 0)GO TO 1800
          DO 1700 J=1,LOPIN
              M=ABS(NMIX(J))
C
              IF (M .NE. N)GO TO 1700
                  JJ=FEATURE(J)
                  J1=JJ
 1600             NAM=RANKN(J1)
                  K=K+1
                  NOEG(K)=NAM
                  J1=RANKFL(J1)
                  IF (J1 .NE. JJ)GO TO 1600
 1700     CONTINUE
          KK=KK+1
C
C     CREATION OF AN F-FACE
C
          CALL WRITEF(KK,ITY,K,IP,INM,LIF1,DATA(1),IO,IL,
     1                   NOEG(1),INF,NRC)
          NFTYP(KK)=ITY
          XPOS(KK)=DATA(1)
          CALL WRITROUF(KK,ROUF,XTOL,STOL)
          CALL EDGEUPDT(KK,KFC(1),IIF,NOEG(1),K)
 1800 CONTINUE
 2000 CONTINUE
C
C     INITIALIZATION OF FILE DIRECTORY
```

```
C
      NCEG=NCEG-1
      WRITE(20,REC=1,FMT=22)KK,NCEG,I1,I1,I1,I1
   22 FORMAT(6I4)
      DO 30 K=2,3
      WRITE(20,REC=K,FMT=25)(DIR(J),J=1,100)
   25 FORMAT(100I2)
   30 CONTINUE
      RETURN
      END
C
      SUBROUTINE WRITEF(IFC,ITY,JJ,IP,INM,LIF1,DATA,NFC,ILIN,
     1NOEG,ISO,NRC)
C*************************************************************
C     THIS SUBROUTINE WRITES DATA TO FACE FILE               *
C*************************************************************
C     IFC  : FACE NUMBER WHICH IS ALSO A RECORD NUMBER
C     ITY  : FACE TYPE NUMBER
C     JJ   : NUMBER OF EDGES
C     IP   : RELATED P-FACE NUMBER
C     INM  : DUMMY VARIABLE
C     LIF1 : DUMMY VARIABLE
C     DATA : FACE POSITION
C     NFC  : COMBINED B-FACE NUMBER
C     ILIN : LINKED FACE NUMBER
C     NOEG : EDGE NAMES OF THE SURFACE
C     ISO  : ORIGINAL B-FACE NAME
C     NRC  : B-FACE NUMBER TO BE LINKED
C
      DIMENSION DATA(6),NOEG(99)
      LOGICAL LIF1
      NRCD=IFC
      ILIN=0
      IF(JJ .GT. 12)ILIN=NRC
      IF (JJ .LT. 12) THEN
          DO 202 I=JJ+1,12
  202     NOEG(I)=0
      ENDIF
      WRITE(9,REC=NRCD,FMT=201)IFC,ITY,JJ,IP,INM,LIF1,
     1(I,DATA(I),I=1,3),NFC,ILIN,(NOEG(I),I=1,12),ISO
  201 FORMAT(I3,I5,3I3,L2,3(I1,F9.5),15I3)
      IF(JJ.LE.12)GO TO 220
      WRITE(13,REC=NRC,FMT=211)(NOEG(I),I=13,JJ)
  211 FORMAT(33I3)
      NRC=NRC+1
  220 RETURN
      END
C
      SUBROUTINE READF(IFC,ITY,JJ,IP,INM,LIF1,DATA,NFC,ILIN,
     1                 NOEG,ISO)
C*****************************************************************
C     THIS SUBROUTINE READS THE DATA FROM FACE FILE.             *
C     VARIBLE NAMES ARE THE SAME AS DESCRIBED IN                 *
C     SUBROUTINE WRITEF.                                         *
C*****************************************************************
      DIMENSION DATA(6),NOEG(99)
      LOGICAL LIF1
      NRCD=IFC
      READ(9,REC=NRCD,FMT=201)IFC,ITY,JJ,IP,INM,LIF1,
     1(I,DATA(I),I=1,3),NFC,ILIN,(NOEG(I),I=1,12),ISO
  201 FORMAT(I3,I5,3I3,L2,3(I1,F9.5),15I3)
      IF (ILIN .EQ. 0)GO TO 220
      READ(13,REC=ILIN,FMT=211)(NOEG(I),I=13,JJ)
  211 FORMAT(33I3)
      NRC=NRC+1
  220 RETURN
      END
C
      SUBROUTINE EDGEUPDT(KK,KFC,IIF,NOEG,NEDG)
C***********************************************************
C     EDGEUPDT UPDATES THE EDGE FILE RECORDS               *
C***********************************************************
```

```
C      KK   : NEW SURFACE NAME
C      KFC  : OLD SURFACE NAME(S)
C      IIF  : NUMBER OF B-FACES
C      NOEG : EDGE NAMES
C      NEDG : NUMBER OF EDGES
C
       DIMENSION KFC(20),NOEG(99),DDTT(12),NFTYP(99)
       COMMON /INOUT/IW,NFTYP
C

       DO 100 I=1,NEDG
          NRC=NOEG(I)
          CALL READE(NRC,IYP,INAM,IF1,IF2,ILIN,DDTT(1))
          DO 50 J=1,IIF
             IF0=KFC(J)
             NRF=NFTYP(IF0)
C
C      MARK THE B-FACE WHICH IS NOT A F-FACE
C
             IF(NRF .GT. 0)NFTYP(IF0)=-NRF
             IF (IF1 .EQ. IF0) THEN
                IF1=KK
                GO TO 60
             ELSE IF (IF2 .EQ. IF0) THEN
                     IF2=KK
                     GO TO 60
             ELSE
             ENDIF
   50   CONTINUE
             WRITE(6,*)' ERROR IN UPDATING EDGE RECORD'
   60        IF (IF1 .LT. IF2)GO TO 70
                NFA=IF1
                IF1=IF2
                IF2=NFA
   70        CALL WRITEE(NRC,IYP,INAM,IF1,IF2,ILIN,DDTT(1))
  100  CONTINUE
       RETURN
       END
C
       SUBROUTINE READE(NM,IYP,NAM,IF1,IF2,ILIN,DDTT)
C*****************************************************************
C      THIS SUBROURTINE READS THE DATA FROM THE EDGE FILE        *
C*****************************************************************
C      NM   : EDGE NAME WHICH IS THE RECORD NUMBER
C      IYP  : EDGE CODE
C      NAM  : DUMMY VARIABLE
C      IF1  : FACE NAME
C      IF2  : FACE NAME
C      ILIN : OVER FLOW RECOD NUMBER
C      DDTT : EDGE DATA
C
       DIMENSION DDTT(12)
C
       READ(10,REC=NM,FMT=100)NAM,IYP,INAM,L1,L2,L3,IF1,IF2,ILIN,
      1(J,DDTT(J),J=1,6)
  100  FORMAT(I3,I6,I3,3I2,2I3,I6,6(I2,F9.5))
       IF (IYP .GT. 2500)READ(14,REC=ILIN,FMT=200)(J,DDTT(J),J=7,12)
  200  FORMAT(6(I2,F9.5))
       RETURN
       END
C
       SUBROUTINE WRITEE(NM,IYP,NAM,IF1,IF2,ILIN,DT)
       DIMENSION DT(12)
C*********************************************************************
C      WRITEE WRITES EDGE INFORMATION TO THE EDGE FILE               *
C      VARIABLES ARE DESCRIBED IN SUBROUTINE READE                   *
C*********************************************************************
       WRITE(10,REC=NM,FMT=100)NM,IYP,INAM,L1,L2,L3,IF1,IF2,
      1ILIN,(J,DT(J),J=1,6)
  100  FORMAT(I3,I6,I3,3I2,2I3,I6,6(I2,F9.5))
       IF (IYP .GT. 2500)WRITE(14,REC=ILIN,FMT=200)(J,DT(J),J=7,12)
  200  FORMAT(6(I2,F9.5))
```

```
      RETURN
      END
C
      SUBROUTINE WRITROUF(NN,ROUF,XTOL,STOL)
C*************************************************************
C     WRITOUF WRITES SURFACE ROUGHNESS, MAXIMUM              *
C     AND MINIMUM TOLERANCE TO FILE TOLROF.                  *
C*************************************************************
C     NN   : SURFACE NUMBER
C     ROUF : SURFACE ROUGHNESS
C     XTOL : MAXIMUM TOLERANCE
C     STOL : MINIMUM TOLERANCE
C
      INTEGER ROUF
C
      WRITE(25,REC=NN,FMT=100)ROUF,XTOL,STOL
  100 FORMAT(I10,2F10.5)
      RETURN
      END
C
      SUBROUTINE READ21(NFN,NFNUM,NN,NSR)
C*************************************************************
C**   READ21 READS RECORD NFN FROM FEATURE FILE              *
C*************************************************************
C     NFNUM : FEATURE TYPE NUMBER
C     NN    : NUMBER OF SURFACES
C     NSR   : SURFACE NAMES
C
      DIMENSION NSR(20)
C
      READ(21,REC=NFN,FMT=100)NFN1,NFNUM,NN,(NSR(J),J=1,NN)
  100 FORMAT (30I2)
C
      RETURN
      END
C
C
      SUBROUTINE WRIT21(NFN,NFNUM,NN,NSR)
C************************************************************
C**   WRIT21 WRITES RECORD NFN TO FEATURE FILE              *
C************************************************************
      DIMENSION NSR(20)
C
      WRITE(21,REC=NFN,FMT=100)NFN,NFNUM,NN,(NSR(J),J=1,NN)
  100 FORMAT (30I2)
C
      RETURN
      END
C
C
      SUBROUTINE READ20(NRC,NR)
C***********************************************************
C**   READ20 READS RECORD 1 DATA FROM THE DIRECTORY FILE *
C***********************************************************
C
      READ(20,REC=1,FMT=100)NF,NE,NFNUM,NFD,NIF,NLR
  100 FORMAT (10I4)
C
C**   NF: NUMBER OF FACES
C**   NE: NUMBER OF EDGES
C**   NFNUM: CURRENT AVAILABLE FEATURE NUMBER
C**   NFD: CURRENT AVAILABLE FIELD NUMBER OF RECORD 3
C**   NIF: RECORD NUMBER AVAILABLE FOR INTERNAL FEATURE FILE
C**   NLR: RECORD NUMBER FOR THE FINAL RESULT OF THE SYSTEM
C
      GO TO (200,300,400,500,600,700),NRC
  200 NR=NF
      RETURN
  300 NR=NE
      RETURN
  400 NR=NFNUM
      GO TO 800
```

```
  500 NR=NFD
      GO TO 800
  600 NR=NIF
      GO TO 800
  700 NR=NLR
      NLR=NLR+1
      CALL WRIT20(NRC,NLR)
  800 RETURN
      END
C
C
      SUBROUTINE WRIT20(NRC,NR)
C***********************************************************
C**   WRIT20 WRITES DATA TO THE FIRST RECORD OF            *
C**   THE DIRECTORY FILE                                   *
C***********************************************************
      READ(20,REC=1,FMT=100)NF,NE,NFNUM,NFD,NIF,NLR
  100 FORMAT (10I4)
C
C**   NF: NUMBER OF FACES
C**   NE: NUMBER OF EDGES
C**   NFNUM: CURRENT AVAILABLE FEATURE NUMBER
C**   NFD: CURRENT AVAILABLE FIELD NUMBER OF RECORD 3
C**   NIF: RECORD NUMBER AVAILABLE FOR INTERNAL FEATURE FILE
C
      GO TO (200,300,400,500,600,700),NRC
  200 NF=NR
C     THIS SUBROUTINE EDGE INFORMATION TO EDGE FILE
C     VARIABLES ARE DESCRIBED IN SUBROUTINE READE
C
      RETURN
  300 NE=NR
      RETURN
  400 NFNUM=NR
      GO TO 800
  500 NFD=NR
      GO TO 800
  600 NIF=NR
      GO TO 800
  700 NLR=NR
  800 WRITE(20,REC=1,FMT=100)NF,NE,NFNUM,NFD,NIF,NLR
      RETURN
      END
C
C
      SUBROUTINE WRIT22(NRC,NFN,KK,KL,NSR)
C***********************************************************
C**   WRIT22 WRITES DATA TO INTERNAL FILE (INTERF)         *
C***********************************************************
C**   NFN: FEATURE NUMBER
C**   KK: NUMBER OF INTERNAL FEATURE
C**   NSR: SIDE SURFACES NAME
C
      DIMENSION NSR(20)
C
      IF (NRC .GT. 0)GO TO 10
         NN5=5
         CALL READ20(NN5,NRC)
         IRC=NRC+1
         CALL WRIT20(NN5,IRC)
C
   10 WRITE(22,REC=NRC,FMT=100)NFN,KK,KL,(NSR(I),I=1,KK)
  100 FORMAT(30I2)
C
      RETURN
      END
C
C
      SUBROUTINE READ22(NRC,NFN,KK,KL,NSR)
C*********************************************************
C**   READ22 READS DATA FROM INTERNAL FEATURE FILE       *
C*********************************************************
```

```
C**     NFN: FEATURE NUMBER
C**     KK: NUMBER OF INTERNAL FEATURE
C**     NSR: SIDE SURFACES NAME
C
        DIMENSION NSR(20)
C
C
     10 READ(22,REC=NRC,FMT=100)NFN,KK,KL,(NSR(I),I=1,KK)
    100 FORMAT(30I2)
C
        RETURN
        END
C
        SUBROUTINE READ24(NRC,N1,N2,N3)
C**************************************************************
C       READ24 READS DATA FROM INTERNAL EXTERNAL RELATION FILE  *
C**************************************************************
C**     N1 INDICATES THE EXTERNAL(0) OR INTERNAL(1)
C**     N2 IS THE RECORD NUMBER OF FILE22 IF N1=0
C**        IS THE FEATURE NUMBER WHICH THE INTERNAL
C**        FEATURE BELONGS TO, IF N1 IS GREATER THAN 0
C**     N3 IS THE FEATURE NUMBER WHICH THE INTERNAL
C**        FEATURE BELONGS TO
C
        READ (24,REC=NRC,FMT=100)N1,N2,N3
    100 FORMAT(I6,I7,I7)
        RETURN
        END
C
C
        SUBROUTINE EDGERANK(NX,NY,MM,ML,RANKN,RANKP,RANKS,RANKE,
       1RANKL,RANKFL,EDGSML,EDGLAR,IDOIO,NFNAM,RANKR)
C
C*******************************************************************
C             THIS SUBROUTINE ORDERS THE EDGES OF THE SAME EDGE TYPE *
C       ACCORDING TO THE EDGE AXIS OFFSET.                          *
C       NX, NY : NUMBER OF EDGES OF RIGHT AXIS-TYPE                 *
C                AND UP AXIS-TYPE RESPECTIVELY                      *
C       MM     : THE ACTIVE SURFACE NAME                            *
C       ML     : THE ACTIVE SURFACE NAME                            *
C       RANKN  : EDGE NAME                                          *
C       RANKP  : EDGE AXIS OFFSET                                   *
C       RANKS  : STARTING POSITION OF AN EDGE                       *
C       RANKE  : ENDING POSITION OF AN EDGE                         *
C       RANKL  : EDGES LINKED FROM LOW AXIS OFFSET TO HIGH AXIS     *
C                OFFSET                                             *
C       RANKFL : THE LINK NODES OF LOOPS                            *
C       EDGSML : EDGE NAMES WITH THE LOWEST AXIS OFFSET             *
C       EDGLAR : EDGE NAMES WITHE THE HIFHEST AXIS OFFSET           *
C       IDOIO  : DUMMY VARIABLE                                     *
C       NFNAM  : FACE NAMES.  FACES THAT SHARE AN EDGE WITH THE     *
C                ACTIVE SURFACE.                                    *
C       RANKR  : EDGES RANKED FROM HIGH AXIS OFFSET TO LOW AXIS     *
C                OFFSET                                             *
C*******************************************************************
C
        DIMENSION NOEG(99),DATA(6),DDTT(12),EDGSML(9),EDGLAR(9)
        DIMENSION RANKN(99),RANKS(99),RANKR(99),RANKP(99),RANKL(99)
        DIMENSION RANKE(99),RANKFL(99),KONER(4),IDOIO(99),NSORT(99)
        DIMENSION NETYP(99),NFTYP(99),NFNAM(99)
        INTEGER RANKN,RANKR,RANKL,EDGSML,EDGLAR,RANKFL
        COMMON /INOUT/IW,NFTYP,NETYP
        LOGICAL LIF1,TRUE
C
        DO 490 I=1,9
        EDGSML(I)=0
    490 EDGLAR(I)=0
        TRUE=.TRUE.
C
C**     N1 IS THE NODE NUMBER
C
        N1=0
```

```
      MFACE=MM
      JFACE=MFACE
C     WRITE(IW,980)MFACE
      IFC=MM
  500 CALL READF(IFC,ITY,NEG,IP,INM,LIF1,DATA,LIN,ILIN,NOEG,ISO)
      NFP=ITY-100
      NTYP=IABS(NFP)
C
      DO 800 I=1,NEG
          NM=NOEG(I)
          CALL READE(NM,IYP,INAM,IF1,IF2,ILIN,DDTT)
  510     CONTINUE
          NNN=IYP-(IYP/1000)*1000
          IF(IYP .GT. 2000)NNN=NNN+3
          IF(IYP .GT. 3000)NNN=NNN+3
C
C***** EDGE TYPE NUMBER HAS BEEN TRANSFORMED TO THE NUMBER 1-9
C
          IF (ITY .GT. 150)GO TO 580
          GO TO (520,540,560),NTYP
C
C***** STORE EDGE DATA OF X-TYPE SURFACE
C
  520     IF (NNN .EQ. 2) THEN
C
C***** Y-TYPE EDGE DATA
C
             RDATA=DDTT(3)
             SRDATA=DDTT(2)
             ENDATA=DDTT(5)
          ELSE
C
C***** Z-TYPE EDGE DATA
C
             RDATA=DDTT(2)
             SRDATA=DDTT(3)
             ENDATA=DDTT(6)
          ENDIF
          NX=2
          NY=3
          GO TO 600
C
C***** STORE EDGE DATA OF Y-TYPE SURFACE
C
  540     IF (NNN .EQ. 3) THEN
C
C***** Z-TYPE EDGE
C
             RDATA=DDTT(1)
             SRDATA=DDTT(3)
             ENDATA=DDTT(6)
          ELSE
C
C***** X-TYPE EDGE
C
             RDATA=DDTT(3)
             SRDATA=DDTT(1)
             ENDATA=DDTT(4)
          ENDIF
          NX=1
          NY=3
          GO TO 600
C
C***** STORE EDGE DATA OF Z-TYPE FACE
C
  560     IF (NNN .EQ. 2) THEN
C
C***** Y-TYPE EDGE
C
             RDATA=DDTT(1)
             SRDATA=DDTT(2)
             ENDATA=DDTT(5)
```

```
              ELSE
C
C***** X-TYPE EDGE
C
                 RDATA=DDTT(2)
                 SRDATA=DDTT(1)
                 ENDATA=DDTT(4)
              ENDIF
              NX=1
              NY=2
              GO TO 600
C
  580    IF (ITY .GT. 250) THEN
            WRITE(6,585)
  585       FORMAT(1X,'***** THIS IS A CYLINDER FACE *****')
         ELSE
            WRITE(6,590)
  590       FORMAT(1X,'***** THIS IS A CIRCULAR PLANE FACE *****')
         ENDIF
         N1=0
         RETURN
C
  600         CONTINUE
              N1=N1+1
              RANKN(N1)=NM
              RANKP(N1)=RDATA
              RANKS(N1)=SRDATA
              RANKE(N1)=ENDATA
              IF (IF1 .EQ. JFACE) THEN
                  NFNAM(N1)=IF2
              ELSE
                  NFNAM(N1)=IF1
              END IF
              RANKL(N1)=0
              RANKR(N1)=0
              IF (EDGSML(NNN) .NE. 0)GO TO 700
              EDGSML(NNN)=N1
              EDGLAR(NNN)=N1
              GO TO 800
C
C****         LINK EDGES FROM LOW AXIS OFFSET TO HIGH AXIS OFFSET
C****         (RANKL) AND FORM HIGH TO LOW (RANKR).
C
  700         N2=EDGSML(NNN)
  710         IF (RDATA .LE. RANKP(N2))GO TO 760
  720         N3=RANKL(N2)
              IF (N3 .NE. 0)GO TO 730
              RANKL(N2)=N1
              EDGLAR(NNN)=N1
              RANKR(N1)=N2
              GO TO 800
  730         N0=N2
              N2=N3
              GO TO 710
  760         IF (RDATA .EQ. RANKP(N2))GO TO 790
  770         IF (N2 .NE. EDGSML(NNN))GO TO 780
              RANKL(N1)=N2
              EDGSML(NNN)=N1
              RANKR(N1)=0
              RANKR(N2)=N1
              GO TO 800
  780         RANKL(N1)=N2
              RANKL(N0)=N1
              RANKR(N2)=N1
              RANKR(N1)=N0
              GO TO 800
  790         IF(SRDATA .GT. RANKS(N2))GO TO 720
              GO TO 770
C
  800 CONTINUE
C
        IF (LIN .LE. 0)GO TO 900
```

```
  870 FORMAT(1X,'LIN=',I6)
      MM=LIN
      JFACE=LIN
      GO TO 500
  900 CONTINUE
      RETURN
      END
C
      SUBROUTINE EDGELINK(NX,NY,LOPIN,INL,KONER,RANKN,RANKP,RANKS,
     1RANKE,RANKL,RANKFL,EDGSML,EDGLAR,IDOIO,NFNAM,FEATURE,RANKR)
C
C*******************************************************************
C                                                                  *
C     SUGROUTINE EDGELINK CONSTRUCTS THE LOOPS OF A B-FACE OR      *
C     COMBINED B-FACES.  EDGES ARE STUDIED FROM THE ONE WITH THE   *
C     LOWEST AXIS OFFSET TO THE ONE WITH THE HIGHEST AXIS OFFSET.  *
C                                                                  *
C     NX   : EDGE CODE INDICATOR                                   *
C     NY   : EDGE CODE INDICATOR                                   *
C     LOPIN : NUMBER OF LOOPS ARE CONSTRUCTED IN THIS PROCEDURE    *
C     INL : DUMMY VARIABLE                                         *
C     KONER : DUMMY VARIABLE                                       *
C     FEATURE : THE NUMBER OF THE FIRST NODE OF A LOOP            *
C     RANKFL: NODES FOR LINKING A FEATURE                          *
C     IDOIO: STORES THE LOOP NUMBER AND INDICATES IF AN EDGE HAS   *
C              BEEN USED TO CONSTRUCT A LOOP.                      *
C     RANKN,RANKP,RANKS,RANKE,RANKL,RANKFL,SDGSML,SDGLAR,NFNAM,     *
C     AND RANKR ARE DEFINED IN SUBROUTINE EDGERANK.                *
C     ISEAR : 1 INDICATES SEARCH WAS SUCCESSFUL, 0 INDICATES       *
C               SEARCH WAS NOT SUCCESSFUL                          *
C                                                                  *
C*******************************************************************
      DIMENSION RANKN(99),RANKS(99),RANKR(99),RANKP(99),RANKL(99)
      DIMENSION RANKE(99),RANKFL(99),KONER(4),IDOIO(99),NFNAM(99)
      DIMENSION NETYP(99),NFTYP(99)
      INTEGER RANKN,RANKR,RANKL,EDGSML(9),EDGLAR(9),RANKFL,
     1        CONER(4),FEATURE(20)
      COMMON /INOUT/IW,NFTYP,NETYP
      COMMON /LLINK/AMAX1,AMAX2
      LOGICAL LIM1,LIM2,LIM3,LIF1,TRUE
C
C     AMAX1, AMAX2 : THE LARGEST AXIS OFFSET
C
      INL=0
      LOPIN=0
      DO 3 I=1,99
      RANKFL(I)=0
    3 IDOIO(I)=0
      DO 5 I=1,20
    5 FEATURE(I)=0
C
      MX1=EDGLAR(NX)
      AMAX1=RANKP(MX1)
      MX2=EDGLAR(NY)
      AMAX2=RANKP(MX2)
C
C     SET THE STARTING EDGES.  JM1 IS THE EDGE THAT HAS THE LOWEST
C     AXIS OFFSET IN THE RIGHT-EDGES GROUP.  JL1 IS THE EDGES THAT
C     HAS THE LOWEST AXIS OFFSET IN THE UP-AXIS EDGE GROUP.
C
   10 JM1=EDGSML(NX)
      JK1=EDGSML(NY)
      JKS1=0
      JMS1=0
      LK1=-1
      LM1=-1
      NS=JM1
      NT=JK1
C
   20 IF (NS .EQ. 0)GO TO 50
      IF (IDOIO(NS) .LT. 1)GO TO 40
      NS=RANKL(NS)
```

```
      GO TO 20
C
C**** SET THE EDGE NODE
C
   40 JMS1=NS
      M1=JMS1
      STAR1=RANKS(M1)
      ENDI1=RANKP(M1)
      DB2=RANKP(M1)
      DB1=RANKS(M1)
C
   50 IF (NT .EQ. 0)GO TO 80
      IF (IDOIO(NT) .LT. 1)GO TO 70
      NT=RANKL(NT)
      GO TO 50
C
C**** SET THE EDGE NODE TO BE LINKED
C
   70 JKS1=NT
      K1=JKS1
      STAR2=RANKP(K1)
      ENDI2=RANKS(K1)
      DA1=RANKP(K1)
      DA2=RANKS(K1)
C
   80 IF (JMS1 .EQ. 0  .AND. JKS1 .EQ. 0)GO TO 500
      LOPIN=LOPIN+1
   95 IF (JMS1 .NE. 0 .AND. JKS1 .NE. 0)THEN
C
C**** BOTH UP AND RIGHT EDGES ARE AVAILABLE FOR CONSTRUCTING LOOPS
C
           FEATURE(LOPIN)=JMS1
C
C**** FIND AN EDGE IN THE RIGHT-AXIS EDGE GROUP
C
  100      CALL NXLINK(LOPIN,JMS1,LK1,LM1,DA1,DA2,DB1,DB2,ISEAR,NX,
     1              RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2              RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
           NXY=1
  150      IF (ISEAR .EQ. 0)GO TO 260
C
C**** FIND AN EDGE IN THE UP-AXIS EDGE GROUP
C
  200      CALL NYLINK(LOPIN,JKS1,LM1,LK1,DA1,DA2,DB1,DB2,ISEAR,NY,
     1     RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),RANKFL(1),
     2     IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
           NXY=2
           IF (ISEAR .EQ. 0)GO TO 260
  250      IF (DB1 .EQ. STAR1 .AND. DB2 .EQ. ENDI1)GO TO 270
           GO TO 100
C
C****  FIND AN EDGE IN THE ARC EDGE GROUP
C
  260      CALL ARCLIN(EDGSML,EDGLAR,DA1,DA2,DB1,DB2,ISEAR,NXY,N,
     1              RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2              RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
           IF(ISEAR .EQ. 0)GO TO 550
           IF (NXY .EQ. 1) THEN
              RANKFL(LK1)=N
           ELSE
              RANKFL(LM1)=N
              DB1=DA1
              DB2=DA2
           ENDIF
C
           IDOIO(N)=LOPIN
           LK1=N
           CALL NXLINK(LOPIN,JMS1,LK1,LM1,DA1,DA2,DB1,DB2,ISEAR,NX,
     1              RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2              RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
           IF (ISEAR .GT. 0)GO TO 200
           LM1=N
```

```
                 DA1=DB1
                 DA2=DB2
                 CALL NYLINK(LOPIN,JKS1,LM1,LK1,DA1,DA2,DB1,DB2,ISEAR,NY,
     1                    RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2                    RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
                 IF (ISEAR .GT. 0)GO TO 250
                 GO TO 550
  270            NG=JMS1
                 RANKFL(LK1)=NG
                 NS=NG
            ELSE
             IF (JMS1 .EQ. 0) THEN
C
C**** ONLY THE UP-AXIS EDGE GROUP IS AVAILABLE
C
                 FEATURE(LOPIN)=JKS1
  300            CALL NYLINK(LOPIN,JKS1,LM1,LK1,DA1,DA2,DB1,DB2,ISEAR,NY,
     1                   RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2                   RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
                 IF (ISEAR .EQ. 0)GO TO 550
                 NXY=2
                 DA1=DB1
                 DA2=DB2
                 CALL ARCLIN(EDGSML,EDGLAR,DA1,DA2,DB1,DB2,ISEAR,NXY,N,
     1                  RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2                  RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
                 IF (ISEAR .EQ. 0)GO TO 550
                 RANKFL(LK1)=N
                 LM1=N
                 IDOIO(N)=LOPIN
                 DF1=DA1-STAR2
                 DF2=DA2-ENDI2
                 IF(DF2 .LE. 0.00001 .AND. DF1 .LE. 0.00001)GO TO 310
                 GO TO 300
  310            RANKFL(LM1)=JKS1
                 NG=JKS1
                 NS=JKS1
             ELSE
C
C****  ONLY THE UP-AXIS EDGE GROUP IS AVAILABLE
C
                 FEATURE(LOPIN)=JMS1
  400            CALL NXLINK(LOPIN,JMS1,LK1,LM1,DA1,DA2,DB1,DB2,ISEAR,NX,
     1                   RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2                   RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
                 IF (ISEAR .EQ. 0)GO TO 550
                 NXY=1
                 DB1=DA1
                 DB2=DA2
                 CALL ARCLIN(EDGSML,EDGLAR,DA1,DA2,DB1,DB2,ISEAR,NXY,N,
     1                   RANKN(1),RANKP(1),RANKS(1),RANKE(1),RANKL(1),
     2                   RANKFL(1),IDOIO(1),KONER(1),NFNAM(1),RANKR(1))
                 IF (ISEAR .EQ. 0)GO TO 550
                 RANKFL(LM1)=N
                 IDOIO(N)=LOPIN
                 LK1=N
                 DF1=DB1-STAR1
                 DF2=DB2-ENDI1
                 IF (DF1 .LE. 0.00001 .AND. DF2 .LE. 0.00001)GO TO 410
                 GO TO 400
  410            RANKFL(LK1)=JMS1
                 NG=JMS1
                 NS=JMS1
C
                 ENDIF
            ENDIF
  450       CONTINUE
C 450       WRITE(IW,*)NS,RANKN(NS),RANKP(NS),RANKS(NS),RANKE(NS),LOPIN
            NS=RANKFL(NS)
            IF(NS .NE. NG)GO TO 450
            GO TO 10
C
```

```
C**** CHECK IF ANY COMPLETE CIRCLE EDGE EXIST
C
  500 DO 510 I=4,6
      IF (EDGSML(I) .NE. 0)GO TO 520
  510 CONTINUE
  520 N=EDGSML(I)
  530 IF (N .EQ. 0)RETURN
C
C**** AN UNUSED ARC EDGE IS A COMPLETE CIRCLE EDGE
C**** WE HAVE ASSUMED NO INTERCLOSED CLINDRICAL FACES EXIST
C
      IF (IDOIO(N) .EQ. 0)GO TO 540
      N=RANKL(N)
      GO TO 530
  540 LOPIN=LOPIN+1
      FEATURE(LOPIN)=N
      IDOIO(N)=1
      RANKFL(N)=N
C
      N=RANKL(N)
      GO TO 530
  550 CONTINUE
C 550 WRITE(IW,560)
  560 FORMAT(1X,'***** EDGELINK ERROR *****')
      STOP
      END
C
      SUBROUTINE NXLINK(LOPIN,JM1,LK1,LM1,DA1,DA2,DB1,DB2,ISEAR,
     1NXY,RANKN,RANKP,RANKS,RANKE,RANKL,RANKFL,IDOIO,KONER,NFNAM,
     2RANKR)
C*********************************************************************
C     THIS SUBROUTINE IS USED TO FIND AN AVAILABLE EDGE IN THE       *
C     RIGHT-AXIS GROUP THAT LINKS TO THE LOOP.  EDGES ARE            *
C     EXAMINED FROM THE ONE WITH THE LOWEST AXIS OFFSET TO THE       *
C     ONE WITH THE HIGHEST AXIS OFFSET.                              *
C                                                                    *
C     LOPIN : THE LOOP NUMBER                                        *
C     JM1   : THE INPUT LOOP NODE                                    *
C     LM1   : LAST NODE LINKED IN A LOOP                             *
C     LK1   : NEW NODE TO BE LINKED IN ANOTHER EDGE GROUP            *
C     DA1,DA2 : THE POINTS FOUND IN THIS SUBROUTINE                  *
C     DB2,DB2 : THE POINTS TO BE LINKED                              *
C     ISEAR   : INDICATOR FOR A SEARCH IS SUCCESSFU(1) OR            *
C               NOT SUCCESSFUL(0)                                    *
C     NXY     : EDGE TYPE INDICATOR                                  *
C*********************************************************************
      DIMENSION RANKN(99),RANKS(99),RANKR(99),RANKP(99),RANKL(99)
      DIMENSION RANKE(99),RANKFL(99),KONER(4),IDOIO(99),NFNAM(99)
      INTEGER RANKN,RANKR,RANKL,EDGSML,EDGLAR,RANKFL
      COMMON /INOUT/IW,NFTYP,NETYP
      COMMON /LLINK/AMAX1,AMAX2
      M1=JM1
 1120 MM1=M1
      LM1=M1
      DA2=RANKP(LM1)
      DF=ABS(DA2-DB2)
C
C**** COMPARING THE LOCATION OF THE POINT
C
      IF (DF .LE. 0.00001)GO TO 1130
      M1=RANKL(LM1)
      IF(M1 .EQ. 0) THEN
          ISEAR=0
          RETURN
      ENDIF
      GO TO 1120
 1130 LM1=M1
      M1=RANKL(LM1)
      IF(M1 .GT. 0)GO TO 1140
C
C**** CHECK IF THE EDGE HAS THE HIGHEST AXIS OFFSET
C
```

```
      IF (RANKP(LM1) .NE.. AMAX1) THEN
          ISEAR=0
          RETURN
      ENDIF
      GO TO 1150
 1140 TDA2=RANKP(M1)
      IF(TDA2 .NE. DA2)GO TO 1150
C
C**** CHECK IF A CONTINUOUS EDGE EXIST
C
      IF(RANKE(LM1) .NE. RANKS(M1))GO TO 1150
      RANKFL(LM1)=M1
      GO TO 1130
 1150 DA1=RANKS(MM1)
      DAA=RANKE(LM1)
C
C**** COMPARING THE LOCATION OF A POINT
C
      DF=ABS(DA1-DB1)
      IF (DF .LE. 0.00001)GO TO 1160
      DA1=RANKE(LM1)
      DAA=RANKS(MM1)
      DF=ABS(DA1-DB1)
      IF(DF .LE. 0.00001)GO TO 1160
      M1=RANKL(LM1)
      IF( M1 .GT. 0)GO TO 1120
      ISEAR=0
      RETURN
C
C**** SET THE POINT TO LINKED
C
 1160 DA1=DAA
 1180 IF(LK1 .GT. 0)RANKFL(LK1)=MM1
      I=MM1
 1184 IF(I .EQ. LM1)GO TO 1185
C
C**** MARK THE EDGE AS AN USED EDGE
C
      IDOIO(I)=LOPIN
      I=RANKFL(I)
      GO TO 1184
 1185 IDOIO(I)=LOPIN
C     WRITE(IW,1190)DA1,DA2,RANKN(I),NFNAM(I)
 1190 FORMAT(1X,'*****',2F9.4,2I4)
      ISEAR=1
      RETURN
      END
C
      SUBROUTINE NYLINK(LOPIN,JK1,LM1,LK1,DA1,DA2,DB1,DB2,ISEAR,
     1NXY,RANKN,RANKP,RANKS,RANKE,RANKL,RANKFL,IDOIO,KONER,NFNAM,
     2RANKR)
C*****************************************************************
C     THIS SUBROUTINE IS USED TO FIND AN AVAILABLE EDGE IN THE   *
C     UP-AXIS GROUP THAT LINKS TO THE LOOP.  EDGES ARE           *
C     EXAMINED FROM THE ONE WITH THE LOWEST AXIS OFFSET TO THE   *
C     ONE WITH THE HIGHEST AXIS OFFSET.                          *
C     LOPIN : THE LOOP NUMBER                                    *
C     JK1   : THE INPUT LOOP NODE                                *
C     LK1   : LAST NODE LINKED IN A LOOP                         *
C     LM1   : NEW NODE TO BE LINKED IN ANOTHER EDGE GROUP        *
C     DB1,DB2 : THE POINTS FOUND IN THIS SUBROUTINE              *
C     DA2,DA2 : THE POINTS TO BE LINKED                          *
C     ISEAR   : INDICATOR FOR A SEARCH IS SUCCESSFU(1) OR        *
C               NOT SUCCESSFUL(0)                                *
C     NXY     : EDGE TYPE INDICATOR                              *
C*****************************************************************
      DIMENSION RANKN(99),RANKS(99),RANKR(99),RANKP(99),RANKL(99)
      DIMENSION RANKE(99),RANKFL(99),KONER(4),IDOIO(99),NFNAM(99)
      INTEGER RANKN,RANKR,RANKL,EDGSML,EDGLAR,RANKFL
      COMMON /INOUT/IW,NFTYP,NETYP
      COMMON /LLINK/AMAX1,AMAX2
 1210 K1=JK1
```

```
 1220 MK1=K1
      LK1=K1
      DB1=RANKP(LK1)
C
C**** COMPARING THE LOCATION OF THE POINT
C
      DF=ABS(DA1-DB1)
      IF(DF .LE. 0.00001)GO TO 1300
      K1=RANKL(LK1)
      IF(K1 .EQ. 0) THEN
          ISEAR=0
          RETURN
      ENDIF
      GO TO 1220
 1300 LK1=K1
      K1=RANKL(LK1)
      IF (K1 .GT. 0)GO TO 1320
C
C**** CHECK IF THE EDGE HAS THE HIGHESE AXIS OFFSET
C
      IF (RANKP(LK1) .NE. AMAX2) THEN
          ISEAR=0
          RETURN
      ENDIF
      GO TO 1330
 1320 TDB1=RANKP(K1)
C
C**** CHECK IF A CONTINUOUS EDGE EXIST
C
      IF(TDB1 .NE. DB1)GO TO 1330
          IF (RANKE(LK1) .NE. RANKS(K1))GO TO 1330
          RANKFL(LK1)=K1
          GO TO 1300
 1330 DB2=RANKS(MK1)
      DBB=RANKE(LK1)
C
C**** COMPARE THE LOCATION OF THE POINT
C
      DF=ABS(DB2-DA2)
      IF(DF .LE. 0.00001)GO TO 1400
      DB2=RANKE(LK1)
      DBB=RANKS(MK1)
      DF=ABS(DB2-DA2)
      IF(DF .LE. 0.00001)GO TO 1400
      K1=RANKL(LK1)
      IF (K1 .GT. 0)GO TO 1220
      ISEAR=0
      RETURN
C
C**** SET THE POINT BE THE NEW POINT TO BE LINKED
C
 1400 DB2=DBB
 1460 IF(LM1 .GT. 0)RANKFL(LM1)=MK1
 1465 FORMAT(1X,'*****',2F9.4,2I4)
      I=MK1
 1470 IF(I .EQ. LK1)GO TO 1480
C
C**** MARK THE EDGE AS AN USED EDGE
C
      IDOIO(I)=LOPIN
      I=RANKFL(I)
      GO TO 1470
 1480 IDOIO(I)=LOPIN
C     WRITE(IW,1465)DB1,DB2,RANKN(I),NFNAM(I)
C
C
      ISEAR=1
C
      RETURN
      END
C
      SUBROUTINE ARCLIN(EDGSML,EDGLAR,DA1,DA2,DB1,DB2,ISEAR,NXY,N,
```

```
     1RANKN,RANKP,RANKS,RANKE,RANKL,RANKFL,IDOIO,KONER,NFNAM,
     2RANKR)
C*********************************************************************
C      THIS SUBROUTINE IS USED TO FIND AN ARC EDGE THAT LINKS        *
C      TO THE CURRENT LOOP.                                          *
C      THE VARIABLES ARE DEFINED IN THE SUBROUTINE EDGELINK.         *
C*********************************************************************
      DIMENSION RANKN(99),RANKS(99),RANKR(99),RANKP(99),RANKL(99)
      DIMENSION RANKE(99),RANKFL(99),KONER(4),IDOIO(99),NFNAM(99)
      INTEGER RANKN,RANKR,RANKL,EDGSML(9),EDGLAR(9),RANKFL
      COMMON /INOUT/IW,NFTYP,NETYP
      LOGICAL LIF1,TRUE
C
      ISEAR=0
      DO 100 I=4,6
      N=EDGSML(I)
      IF (N .NE. 0)GO TO 120
  100 CONTINUE
C
  110 IF(N .EQ. 0)RETURN
C
C**** FIND AN AVAILABLE EDGE
C
  120 IF (IDOIO(N) .EQ. 0)GO TO 130
      N=RANKL(N)
      GO TO 110
  130 A=RANKS(N)
      B=RANKP(N)
      R=RANKE(N)
      IREC=RANKN(N)
C
C**** READ IN THE EDGE INFORMATION
C
      READ(10,REC=IREC,FMT=81)NAM,IYP,INAM,LIM1,LIM2,LIM3,IF1,IF2,
     1ILIN,I1,D1,I2,D2,I3,D3,I4,D4,I5,D5,I6,D6
      IF(IYP .GT. 2500)READ(14,REC=ILIN,FMT=97)I7,E1,I2,E2,I3,E3,
     1I4,E4,I5,E5,I6,E6
   81 FORMAT(I3,I6,I3,3I2,2I3,I6,6(I2,F9.5))
   97 FORMAT(6(I2,F9.5))
C
C**** SET UP THE MAXIMUM AND MINUM ANGLES
C
      IF (I .EQ. 4) THEN
         THDA1=D4
         THDA2=D5
      ELSE
         IF (I .EQ. 5) THEN
            THDA1=D5
            THDA2=D6
         ELSE
            THDA1=D6
            THDA2=D5
         ENDIF
      END IF
C
C**** CALCULATE THE STARTING POINT AND THE ENDING POINT
C
      DC1=A+R*COSD(THDA1)
      DC2=B-R*SIND(THDA1)
      DC3=A+R*COSD(THDA2)
      DC4=B-R*SIND(THDA2)
      IF (I .EQ. 4)THEN
      D=DC1
          DC1=DC2
          DC2=D
      D=DC3
          DC3=DC4
          DC4=D
      ENDIF
C
C     WRITE(IW,150)DC1,DC2,DC3,DC4,RANKN(N),NFNAM(N)
      DF1=ABS(DC1-DB1)
```

```
      DF2=ABS(DC2-DB2)
      DF3=ABS(DC3-DB1)
      DF4=ABS(DC4-DB2)
C
      DF5=ABS(DC1-DA1)
      DF6=ABS(DC2-DA2)
      DF7=ABS(DC3-DA1)
      DF8=ABS(DC4-DA2)
  150 FORMAT(1X,4F9.4,2I4)
      IF(NXY .EQ. 1) THEN
C
C**** THE ARC EDGE IS LINKED TO THE UP-AXIS TYPE EDGE
C
         IF(DF1 .LE. 0.0001 .AND. DF2 .LE. 0.0001) THEN
            DB1=DC3
            DB2=DC4
         ELSE
C
C**** THE ARC IS LINKED TO THE RIGHT-AXIS TYPE EDGE
            IF(DF3 .LE. 0.0001 .AND. DF4 .LE. 0.0001) THEN
               DB1=DC1
               DB2=DC2
            ELSE
               N=RANKL(N)
               GO TO 110
            ENDIF
         ENDIF
      ELSE
         IF(DF5 .LE. 0.0001 .AND. DF6 .LE. 0.0001) THEN
            DA1=DC3
            DA2=DC4
         ELSE
            IF(DF7 .LE. 0.0001 .AND. DF8 .LE. 0.0001) THEN
               DA1=DC1
               DA2=DC2
            ELSE
               N=RANKL(N)
               GO TO 110
            ENDIF
         ENDIF
      ENDIF
C
      ISEAR=1
      RETURN
      END
C
C
C****************************************************************
C**** THIS SUBROUTINE IS USED TO DECIDE THE RELATION BETWEEN *
C**** TWO FACE NORMAL DIRECTIONS                              *
C****************************************************************
C
      SUBROUTINE DIRECT(DA1,DA2,NTYP1,NTYP2,NDR1,NDR2)
C
C****************************************************************
C**   DA1 AND DA2 ARE AXIS OFFSETS OF TWO FACES
C**   NTYP1 AND NTYP2 ARE FACE TYPES
C**   NDR1 AND NDR2 ARE FLAG INDICATE THE RELATION BETWEEN
C**                 SURFACE NORMAL AND AXIS OFFSET
C**                 1 : SURFACE NORMALS POINT TO THE POSITIVE
C**                     DIRECTION AND THE SURFACE HAS A HIGH
C**                     AXIS OFFSET; OR SURFACE NORMALS POINT
C**                     TO THE NEGATIVE DIRECTION AND THE
C**                     SURFACE HAS A LOW AXIS OFFSET.
C**                -1 : OTHERWISE
C****************************************************************
      N=(NTYP1+3)/100
      NN=NTYP1-N*100
      IF (NN .GT. 0) THEN
          IF (DA1 .LT. DA2) THEN
              NDR1=-1
          ELSE
```

```
                NDR1=1
            ENDIF
        ELSE
            IF (DA1 .LT. DA2) THEN
                NDR1=1
            ELSE
                NDR1=-1
            ENDIF
        ENDIF
C
        M=(NTYP2+3)/100
        MM=NTYP2-M*100
        IF (MM .GT. 0) THEN
            IF (DA1 .LT. DA2) THEN
                NDR2=1
            ELSE
                NDR2=-1
            ENDIF
        ELSE
            IF (DA1 .LT. DA2) THEN
                NDR2=-1
            ELSE
                NDR2=1
            ENDIF
        ENDIF
C
        RETURN
        END
C
C*******************************************************************
C       INEREL IS A ROUTINE TO IDENTIFY WHETHER A LOOP IS AN       *
C       INNER LOOP OR AN EXTERNAL LOOP                             *
C*******************************************************************
        SUBROUTINE INEREL(KL,IWR,INERF,NUMEXT,NMIX,FEATURE,RANKN,
     1                  RANKP,RANKS,RANKE,RANKFL)
C*******************************************************************
C       KL : THE ACTIVE SURFACE
C       IWR : A FLAG WHICH INDICATES THIS SUBROUTINE IS CALLED
C             FROM FCEEDG(0) OR FROM BLOCAG(1).
C       INERF : NUMBER OF LOOPS
C       NUMEXT : NUMBER OF EXTERNAL LOOPS
C       NMIX   : LOOP NUMBERS; POSITIVE NUMBERS STAND FOR EXTERNAL
C*******************************************************************
        INTEGER RANKN(99),FEATURE(20),RANKFL(99)
        DIMENSION RANKP(99),RANKS(99),RANKE(99),AB(4),BB(4)
        DIMENSION NFTYP(99),NETYP(99),NMIX(20),AMIX(10,4)
        COMMON /INOUT/IC,NFTYP,NETYP
        DATA I4/4/,IW/6/,IR/5/
C
        NK=1
        DO 110 I=1,20
    110 NMIX(I)=0
        N=FEATURE(1)
C
C**** FINDING THE EDGE TYPES.
C**** NA, NB: 1 X-TYPE EDGE, 2 Y-TYPE EDGE, 3 Z-TYPE EDGE
C
        NF=ABS(NFTYP(KL)-100)
        IF (NF .EQ. 1) THEN
           NA=3
           NB=2
        ELSE IF (NF .EQ. 2) THEN
           NA=1
           NB=3
        ELSE IF (NF .EQ. 3) THEN
           NA=1
           NB=2
        ELSE
        ENDIF
C
C**** AB(1) THE LOWEST EDGE OF THE NA-TYPE EDGE
C**** AB(2) THE HIGHEST EDGE OF THE NA-TYPE EDGE
```

```
C****  AB(3)  THE LOWEST EDGE OF THE NB-TYPE EDGE
C****  AB(4)  THE HIGHEST EDGE OF THE NB-TYPE EDGE
C
       CALL MINMAX(N,NA,NB,AB(1),RANKN(1),RANKP(1),RANKFL(1))
       II=1
C
C**    AMIX(I,J) STORES ITH OUTER LOOP INFORMATION
C
       NMIX(II)=II
       AMIX(1,1)=AB(1)
       AMIX(1,2)=AB(2)
       AMIX(1,3)=AB(3)
       AMIX(1,4)=AB(4)
       NUMEXT=1
C
C****  DFN AND PFN STORES THE DISTANCE BETWEEN TWO EDGES
C****  IINN INDICATES IF A LOOP IS AN INNER LOOP (1) OR
C****       AN OUTER LOOP (0).
C
       DO 500 II=2,INERF
       DFN=1000
       PFN=1000
       IINN=0
       N=FEATURE(II)
       N1=N
       N2=N
   150 NAM=RANKN(N2)
       IF (NETYP(NAM) .GT. 1003)GO TO 350
       N2=RANKFL(N2)
       IF(N2 .NE. N1)GO TO 150
       CALL MINMAX(N,NA,NB,BB(1),RANKN(1),RANKP(1),RANKFL(1))
       DO 300 I=1,NUMEXT
       DO 200 J=1,4
   200 AB(J)=AMIX(I,J)
C
C****  CHECK IF THE LOOP IS AN INNER LOOP OR OUTER LOOP
C
       CALL INOUTF(KL,NAM,INAUT,AB(1),BB(1))
       IF (INAUT .EQ. 0)GO TO 300
C
C****  FIND THE OUTER LOOP TO WHICH THE INNER LOOP BELONGS.
C****  THE DECISION IS BASED ON THE DISTANCE BETWEEN THE
C****  THE TWO LOWEST EDGES OF THE INNER LOOP AND THE OUTER
C****  LOOPS.
C
       DN=ABS(AB(1)-BB(1))
       PN=ABS(AB(3)-BB(3))
       IF (DN .LT. DFN .OR. FN .LT. PFN) THEN
           DFN=DN
           PFN=PN
           NM=NMIX(I)
           IINN=1
       ENDIF
   300 CONTINUE
       IF (IINN .EQ. 1)NMIX(II)=-NM
C
       IF (IINN .EQ. 1)GO TO 350
       NUMEXT=NUMEXT+1
       NMIX(II)=II
       AMIX(NUMEXT,1)=BB(1)
       AMIX(NUMEXT,2)=BB(2)
       AMIX(NUMEXT,3)=BB(3)
       AMIX(NUMEXT,4)=BB(4)
       GO TO 500
C
   350 IF (IWR .EQ. 1)GO TO 500
       IF (NETYP(NAM) .LE. 1003)GO TO 400
C
C****  FIND THE OUTER LOOP TO WHICH A CIRRCLE LOOP BELONGS
C
       N=N2
       DO 380 J=1,NUMEXT
```

```
             DN=ABS(AMIX(J,3)-RANKS(N))
             PN=ABS(AMIX(J,1)-RANKP(N))
             IF (DN .LT. DFN .OR. PN .LT. PFN) THEN
                DFN=DN
                PFN=PN
                NM=NMIX(J)
                NMIX(II)=-NM
             ENDIF
   380 CONTINUE
C
   400 CONTINUE
C   400 WRITE(6,420)II,KL,NM
C   420 FORMAT(1X,'LOOP',I3,' IS AN INTERNAL LOOP OF',I3,
C      1' ON PORTION',I3)
   500 CONTINUE
C
C
       RETURN
       END
C
C*******************************************************************
C      MINMAX IS THE SUBROUTINE TO FIND THE LOWEST AXIS OFFSET      *
C      AND THE HIGHEST EDGE AXIS OFFFSET OF THE NA-TYPE AND         *
C      NB-TYPE EDGES ON A GIVEN LOOP.                               *
C*******************************************************************
C
       SUBROUTINE MINMAX(N,NA,NB,AB,RANKN,RANKP,RANKFL)
C*******************************************************************
C**** N : THE FIRST NODE OF THE GIVEN LOOP
C**** NA : THE RIGHT-AXIS EDGE GROUP CODE
C**** NB : THE UP-AXIS EDGE GROUP CODE
C**** AB(1) THE LOWEST EDGE POSTION OF NA-TYPE EDGE
C**** AB(2) THE HIGHEST EDGE POSTION OF NA-TYPE EDGE
C**** AB(3) THE LOWEST EDGE POSTION OF NB-TYPE EDGE
C**** AB(4) THE HIGHEST EDGE POSTION OF NB-TYPE EDGE
C*******************************************************************
       DIMENSION AB(4),RANKP(99)
       INTEGER RANKN(99),RANKFL(99),NFTYP(99),NETYP(99)
       COMMON /INOUT/IW,NFTYP,NETYP
C
       MF=N
       AB(1)=-1
       AB(3)=-1
C
   100 NAM=RANKN(N)
       IF (NETYP(NAM) .GT. 1004)GO TO 200
       ND=NETYP(NAM)-1000
       IF (ND .EQ. NA) THEN
C
C**** NA-TYPE EDGES
C
       IF (AB(1) .LT. 0) THEN
          AB(1)=RANKP(N)
          AB(2)=AB(1)
          GO TO 200
       ENDIF
C
          APOS=RANKP(N)
          IF (AB(1) .GT. APOS) THEN
              AB(1)=APOS
          ELSE IF (AB(2) .LT. APOS) THEN
              AB(2)=APOS
          ELSE
          ENDIF
C
C**** THE NB-TYPE EDGES
C
       ELSE
          IF (AB(3) .LT. 0) THEN
             AB(3)=RANKP(N)
             AB(4)=AB(3)
             GO TO 200
```

```
            ENDIF
            BPOS=RANKP(N)
            IF (AB(3) .GT. BPOS) THEN
               AB(3)=BPOS
            ELSE IF (AB(4) .LT. BPOS) THEN
               AB(4)=BPOS
            ELSE
            ENDIF
         ENDIF
  200    N=RANKFL(N)
         IF (N .NE. MF)GO TO 100
         RETURN
         END
C
C***************************************************************
C        INOUTF IDENTIFY THE INNER AND OUTER LOOP FROM A GIVEN *
C        OUTER LOOP .                                          *
C***************************************************************
         SUBROUTINE INOUTF(KL,NAM,INAUT,AB,BB)
C****************************************************************
C**** KL IS THE ACTIVE FACE NAME
C**** NAM IS THE EDGE NAME OF LOWEST EDGE OF THE LOOP TO BE
C****     IDENTIFIED
C**** INAUT=1 INDICATE INTERNAL
C**** INAUT=0 INDICATE EXTERNAL
C**** AB STORES THE HIGHEST AND THE LOWEST INFORMATION OF
C****    THE OUTER LOOP
C**** BB STORES THE HIGHEST AND THE LOWEST INFORMATION OF
C****    THE LOOP TO BE IDENTIFIED
C****************************************************************
         DIMENSION AB(4),BB(4),NEOG(12),DATA(6)
         DIMENSION NETYP(99),NFTYP(99),XPOS(99)
         COMMON /INOUT/IW,NFTYP,NETYP,XPOS
C
C**** IDENTIFY THE OUTER LOOP
C
         IF (BB(2) .LT. AB(1) .OR. BB(1) .GT. AB(2))GO TO 300
         IF (BB(4) .LT. AB(3) .OR. BB(3) .GT. AB(4))GO TO 300
         IF (BB(1) .LE. AB(1) .OR. BB(3) .LE. AB(3))GO TO 300
         IF (BB(2) .GE. AB(2) .OR. BB(4) .GE. AB(4))GO TO 300
         NYP=NETYP(NAM)
         CALL FACEN(KL,KK,NAM,DD)
         IF (NFTYP(KK) .GT. 100)GO TO 200
         READ(9,REC=KK,FMT=50) (NEOG(I),I=1,12)
   50    FORMAT(55X,12I3)
         DO 100 I=1,12
         NG=NEOG(I)
         IF (NETYP(NG) .NE. NYP)GO TO 150
  100    CONTINUE
         WRITE(6,*)'ERROR IN INOUTF SUBROUTINE'
         STOP
  150    READ (10,REC=NG,FMT=155)(DATA(J),J=1,6)
  155    FORMAT(30X,6(2X,F9.5))
         DO 160 K=1,3
         IF (DATA(K) .NE. DATA(K+3))GO TO 180
  160    CONTINUE
C
C**** CHECK IF THE SHARED SURFACE IS ABOVE THE ACTIVE FACE
C
  180    SP=DATA(K)
         EP=DATA(K+3)
         IF (NFTYP(KL) .GT. 100) THEN
C
C**** ACTIVE SURFACE NORMAL POINTS TO THE POSITIVE DIRECTION
C
            IF (EP .LE. XPOS(KL))GO TO 300
C**** THE ENDING POINT OF AN EDGE OF THE SHARED SURFACE
C**** IS HIGHER THAN THE ACTIVE SURFACE POSITION.
         ELSE
C
C**** ACTIVE SURFACE NORMAL POINTS TO THE NEGATIVE DIRECTION
C
```

```
           IF (SP .GE. XPOS(KL))GO TO 300
C**** THE ENDING POINT OF AN EDGE OF THE SHARED SURFACE
C**** IS LOWER THAN THE ACTIVE SURFACE POSITION.
      ENDIF
  200 INAUT=1
      RETURN
  300 INAUT=0
      RETURN
      END
      SUBROUTINE FACESORT(NX,NY,NZ,KK,NSORT,EEDGE)
C
C*****************************************************************
C          THIS SUBROUTINE IS USED TO ORDER DIFFERENT TYPES OF      *
C     SURFACES AND STORES THE SURFACES INTO DIFFERENT GROUPS        *
C     NX, NY, NZ : NUMBER OF X-TYPE, Y-TYPE, AND Z-TYPE PLANE       *
C                  SURFACES.                                        *
C     KK : TOTAL NUMBER OF SURFACES                                 *
C     NSORT : SURFACES ORDERED ACCORDING TO THE NUMBER OF EGES.     *
C     EEDGE : NUMBER OF EDGES OF A SURFACE                          *
C     XFACE, YFACE, ZFACE : ORDERED X-TYPE, Y-TYPE, AND Z-TYPE      *
C                           PLANE AND DISC SURFACES.                *
C     XCYL, YCYL, ZCYL : ORDERED X-TYPE, Y-TYPE, AND Z-TYPE         *
C                        CYLINDERS                                  *
C*****************************************************************
C
      DIMENSION NFACE(99),EEDGE(99),NSORT(99),DATA(6),NOEG(99)
      DIMENSION NFTYP(99),XPOS(99),CPOS(99),NETYP(99)
      INTEGER XFACE(35),ZFACE(35),YFACE(35)
      INTEGER XCYL(35),YCYL(35),ZCYL(35)
      COMMON /INOUT/IW,NFTYP,NETYP,XPOS,CPOS
      COMMON /INFOF/XFACE,YFACE,ZFACE,XCYL,YCYL,ZCYL
      LOGICAL LIF1
      DATA I1/1/
      DO 10 I=1,99
      EEDGE(I)=0
   10 NSORT(I)=0
  250 KK=0
      CALL READ20(NF,I1)
  100 FORMAT(6I4)
      DO 350 II=1,NF
C
C*****EEDGE LESS THAN 0 INDICATES THAT THIS FACE HAS BEEN LINKED
C*****TO ANOTHER FACE
C
      IF(EEDGE(II) .LT. 0)GO TO 350
      NRC=II
      CALL READF(IFC,NFYP,NEDG,IP,INM,LI1,DATA,LIN,ILIN,NOEG,ISO)
      EEDGE(II)=NEDG
C
C      CLASSIFYING SURFACES BY FACE TYPE CODE
C
      IF (NFTYP(II) .LT. 0)GO TO 350
      IND=(NFTYP(II)+3)/100
      IC=ABS(NFTYP(II)-IND*100)
      IF(NFTYP(II) .GT. 296)IC=IC+3
  300 GO TO (301,302,303,320,322,324),IC
  301 NX=NX+1
      XFACE(NX)=IFC
      GO TO 305
  302 NY=NY+1
      YFACE(NY)=IFC
      GO TO 305
  303 NZ=NZ+1
      ZFACE(NZ)=IFC
  305 XPOS(IFC)=DATA(1)
      GO TO 330
  320 NCX=NCX+1
      XCYL(NCX)=IFC
      GO TO 326
  322 NCY=NCY+1
      YCYL(NCY)=IFC
      GO TO 326
```

```
  324 NCZ=NCZ+1
      ZCYL(NCZ)=IFC
  326 XPOS(IFC)=DATA1
      CPOS(IFC)=DATA2
  330 CONTINUE
      KK=KK+1
      NSORT(KK)=II
  350 CONTINUE
      IO=1
      ISL=1
      CALL BUBLE(IO,KK,EEDGE(1),CPOS(1),NSORT(1),ISL)
C
C     SORT X-FACE AND X-CYLINDER
C
      CALL BUBLE(IO,NX,XPOS(1),CPOS(1),XFACE(1),ISL)
      CALL BUBLE(IO,NCX,XPOS(1),CPOS(1),XCYL(1),ISL)
C
C     SORT Y-FACE AND Y-CYLINDER
C
      CALL BUBLE(IO,NY,XPOS(1),CPOS(1),YFACE(1),ISL)
      CALL BUBLE(IO,NCY,XPOS(1),CPOS(1),YCYL(1),ISL)
C
C     SORT Z-FACE AND Z-CYLINDER
C
      CALL BUBLE(IO,NZ,XPOS(1),CPOS(1),ZFACE(1),ISL)
      CALL BUBLE(IO,NCZ,XPOS(1),CPOS(1),ZCYL(1),ISL)
      RETURN
      END
C
      SUBROUTINE BUBLE(IO,II,DATA,DASA,ORDER,ISL)
C
C*****************************************************************
C       THIS SUBROUTINE IS BASED ON BUBBLE SORT ALGORITHM. IT   *
C       SORTS DATA FROM SMALL TO LARGE OR VICE VERSA.           *
C*****************************************************************
C**** IO IS THE STARTING NUMBER
C**** II IS THE ENDING NUMBER
C**** DATA IS THE VALUE TO BE COMPARED
C**** ORDER IS THE RANK SEQUENCE
C**** ISL=1 OREDER FROM SMALL TO LARGE
C****     2 ORDER FROM LARGE TO SMALL
C*****************************************************************
C
      DIMENSION DATA(99),DASA(99)
      INTEGER ORDER(99)
      DO 400 I=IO,II
      N=ORDER(I)
      SMALA=DATA(N)
      SMALL=DASA(N)
      DO 400 J=I+1,II
          M=ORDER(J)
          IF (ISL.EQ.2)GO TO 350
              IF(DATA(M) .LT. SMALA)GO TO 380
              IF(DATA(M) .GT. SMALA)GO TO 400
              IF(DASA(M) .EQ. SMALL)GO TO 380
              GO TO 400
  350         IF (DATA(M) .LE. SMALA)GO TO 400
              IF (DATA(M) .LT. SMALA)GO TO 400
              IF (DASA(M) .LE. SMALL)GO TO 400
  380         SMALA=DATA(M)
              ORDER(I)=M
              ORDER(J)=N
              N=M
  400 CONTINUE
C     WRITE(IW,*)(ORDER(I),I=IO,II),IO,II
      RETURN
      END
C*****************************************************************
C***** THIS SUBROUTINE ASKS USER TO INPUT THE ROUGHNESS OF A     *
C***** SURFAVCE AND THE TOLERANCES OF THE HOLE DIAMETER          *
C*****************************************************************
C
```

```
      SUBROUTINE ROUFINPT(NNX,NNY,NNZ,XPOS)
      INTEGER PC,XX,YY,ZZ,XYZ,ROUF
      DIMENSION XFACE(35),YFACE(35),ZFACE(35),XPOS(99)
      DIMENSION XCYL(35),YCYL(35),ZCYL(35)
      COMMON/INFOF/XFACE,YFACE,ZFACE,XCYL,YCYL,ZCYL
C
      DATA IP/'P'/,IC/'C'/,XX/'X'/,YY/'Y'/,ZZ/'Z'/
C
      WRITE(6,*)' '
      WRITE(6,*)'      SURFACE FINISH ATTRIBUTES AND TOLERANCES'
      WRITE(6,*)'                          ENTERING PROCEDURE            '
C
      WRITE(6,*)' '
      WRITE(6,*)'IS THERE ANY SURFACE THAT NEEDS TO BE UPDATED?'
      READ (5,100)IYE
      IF(IYE .NE. YY)RETURN
      WRITE(6,*)' '
 10   WRITE(6,*)'IS THIS A PLANE SURFACE(P) OR'
      WRITE(6,*)'A CYLINDRIC SURFACE(C)?'
      READ(5,100)PC
 100  FORMAT(A1)
      IF (PC .NE. IP .AND.  PC .NE. IC)GO TO 10
      WRITE(6,*)'ENTER THE SURFACE TYPE(X,Y OR Z)'
      READ (5,100)XYZ
C
      WRITE(6,*)'ENTER THE REQUIRED SURFACE FINISH'
      WRITE(6,*)'IN MICRO-INCHES:'
      READ (5,*)ROUF
      IF (PC .EQ. IC)GO TO 500
C
      WRITE(6,*)'ENTER THE POSITION OF A CORNER POINT'
      WRITE(6,*)'OF THE SURFACE:'
      READ(5,*)X,Y,Z
C
      IF (XYZ .EQ. ZZ)CALL ROUFP(NNZ,ZFACE(1),XPOS(1),Z,ROUF)
      IF (XYZ .EQ. YY)CALL ROUFP(NNY,YFACE(1),XPOS(1),Y,ROUF)
      IF (XYZ .EQ. XX)CALL ROUFP(NNX,XFACE(1),XPOS(1),X,ROUF)
      GO TO 800
C
 500  CONTINUE
C
      WRITE(6,*)'ENTER THE DIAMETER OF THE HOLE, (IN INCHES)'
      WRITE(6,*)'THE MAXIMUM TOLERANCE OF THE HOLE (IN 0.001") '
      WRITE(6,*)'THE MINIMUN TOLERANCE OF THE HOLE (IN 0.001"):'
      READ (5,*)DIAM,XTOL,STOL
      IF (XYZ .EQ. ZZ)GO TO 700
      IF (XYZ .EQ. YY)GO TO 600
C
      WRITE(6,610)
 510  FORMAT(1X,'ENTER THE "X" POSITION OF THE AXIS:')
      READ(5,*)P1
      WRITE(6,710)
      READ(5,*)P2
      CALL ROUFR(XCYL,ROUF,XTOL,STOL,DIAM,P1,P2)
      GO TO 800
 600  WRITE(6,510)
 610  FORMAT(1X,'ENTER THE "Y" POSITION OF THE AXIS:')
      READ(5,*)P1
      WRITE(6,710)
      READ(5,*)P2
      CALL ROUFR(YCYL,ROUF,XTOL,STOL,DIAM,P1,P2)
      GO TO 800
 700  WRITE(6,510)
 710  FORMAT(1X,'ENTER THE "Z" POSITION OF THE AXIS:')
      READ(5,*)P1
      WRITE(6,610)
      READ(5,*)P2
      CALL ROUFR(ZCYL,ROUF,XTOL,STOL,DIAM,P1,P2)
C
 800  WRITE(6,*)'IS THERE ANY OTHER SURFACE TO BE UPDATED?'
      READ (5,100)IYE
      IF(IYE .EQ. YY)GO TO 10
```

```
      RETURN
      END
C
C***********************************************************
C     THIS SUBROUTINE UPDATE THE ROUGHNESS INFORMATION     *
C***********************************************************
      SUBROUTINE ROUFP(NNN,MXYZ,XPOS,POS,ROUF)
C
C**   NNN IS THE FACE -TYPE INDICATOR
C
      DIMENSION MXYZ(35),XPOS(99)
C
      XTOL=0
      STOL=0
C
      DO 100 I=1,NNN
          N=MXYZ(I)
          IF (XPOS(N) .LT. POS)GO TO 100
          IF (XPOS(N) .GT. POS)RETURN
          CALL WRITROUF(N,ROUF,XTOL,STOL)
  100 CONTINUE
      RETURN
      END
C
C***********************************************************
C     THIS SUBROUTINE UPDATE THE TOLERANCE INFORMATION     *
C***********************************************************
      SUBROUTINE ROUFR(MXYZ,ROUF,XTOL,STOL,DIAM,P1,P2)
C
      DIMENSION MXYZ(35),DATA(6),NOEG(12)
      LOGICAL LIF1
C
      N=0
      XTOL=XTOL*0.001
      STOL=STOL*0.001
  100 N=N+1
      K=MXYZ(N)
      IF (MXYZ(N) .EQ. 0)RETURN
      CALL READF(K,ITY,JJ,IP,INM,LIF1,DATA(1),NFC,IN,NOEG(1),ISO)
C     WRITE(6,*) (DATA(I),I=1,3),P1,P2,DIAM
      DD=DATA(3)*2.0
      IF (DATA(1).EQ.P1 .AND. DATA(2).EQ.P2 .AND. DD.EQ.DIAM) THEN
      CALL WRITROUF(K,ROUF,XTOL,STOL)
      RETURN
      END IF
      GO TO 100
      END
C*****************************************************************
C    THIS SUBROUTINE IS USED TO RECOGNIZE THE BORE TYPE AND THE *
C    HOLE TYPE FEATURES.  NX, NY, AND NZ ARE THE NUMBER OF      *
C    PLANE AND DISK FACES OF X-TYPE, Y-TYPE, AND Z-TYPE         *
C    RESPECTIVELY.                                              *
C*****************************************************************
     SUBROUTINE HOLBOR(NX,NY,NZ)
     INTEGER XCYL(35),YCYL(35),ZCYL(35)
     INTEGER XFACE(35),YFACE(35),ZFACE(35)
     DIMENSION XPOS(99),CPOS(99),NOEG(35),NCX(99)
     DIMENSION NFTYP(99),NETYP(99)
     DIMENSION MAINF(99),MSFRL(99),NFEAT(20),NODE(300),LINK(300)
     DIMENSION EXAMF(99),NSR(20),NMANF(20)
     INTEGER BOR1,BOR2,BOR3,BOR4,BOR5,BOSS,HOL1,CBOR,SBOR,DSBO
     COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
     COMMON /INOUT/IW,NFTYP,NETYP,XPOS,CPOS
     COMMON /INFOF/XFACE,YFACE,ZFACE,XCYL,YCYL,ZCYL
     DATA BOR1/1/,BOR2/2/,BOR3/3/,BOR4/4/,BOR5/5/,BOSS/6/,HOL1/8/,
    1    NN1/1/,NN2/2/,NN3/3/,NN4/4/,NN5/5/,NN6/6/,NRP/0/,NN7/7/,
    2    NN8/8/,NN9/9/,NO/0/,CBOR/7/,SBOR/50/,DSBO/51/,NOC2/10/
     DATA NCBL/53/
     WRITE(6,*)' ***CYLINDRICAL FEATURE RECOGNITION BEGIN'
C
C**** FIND THE DISK FACES
C
```

```
      N=0
      IJK=1
      IF (NX .EQ. 0)GO TO 519
C
C**** FIND THE X-TYPE DISK FACE
C
  510 N=N+1
      IF( N .GT. NX)GO TO 519
      K=XFACE(N)
      IF (NFTYP(K) .LT. 197)GO TO 510
      GO TO 540
  519 N=0
      IJK=2
      IF (NY .EQ. 0)GO TO 529
C
C**** FIND THE Y-TYPE DISK FACE
C
  520 N=N+1
      IF (N .GT. NY)GO TO 529
      K=YFACE(N)
      IF (NFTYP(K) .LT. 197)GO TO 520
      GO TO 540
  529 N=0
      IJK=3
      IF (NZ .EQ. 0)GO TO 590
C
C**** FIND THE Z-TYPE DISK FACE
C
  530 N=N+1
      IF (N .GT. NZ)GO TO 590
      K=ZFACE(N)
      IF (NFTYP(K) .LT. 197)GO TO 530
  540 MK=K
      READ(9,REC=MK,FMT=545)NEG,ILIN,(NOEG(I),I=1,12)
  545 FORMAT(8X,I3,41X,13I3)
      LK=0
C
C**** CHECK IF THE DISK FACE IS BOUNDED BY COMPLETE CIRCLE
C
      DO 550 I=1,NEG
      L=NOEG(I)
      IF(NETYP(L) .LT. 2000)GO TO 580
      LK=LK+1
  550 CONTINUE
      IF (LK .EQ. 1) THEN
         L=NOEG(1)
C
C**** FIND THE EDGE WHICH SHARE THE EDGES
C
         CALL FACEN(K,KK,L,DD)
         IF (NFTYP(KK) .GT. 300) THEN
            WRITE(6,555)KK,K
C
  555       FORMAT(1X,'A CYLINDRICAL BOSS',I4,'WITH A DISK TOP',I4)
         CALL DABASE(NN2,NN1,NN2,BOSS,NRP,K,KK,NN3,NN4,NN5,NN6,
     1                   NN7,NN8)
         ELSE
            WRITE(6,556)KK,K
C
  556       FORMAT(/1X,'A BLIND HOLE',I4,' WITH A BOTTOM FACE',I4)
         CALL DABASE(NN1,NN1,NN2,BOR4,NRP,K,KK,NN3,NN4,NN5,NN6,
     1                   NN7,NN8)
         ENDIF
      ELSE
         IF (LK .GT. 2)THEN
            WRITE(6,560)
  560       FORMAT(/1X,'MUTIPLE HOLES OR CYLINDER ON A DISK FACE')
            GO TO 580
         ELSE
            L1=NOEG(1)
            L11=L1
            CALL FACEN(K,K1,L11,DD1)
```

```
              L2=NOEG(2)
              L21=L2
              CALL FACEN(K,K2,L21,DD2)
              NFP1=NFTYP(K1)
              NFP2=NFTYP(K2)
              IF (NFP1 .EQ. NFP2) THEN
                  IF(NFP1 .GT. 300) THEN
                     WRITE(6,565)K1,K2
 565                 FORMAT(/1X,'A STEP BOSS',I4,I4)
                  ELSE
                     D1=XPOS(K1)
                     D2=CPOS(K1)
                     E1=XPOS(K2)
                     E2=CPOS(K2)
                     IF (D1 .NE. E1 .OR. D2 .NE. E2) THEN
                        WRITE(6,566) K1,K2
 566                    FORMAT(/1X,'NON CONCENTER STEP BORE',I4,I4)
                     ELSE
                        NSIGN=1
                        IF (NFTYP(K) .LT. 200)NSIGN=-1
                        KG=K1
                        NG=K2
                        LG=L1
                        IF (DD1 .LT. DD2) THEN
                          KG=K2
                          NG=K1
                          LG=L2
                        ENDIF
                        KK=KG
                        READ(9,REC=KK,FMT=567)NE1,NE2
 567                    FORMAT(55X,2I3)
                        IF (NE1 .EQ. LG) THEN
                            NEG=NE2
                        ELSE
                            NEG=NE1
                        ENDIF
                        CALL FACEN(KG,MM,NEG,DDD)
                        KK=NG
                        READ(9,REC=KK,FMT=567)NE1,NE2
                        IF(NE1 .EQ. LS) THEN
                            NEG=NE2
                        ELSE
                            NEG=NE1
                        ENDIF
                        CALL FACEN(NG,NN,NEG,DDE)
                    WRITE(6,570)KG,NG,K,MM,NN,NSIGN
 570                  FORMAT(/1X,'STEP BORES'/1X,6I4)
                      CALL DABASE(NN1,NN2,NN3,SBOR,NPR,K,KG,NG,NN4,
     1                            NN5,NN6,NN7,NN8)
                     EXAMF(KG)=1
                     EXAMF(NG)=1
                     ENDIF
                ENDIF
              ELSE
                 WRITE(6,575)K1,K2
 575             FORMAT(/1X,'A CYLINDER IN A BLIND HOLE',I4,I4)
                 CALL DABASE(NN1,NO,NN2,NCBL,NPR,K1,K2,NN3,NN4,NN5,
     1                       NN6,NN7,NN8)
              ENDIF
         ENDIF
      ENDIF
 580  CONTINUE
      GO TO (510,520,530),IJK
C
 590  CONTINUE
C
C**** FIND THE THROUGH HOLES
C
      JKL=1
      I=0
 600  I=I+1
      IF(XCYL(I) .EQ. 0)GO TO 699
```

```
      K=XCYL(I)
      IF(EXAMF(K) .EQ. 1)GO TO 600
      GO TO 900
  699 I=0
      JKL=2
  700 I=I+1
      K=YCYL(I)
      IF (K .EQ. 0)GO TO 799
      IF (EXAMF(K) .EQ. 1)GO TO 700
      GO TO 900
  799 I=0
      JKL=3
  800 I=I+1
      K=ZCYL(I)
      IF (K .EQ. 0)GO TO 1000
      IF (EXAMF(K) .EQ. 1)GO TO 800
  900 K1=K
C
      READ(9,REC=K1,FMT=545)NEG,ILIN,(NOEG(J),J=1,12)
      DO 950 J=1,12
          NM=NOEG(J)
          IF (NM .LT. 1)GO TO 955
          READ(10,REC=NM,FMT=930)NNPP
  930     FORMAT(3X,I6)
          IF (NNPP .LT. 2000)GO TO 980
  950 CONTINUE
  955 WRITE(6,960)K
  960 FORMAT(/1X,'CYLINDER',I3.' IS A THROUGH HOLE')
      CALL DABASE(NN1,NO,NN1,HOL1,NPR,K,NN2,NN3,NN4,NN5,NN6,
     1            NN7,NN8)
  980 GO TO (600,700,800),JKL
 1000 CONTINUE
      NPR=0
      CALL READ20(NN3,NUMX)
      NUMX=NUMX-1
 1100 DO 1111 I=1,NUMX
 1111 NCX(I)=0
      DO 1600 IALL=1,NUMX
      IF (NFEAT(IALL) .NE. SBOR)GO TO 1600
      IF (NCX(IALL) .NE. 0)GO TO  1600
      NCX(IALL)=1
C
C*****READ IN A SBOR FEATURE INFORMATION
C
      CALL READ21(IALL,NP,NN,NSR(1))
      NMAN=NSR(1)
      NUP=NSR(2)
      NLOW=NSR(3)
C
 1110 FORMAT(15I4)
C
C     CHECK IF NUP IS A SIDE SURFACE OF TWO FEATURES
C
      NL=MSFRL(NUP)
      IF (LINK(NL) .LE. 0)GO TO 1300
      IF (NODE(NL) .EQ. NMAN) THEN
          NNEW=NODE(LINK(NL))
      ELSE IF (NODE(LINK(NL)) .EQ. NMAN) THEN
          NNEW=NODE(NL)
      ELSE
          WRITE(6,*)'UNCLASSIFIED FEATURE AT 1110 IN HOLBOR'
          GO TO 1600
      ENDIF
C
C     FIND THE FEATURE OF THE MAIN SURFACE NNEW
C
      NFU=MAINF(NNEW)
      CALL READ21(NFU,NP,NN,NSR(1))
      NMAN1=NSR(1)
      NUP1=NSR(2)
      NLOW1=NSR(3)
      IF (NP .NE. SBOR)GO TO 1200
```

```
      NCX(NFU)=1
      IF (NUP .NE. NLOW1)GO TO 1200
       WRITE(6,1130)NFU,IALL,NUP1,NMAN1,NLOW1,NMAN,NLOW
 1130 FORMAT(1X,'A DOUBLE STEP FEATURE',7I4)
      NPR=0
      CALL DABASE(NN2,NO,NN5,DSBO,NPR,NMAN1,NMAN,NUP1,NUP,NLOW,
     1N6,N7,N8)
      GO TO 1600
 1200 WRITE(6,*)'UNCLASSIFIED FEATURE AT 1200 IN HOLBOR'
      GO TO 1600
C
C     THE UPPER SURFACE ASSOCIATED WITH ONLY ONE FEATURE
C     EXAMINING THE LOWER SURFACE OF A SINGLE STEP BORE
C
C***  CHECK IF NLOW IS A SIDE SURFACE OF TWO FEATURE
C
 1300 NL=MSFRL(NLOW)
      IF (LINK(NL) .LE. 0)GO TO 1500
      IF (NODE(NL) .EQ. NMAN) THEN
         NNEW=NODE(LINK(NL))
      ELSE IF (NODE(LINK(NL)) .EQ. NMAN) THEN
         NNEW=NODE(NL)
      ELSE
         WRITE(6,*)'UNCLASSIFIED FEATURE AT 1300 IN HOLBOR'
         GO TO 1600
      ENDIF
C
C     FIND THE FEATURE OF THE MAIN SURFACE NNEW
C
      NFW=MAINF(NNEW)
      CALL READ21(NFW,NP,NN,NSR(1))
      NMAN1=NSR(1)
      NUP1=NSR(2)
      NLOW1=NSR(3)
C
      IF (NP .NE. SBOR)GO TO 1400
      NCX(NFW)=1
      IF (NUP1 .NE. NLOW)GO TO 1200
      WRITE(6,1130)IALL,NFW,NUP,NMAN,NLOW,NMAN1,NLOW1
      NPR=0
      CALL DABASE(NN2,NO,NN5,DSBO,NPR,NMAN,NMAN1,NUP,NLOW,NLOW1,
     1N6,N7,N8)
      GO TO 1600
 1400 IF (NP .NE. BOR4)GO TO 1200
      WRITE(6,1410)IALL,NFW,NUP,NMAN,NLOW,NMAN1
 1410 FORMAT(1X,'A BORE 5 TYPE FEATURE',6I4)
      CALL DABASE(NN2,NO,NN4,BOR5,NPR,NMAN,NMAN1,NUP,NLOW,
     1NN5,NN6,NN7,NN8)
      GO TO 1600
C
C     THIS IS A COUNTER BORE FEATURE
C
 1500 CALL DABASE(NN1,NO,NN3,CBOR,IALL,NMAN,NUP,NLOW,NN4,
     1NN5,NN6,NN7,NN8)
 1600 CONTINUE
C
C     FIND THE BORE 3 AND BORE 4 TYPE FEATURE
C
      CALL READ20(NN3,NUMX)
      NUMX=NUMX-1
      DO 1900 IBOR=1,NUMX
      IF (NFEAT(IBOR) .NE. DSBO)GO TO 1900
      IF (NCX(IBOR) .EQ. 1)GO TO 1900
      CALL READ21(IBOR,NP,NN,NSR(1))
      NMAN=NSR(1)
      NMAN1=NSR(2)
      NUP=NSR(3)
      NMDL=NSR(4)
      NLOW=NSR(5)
C
      NL=MSFRL(NUP)
      IF (LINK(NL) .NE. 0)GO TO 1800
```

```
      NL=MSFRL(NLOW)
      IF (LINK(NL) .EQ. 0)GO TO 1700
      IF (NODE(NL) .EQ. NMAN1) THEN
          NNEW=NODE(LINK(NL))
      ELSE IF (NODE(LINK(NL)) .EQ. NMAN1) THEN
          NNEW=NODE(NL)
      ELSE
          WRITE(6,*)'UNCLASSIFIED FEATURE IN LINE 27500'
          GO TO 1900
      ENDIF
C
      NFW=MAINF(NNEW)
      IF (NFEAT(NFW) .NE. DSBO)GO TO 1800
      CALL READ21(NFW,NP,NN,NSR(1))
      NMAN2=NSR(1)
      NMAN3=NSR(2)
      NUP1=NSR(3)
      NMD1=NSR(4)
      NLW1=NSR(5)
C
      IF (NLOW .NE. NLW1)GO TO 1800
      WRITE(6,*)'BOR3 FEATURE',NMAN,NMAN1,NMAN2,NMAN3
      CALL DABASE(NN4,NO,NN9,BOR3,NPR,NMAN,NMAN1,NMAN2,NMAN3,
     1NUP,NMDL,NLOW,NMD1,NUP1)
      NCX(NFW)=1
      GO TO 1900
 1700 CALL DABASE(NN2,NO,NN5,BOR2,NPR,NMAN,NMAN1,NUP,NMDL,NLOW,
     1NN6,NN7,NN8)
       WRITE(6,*)'BOR2 TYPE FEATURE'
      GO TO 1900
 1800 WRITE(6,*)'UNCLASSIFIED FEATURE AT 1800 IN HOLBOR'
 1900 CONTINUE
C     WRITE(6,2000)
 2000 FORMAT(1X,'CYLINDRICAL FEATURE RECOGNITION COMPLETED')
      RETURN
      END
C*********************************************************************
C     THIS SUBROUTINE IS THE MAIN ROUNTINE FOR RECOGNIZING THE       *
C     NON-CYLINDRICAL FEATURES.  THE KEY FACES ARE IDENTIFUED        *
C     FIRST, THEN FEATURES ON EACH LEVEL WILL BE RECOGNIZED.         *
C     KK IS THE NUMBER OF SURFACES.                                  *
C     NSORT STORES THE ORDERED SURFACES ACCORDING TO THE NUMBER      *
C           OF EDGES.                                                *
C     EEDGE STORS THE NUMBER OF EDGES THAT A SURFACE HAS.            *
C*********************************************************************
      SUBROUTINE BLOCGROV(KK,NSORT,EEDGE)
C
      DIMENSION NSL(4),PSL(4),NP(4),NSORT(99),EEDGE(99),NMIX(10)
      DIMENSION NETYP(99),NOEG(99),DATA(6),DDTT(12),NFTYP(99)
      DIMENSION RANKN(99),RANKFL(99),NFNAM(99),FEATURE(20)
      DIMENSION RANKP(99),RANKS(99),RANKE(99),EDGSML(9),EDGLAR(9),
     1          RANKR(99),RANKL(99),KONER(4),IDOIO(99)
      INTEGER RANKN,RANKR,RANKL,RANKFL,FEATURE,EDGSML,EDGLAR
      INTEGER XFACE(35),ZFACE(35),YFACE(35),PASS,A,B,C,D
      INTEGER XCYL(35),YCYL(35),ZCYL(35)
      DIMENSION MAINF(99),MFSRL(99),MF12(2),NFEAT(20),NODE(300)
      DIMENSION LINK(300),EXAMF(99),NSR(20),NMANF(20)
      INTEGER BOR1,BOR2,BOR3,BOR4,BOR5,BOSS,CBOR,HOL1,HOL6,
     1NOC2,PAD1,POK1,POK2,SLO1,SLO2,GROV,STEP,PLAN,BOS1,BOS2,
     2GRV1,BRDG,BLOC,CBLO
      COMMON /INOUT/IW,NFTYP,NETYP
      COMMON /FETUR/BOR1,BOR2,BOR3,BOR4,BOR5,BOSS,CBOR,HOL1,HOL6,
     1NOC2,PAD1,POK1,POK2,SLO1,SLO2,GROV,STEP,PLAN,BOS1,
     2BOS2,BOS3,GRV1,BRDG,BLOC,CBLO
      COMMON /INFOR/MAINF,MFSRL,NFEAT,NODE,LINK,EXAMF,NMANF
      COMMON /INFOF/XFACE,YFACE,ZFACE,XCYL,YXYL,ZCYL
      DATA BOR1/1/,BOR2/2/,BOR3/3/,BOR4/4/,BOR5/5/,BOSS/6/,
     1CBOR/7/,HOL1/8/,HOL6/9/,NOC2/10/,PAD1/11/,POK1/12/,
     2POK2/13/,SLO1/14/,SLO2/15/,GROV/16/,STEP/17/,PLAN/18/,
     3BOS1/19/,BOS2/20/,GRV1/21/,BRDG/22/,BLOC/23/,CBLO/24/
      DATA N1/1/,N2/2/,N3/3/,N4/4/,N5/5/,N6/6/,N7/7/,N8/8/,N9/9/
      DATA NRP/0/,NO/0/
```

```
C
      WRITE(6,10)
   10 FORMAT(1X,'*** NON-CYLINDRICAL FEAURE RECOGNITION BEGIN')
C
      DO 2000 PASS=1,3
C
C**** CHECK IF A SURFACE IS AN ELIGIBLE SURFACE
C
      DO 2000 IJK=1,KK
      KL=NSORT(IJK)
      MM=KL
      IF (EXAMF(KL) .GT. 0 .OR. EEDGE(KL) .LT. 4.0)GO TO 2000
      IF (NFTYP(KL) .LT. 0)GO TO 2000
      IF (NFTYP(KL) .GT. 103)GO TO 830
C
C**** CONSTRUCTING THE LOOPS
C
      CALL EDGERANK(NX,NY,MM,MM,RANKN(1),RANKP(1),RANKS(1),
     1RANKE(1),RANKL(1),RANKFL(1),EDGSML(1),EDGLAR(1),IDOIO(1),
     2NFNAM(1),RANKR(1))
      CALL EDGELINK(NX,NY,INERF,INERL,KONER(1),RANKN(1),RANKP(1),
     1RANKS(1),RANKE(1),RANKL(1),RANKFL(1),EDGSML(1),EDGLAR(1),
     2IDOIO(1),NFNAM(1),FEATURE(1),RANKR(1))
C
      K=1
      I=1
      L=FEATURE(K)
      M=L
      N=NFNAM(L)
      NFFTYP=NFTYP(N)
C
C**** CONSTRUCTING THE OUTER LOOP
C
  200 N=NFNAM(L)
      IF (NFTYP(N) .NE. NFFTYP)I=I+1
      L=RANKFL(L)
      NFFTYP=NFTYP(N)
      IF (L .NE. M)GO TO 200
      IF (I .EQ. 4)GO TO 500
      IF (PASS .LT. 2)GO TO 2000
C
C**** THE SURFACE IS ASSUMED TO BE A PLAN FEATURE
C
      CALL BLOCAG(KL,INERF,K,FEATURE(1),RANKFL(1),NFNAM(1),
     1            RANKN(1))
      EXAMF(KL)=1
      GO TO 2000
C
C**** GET THE FIRST NODE OF THE OUTER LOOP
C
  500 I=0
C
C**** NS IS THE NODE NUMBER
C**** NG IS THE EDGE NUMBER
C**** NPO IS THE EDGE TYPE
C
      NS=FEATURE(K)
      NG=RANKN(NS)
      NPO=NETYP(NG)
      NP1=NPO
      NE=NS
      NM=NFNAM(NE)
  505 FORMAT(/1X,'FACE #',I4)
  510 I=I+1
C
C**** NSL STORES THE LOOP NODE NUMBER
C**** PSL IS THE POSITION OF THE EDGE
C
      NSL(I)=NE
      PSL(I)=RANKP(NE)
      NPRV=NPO
      NP(I)=NFTYP(NM)
```

```
C
C**** CHECK THE EDGE CODES OF THE LOOP
C
  530 NE=RANKFL(NE)
      NG=RANKN(NE)
      NPO=NETYP(NG)
      NM=NFNAM(NE)
      IF (NPO .EQ. NPRV)GO TO 530
      IF (I .NE. 4)GO TO 510
      IF (NPO .EQ. NP1)GO TO 550
      WRITE(6,540)KL,NPO,NP1
  540 FORMAT(/1X,'UNCLASSIFIED FEATURE KL,NPO,NP1',3I4)
      GO TO 2000
  550 IK=0
C
C**** CHECK IF THE FACE IS RECTANGULAR OR ECLLIPTICAL
C
      DO 560 I=1,4
      IF (NP(I) .GT. 103)GO TO 850
  560 CONTINUE
      NS1=NSL(1)
      NL1=NSL(3)
      DA1=PSL(1)
      DA3=PSL(3)
      NS2=NSL(2)
      NL2=NSL(4)
      DA2=PSL(2)
      DA4=PSL(4)
C
  610 NFS1=NFNAM(NS1)
      NFL1=NFNAM(NL1)
      NFS2=NFNAM(NS2)
      NFL2=NFNAM(NL2)
      NTYP1=NFTYP(NFS1)
      NTYP2=NFTYP(NFL1)
      NTYP3=NFTYP(NFS2)
      NTYP4=NFTYP(NFL2)
C
C**** CHECK THE SURFACE NORMAL DIRECTIONS
C
      CALL DIRECT(DA1,DA3,NTYP1,NTYP2,A,B)
      CALL DIRECT(DA2,DA4,NTYP3,NTYP4,C,D)
C
      IF (A.EQ.B .AND. A.GT.0 .AND. C.EQ.D .AND. C.GT.0) THEN
          WRITE(6,720)KL,NFS1,NFL1,NFS2,NFL2
  720     FORMAT(1X,'BLOCK :',5I4)
          CALL DABASE(N1,N4,N5,BLOC,NRP,KL,NFS1,NFL1,NFS2,NFL2,
     1                   N6,N7,N8)
      ELSE IF (A.EQ.B .AND. A.LT.0 .AND. C.EQ.D .AND. C.LT.0) THEN
          WRITE(6,730)KL,NFS1,NFL1,NFS2,NFL2
  730     FORMAT(1X,'POKET 1:',5I4)
          CALL DABASE(N5,N4,N5,POK1,NRP,KL,NFS1,NFL1,NFS2,NFL2,
     1                   N6,N7,N8)
      ELSE IF (A.EQ.B .AND. A.GT.0 .AND. C.EQ.D .AND. D.LT.0) THEN
          WRITE(6,740)KL,NFS2,NFL2,NFS1,NFL1
  740     FORMAT(1X,'GROVE:',5I4)
          CALL DABASE(N1,N2,N5,GROV,NRP,KL,NFS2,NFL2,NFS1,NFL1,
     1                   N6,N7,N8)
      ELSE IF (A.EQ.B .AND. A.LT.0 .AND. C.EQ.D .AND. C.GT.0) THEN
          WRITE(6,740)KL,NFS1,NFL1,NFS2,NFL2
          CALL DABASE(N1,N2,N5,GROV,NRP,KL,NFS1,NFL1,NFS2,NFL2,
     1                  N6,N7,N8)
      ELSE
          GO TO 750
      ENDIF
      GO TO 1800
C
  750 IF (PASS .NE. 2)GO TO 800
      IF (A.EQ.B .AND. A.LT.0 .AND. C.NE.D .AND. C.GT.0) THEN
          WRITE(6,770)KL,NFS1,NFL1,NFL2,NFS2
  770     FORMAT(1X,'BLIND GROOVE:',5I4)
          CALL DABASE(N4,N0,N5,GRV1,NRP,KL,NFS1,NFL1,NFL2,NFS2,
```

```
     1                 N6,N7,N8)
      ELSE IF (A.EQ.B .AND. A.LT.0 .AND. C.NE.D .AND. C.LT.0)THEN
          WRITE(6,770)KL,NFS1,NFL1,NFS2,NFL2
          CALL DABASE(N4,N0,N5,GRV1,NRP,KL,NFS1,NFL1,NFS2,NFL2,
     1                 N6,N7,N8)
      ELSE IF (A.NE.B .AND. A.LT.0 .AND. C.EQ.D .AND. C.LT.0) THEN
          WRITE(6,770)KL,NFS2,NFL2,NFS1,NFL1
          CALL DABASE(N4,N0,N5,GRV1,NRP,KL,NFS2,NFL2,NFS1,NFL1,
     1                 N6,N7,N8)
      ELSE IF (A.NE.B .AND. A.GT.0 .AND. C.EQ.D .AND. C.LT.0) THEN
          WRITE(6,770)KL,NFS2,NFL2,NFL1,NFS1
          CALL DABASE(N4,N0,N5,GRV1,NRP,KL,NFS2,NFL2,NFL1,NFS1,
     1                 N6,N7,N8)
      ELSE
      GO TO 800
      ENDIF
      GO TO 1800
C
  800 IF (PASS .NE. 3)GO TO 2000
      IF (A.EQ.B .AND. A.GT.0 .AND. C.NE.D .AND. C.LT.0) THEN
          NFFS=NFS2
          GO TO 825
  810     FORMAT(1X,'A STEP:',5I4)
      ELSE IF (A.EQ.B .AND. A.GT.0 .AND. C.NE.D .AND. C.GT.0) THEN
          NFFS=NFL2
          GO TO 825
      ELSE IF (A.NE.B .AND. A.GT.0 .AND. C.EQ.D .AND. C.GT.0) THEN
          NFFS=NFL1
           GO TO 825
      ELSE IF (A.NE.B .AND. A.LT.0 .AND. C.EQ.D .AND. C.GT.0) THEN
          NFFS=NFS1
          GO TO 825
      ELSE IF (A.NE.B .AND. A.LT.0 .AND. C.NE.D .AND. C.LT.0) THEN
          WRITE(6,820)KL,NFS1,NFS2,NFL1,NFL2
  820     FORMAT(1X,'CORNER BLOCK:',5I4)
          CALL DABASE(N3,N0,N5,CBOR,NRP,KL,NFS1,NFS2,NFL1,NFL2,
     1                 N6,N7,N8)
      ELSE IF (A.NE.B .AND. A.LT.0 .AND. C.NE.D .AND. C.GT.0) THEN
          WRITE(6,820)KL,NFS1,NFL2,NFL1,NFS2
          CALL DABASE(N3,N0,N5,CBOR,NRP,KL,NFS1,NFL2,NFL1,NFS2,
     1                 N6,N7,N8)
      ELSE IF (A.NE.B .AND. A.GT.0 .AND. C.NE.D .AND. C.GT.0) THEN
          WRITE(6,820)KL,NFL1,NFL2,NFS1,NFS2
          CALL DABASE(N3,N0,N5,CBOR,NRP,KL,NFL1,NFL2,NFS1,NFS2,
     1                 N6,N7,N8)
      ELSE IF (A.NE.B .AND. A.GT.0 .AND. C.NE.D .AND. C.LT.0) THEN
          WRITE(6,820)KL,NFL1,NFS2,NFS1,NFL2
          CALL DABASE(N3,N0,N5,CBOR,NRP,KL,NFL1,NFS2,NFS1,NFL2,
     1                 N6,N7,N8)
      ELSE
      GO TO 2000
      ENDIF
C
  825 IF (EEDGE(NFFS) .NE. 4)GO TO 2000
      WRITE(6,810)KL,NFFS
      CALL DABASE(N2,N0,N2,STEP,NRP,KL,NFFS,N3,N4,N5,N6,N7,N8)
      GO TO 1800
C
C
C**** FINDING NOTCH
C
  830 IF (PASS .NE. 3)GO TO 2000
      NRC=KL
      READ(9,REC=NRC,FMT=831)NEG,(NOEG(J),J=1,NEG)
  831 FORMAT(8X,I3,44X,12I3)
C
      NOTCH=0
      KDYP=0
      NORP=-1
      IF (NEG .GT. 4)GO TO 845
      DO 840 I=1,NEG
      L=NOEG(I)
```

```
      NRC=L
      READ(10,REC=NRC,FMT=832)NF1,NF2
  832 FORMAT(18X,2I3)
      KK=NF1
      IF (NF1 .EQ. K)KK=NF2
      NRC=KK
      READ(9,REC=NRC,FMT=833)DATA
  833 FORMAT(20X,F9.5)
      IF (NETYP(L) .GT. 2000)GO TO 838
      KB=KA
      KA=KK
      KYP=NFTYP(KK)
      IF (NORP .LT. 0) THEN
          NORP=KYP
      ELSE
          IF (NORP .EQ. KYP) THEN
              NOTCH=1
          ELSE
              I1=ABS(NORP-100)
              I2=ABS(KYP-100)
              IF (I1 .EQ. I2) THEN
                  WRITE(6,834)KL
  834             FORMAT(1X,'A SLOT WITH THE NOTCH',I4)
              ELSE
                  WRITE(6,835)KL
  835             FORMAT(1X,'CYLINDER FACE',I4,'IS A FILET')
              ENDIF
          ENDIF
      ENDIF
C
  838 DA2=DA1
      DA1=DADA
      K2=K1
      K1=KK
  840 CONTINUE
      KYP1=NFTYP(K1)
      KYP2=NFTYP(K2)
      IF (NOTCH .EQ. 0)GO TO 2000
      IF (KYP1 .GT. 103 .OR. KYP2 .GT. 103) THEN
          WRITE(6,841)KL
  841     FORMAT(1X,'NOTCH',I4,' HAS A BLIND END')
      ELSE
          CALL DIRECT(DA1,DA2,KYP1,KYP2,A,B)
          IF (A .NE. B) THEN
              WRITE(6,841)KL
          ELSE
              IF (A.LT.0) THEN
              WRITE(6,842)KL
  842         FORMAT(1X,'BOTH ENDS OF NOTCH',I4,' ARE BLIND')
              ELSE
                  WRITE(6,843)KL
  843             FORMAT(1X,'A NOTCH:',I4)
                  CALL DABASE(N1,N0,N5,NOC2,NRP,K,KA,KB,K1,K2,
     1                        N6,N7,N8)
              ENDIF
          ENDIF
      ENDIF
      GO TO 2000
C
  845 WRITE(6,848)
  848 FORMAT(1X,'UNCLASSIFIED FEATURE IN BLOCGROV 848')
      GO TO 2000
C
C**** THE KEY SURFACE IS ECLLIPTICAL
C
  850 I1=(NP(1)+3)/100
      I2=(NP(3)+3)/100
      ID=ABS(NP(1)-I1*100)-ABS(NP(3)-I2*100)
      IF (ID .NE. 0)GO TO 2000
      I1=(NP(2)+3)/100
      I2=(NP(4)+3)/100
      ID=ABS(NP(2)-I1*100)-ABS(NP(4)-I2*100)
```

```
      IF (ID .NE. 0)GO TO 2000
      II=1
      IF (NP(1) .GT. 103)II=2
      III=II+2
      IF (NP(II) .NE. NP(III))GO TO 880
      IF (ID .NE. 0)GO TO 2000
  860 WRITE(6,870)
  870 FORMAT(1X,'UNCLASSIFIED FEATURE IN BLOCGROV 870')
      GO TO 2000
  880 I2=II+1
      IF (NP(I2) .GT. 203)GO TO 1000
C
C**** FIND THE NAME OF THE CYLINDRICAL SURFACE
C
      NEG=0
      JJ=1
      IF (II .EQ. 1)JJ=2
      JK=NFNAM(JJ)
      CALL FACEN(JK,NCBD1,NEG,DDD)
      NEG=0
      JK=NFNAM(JJ+2)
      CALL FACEN(JK,NCBD2,NEG,DDD)
C
C**** FIND THE OPEN FACES
C
      IRJ=NCBD2
      READ(9,REC=IRJ,FMT=888)(NOEG(I),I=1,12)
  888 FORMAT(55X,12I3)
      NEC=NEG-1
      DO 890 I=1,12
      IF (NOEG(I) .GT. 2000 .AND. NOEG(I) .NE. NEC)GO TO 900
  890 CONTINUE
      WRITE(6,895)
  895 FORMAT(1X,'ERROR IN BLOCGROV 895')
      STOP
  900 NEG=NOEG(I)
      CALL FACEN(NCBD2,NOPNF,NEG,DDD)
      NCURF=KL
      NEND1=NP(II)
      NEND2=NP(III)
      JJ=1
      IF (II .EQ. 1)JJ=2
      NBFL1=NP(JJ)
      NBFL2=NP(JJ+1)
      DA1=PSL(JJ)
      DA2=PSL(JJ+1)
      CALL DIRECT(DA1,DA2,NBFL1,NBFL2,DA1,DA2)
C
      IF (A .EQ. B .AND. A .GT. 0) THEN
          WRITE(6,910)NCURF,NEND1,NEND2,NCBD1,NCBD2,NBLF1,
     1                   NBLF2,NOPNF
          NPIC=BOS1
  910     FORMAT(1X,'A BLOCK WITH ROUNDED ENDS',8I4)
      ELSE IF (A .EQ. B .AND. A.LT.0) THEN
          WRITE(6,915)NCURF,NEND1,NEND2,NCBD1,NCBD2,NBLF1,
     1                   NBLF2,NOPNF
          NPIC=BOS2
  915     FORMAT(1X,'A BLIND SLOT WITH ROUNDED ENDS',8I4)
      ELSE
          GO TO 860
      ENDIF
      CALL DABASE(N7,N0,N8,NPIC,NRP,NCURF,NEND1,NEND2,NCBD1,
     1               NCBD2,NBLF1,NBLF2,NOPNF)
      GO TO 2000
C
 1000 IF (PASS .EQ. 1)GO TO 2000
      NEG=0
      JJ=NSL(I2)
      JK=NFNAM(JJ)
      MM=JK
      READ(9,REC=MM,FMT=888)(NOEG(I),I=1,12)
      N=0
```

```
      DO 1100 I=1,4
      K=NOEG(I)
      NEG=K
      CALL FACEN(JK,KK,NEG,DDD)
      IF (NETYP(K) .GT. 2000) THEN
           NOPF2=NOPF1
           NOPF1=KK
      ELSE
           NBLO2=NBLO1
           NBLO1=KK
      ENDIF
 1100 CONTINUE
      NEND1=JK
      JJ=I2
      IF (I2 .GT. 2) JJ=JJ-4
      JK1=NSL(JJ+2)
      NEND2=NFNAM(JK1)
      WRITE(6,1120)NBLO1,NBLO2,NEND1,NEND2,NOPF1,NOPF2
 1120 FORMAT(1X,'A SLOT:',6I4)
      CALL DABASE(N4,N0,N6,SL01,NRP,NBLO1,NBLO2,NEND1,NEND2,
     1            NOPF1,NOPF2,N7,N8)
      GO TO 2000
 1800 IF (INERF .LT. 2)GO TO 2000
C
C**** RELATING THE EXTERNAL FEATURE WITH THE INTERNAL
C**** LOOPS.
C
      K=0
      DO 1900 J=2,INERF
         NF=FEATURE(J)
         K=K+1
         NM=NFNAM(NF)
         NSR(K)=NM
 1900 CONTINUE
      NFN=MAINF(KL)
      NRC=N0
      CALL WRIT22(NRC,NFN,K,KL,NSR(1))
 2000 CONTINUE
      IF (PASS .EQ. 2) THEN
C
C**** CHECK THE NON-BASIC FEATURES: GROOVE AND HOLE-2
C
          CALL GROVBRDG
      ENDIF
 3000 CONTINUE
      RETURN
      END
C*****************************************************************
C     SUBROUTINE BLOCAG IDENTIFIES THE PLANE FEATURES AND THE    *
C     INNER LOOPS OF A PLANE FEATURE                             *
C*****************************************************************
      SUBROUTINE BLOCAG(KL,INERF,ILOP,FEATURE,RANKFL,NFNAM,RANKN)
C*****************************************************************
C**   KL      : THE KEY SURFACE
C**   INERF   : NUMBER OF FEATURE
C**   ILOP    : THE OUTER LOOP NUMBER
C**   FEATURE : THE FIRST NODE NUMBERS OF LOOPS
C**   RANKFL  : THE LOOP NODE
C**   NFNAM   : SURFACE NAME THAT SHARE THE EDGE WITH THE PLANE
C**   RANKN   : THE EDGE NAME
C**   KOUNT   : NUMBER OF BOUNDARY EDGES
C**   BLEDG   : EDGE NUMBERS OF THE PLANE.  NEGATIVE NUMBER
C**             INDICATES THE SURFACE SHARED THE EDGE IS AN
C**             EXTRUSION TO THIS PLANE SURFACE.
C*****************************************************************
      DIMENSION NETYP(99),XPOS(99),DATA(6),DDTT(12),NFTYP(99)
      INTEGER FEATR(20),CONER(4),EDGES(9),EDGEL(9),PATERN(50)
      INTEGER RANKN(99),RANKFL(99),NFNAM(99),FEATURE(20)
      INTEGER BG(3),PLAN,BLEDG(99),NAME(2)
      DIMENSION NRANK(99),SRANK(99),IRANK(99),PRANK(99),LRANK(99),
     1          ERANK(99),LFRANK(99),KKNER(4),IDO(99),NFAM(99),
      DIMENSION MAINF(99),MSFRL(99),NFEAT(20),NODE(300),NMANF(20)
```

```
      DIMENSION LINK(300),EXAMF(99),NMIX(10),NSR(20),NEYP(99)
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
      COMMON /INOUT/IW,NFTYP,NETYP,XPOS
      DATA NN0/0/,NN1/1/,PLAN/18/
      KOUNT=0
      IWR=1
      N1=FEATURE(ILOP)
      NE=N1
  100 NAME(2)=NAME(1)
      NAME(1)=RANKN(NE)
      MM=NFNAM(NE)
      KOUNT=KOUNT+1
      NL=NE
      NN=NE
      KUNT=KOUNT
      BLEDG(KOUNT)=NAME(1)
  120 NMM=NFNAM(NN)
      NN=RANKFL(NN)
      NNM=NFNAM(NN)
C
C**   CHECK IF THE EDGE IS A CONTINUOUS EDGE OF THE LAST ONE
C
      IF (NFTYP(NNM) .NE. NFTYP(NMM))GO TO 150
      NAME(2)=RANKN(NN)
      NL=NN
      KOUNT=KOUNT+1
      BLEDG(KOUNT)=NAME(2)
      GO TO 120
  150 CONTINUE
C
C**   CHECK IF THE BOUNDING SURFACE IS ABOVE THE PLANE FACE
C
      CALL EDGERANK(NX,NY,MM,KL,NRANK(1),PRANK(1),SRANK(1),
     1ERANK(1),LRANK(1),LFRANK(1),EDGES(1),EDGEL(1),IDO(1),
     2NFAM(1),IRANK(1))
C
      CALL EDGELINK(NX,NY,INNR,INNL,KKNER(1),NRANK(1),PRANK(1),
     1SRANK(1),ERANK(1),LRANK(1),LFRANK(1),EDGES(1),EDGEL(1),
     2IDO(1),NFAM(1),FEATR(1),IRANK(1))
C
C**   OUTER FEATURE IS ALWAYS THE FIRST FEATURE
C
      NF=FEATR(1)
      NAN=NRANK(NF)
      N2=NF
      NP=NF
C
C**   FIND THE EDGE WHICH ASSOCIATE WITH THE KEY SURFACE(KL)
C
  200 NF=LFRANK(NF)
      M1=NFAM(NP)
      M2=NFAM(NF)
      NAM=NRANK(NF)
      IF (NFTYP(M2) .EQ. NFTYP(M1))GO TO 260
      NAN=NAM
C
  250 IF (NAM .EQ. NAME(1) .OR. NAM .EQ. NAME(2))GO TO 500
  260 NP=NF
      IF (NF .NE. N2)GO TO 200
  300 CONTINUE
      WRITE(6,400)
  400 FORMAT(/1X,'SYSTEM ERROR IN BLOCAG 400')
      STOP
C
C**   FIND THE NEXT EDGE TO THE EDGE WHICH IS AN EDGE
C**   SHARED WITH THE KEY SURFACE.
C
  500 NS=LFRANK(NF)
      NSM=NFNAM(NS)
      NFM=NFNAM(NF)
      IF (NFTYP(NSM) .NE. NFTYP(NFM))GO TO 600
      NF=NS
```

```
      GO TO 500
  600 CONTINUE
      FP=XPOS(KL)
      PS=SRANK(NP)
      PE=ERANK(NP)
      SS=SRANK(NS)
      SE=ERANK(NS)
      N=0
      IF (NFTYP(KL) .GT. 100) THEN
         IF (PS .EQ. FP .AND. SS .EQ. FP)N=2
      ELSE
         IF (PE .EQ. FP .AND. SE .EQ. FP)N=2
      ENDIF
C
  700 IF (N .LT. 2)GO TO 900
      DO 850 I=KUNT,KOUNT
  850 BLEDG(I)=-BLEDG(I)
C
  900 NE=RANKFL(NL)
      IF (NE .NE. N1)GO TO 100
  950 CONTINUE
      WRITE(6,1000)KL,(BLEDG(J),J=1,KOUNT)
 1000 FORMAT(/1X,'FACE #',I3,' IS A PLANE SURFACE WITH THE',
     1'FOLLOWING BOUNDARIES, NEGATIVE NUMBER MEANS A BLOCK EDGE'/
     21X,10(I5)/1X,10(I5)/1X,10(I5)/1X,10(I5))
      K=0
      N=0
      KP=0
C
C**   CHECK IF A PAD EXIST ON SURFACE KL
C
      DO 1100 I=1,KOUNT
      IF (BLEDG(I) .LT. 0) THEN
          IF (N .EQ. 3)N=0
          N=N+1
          BG(N)=-BLEDG(I)
      ELSE
          IF (N .EQ. 2 .OR. N .EQ. 3) THEN
                CALL PAD(KL,N,KP,BG(1),XPOS(1))
                IF (KP .GT. 0)K=K+1
                IF (KP .GT. 0)NSR(K)=KP
          ENDIF
C
          N=0
      ENDIF
 1100 CONTINUE
      IF (N .EQ. 2 .OR. N .EQ. 3)CALL PAD(KL,N,LP,BG(1),XPOS(1))
      IF (LP .GT. 0)K=K+1
      IF (LP .GT. 0)NSR(K)=LP
      IF (INERF .GT. 1) THEN
          DO 1200 J=2,INERF
             NF=FEATURE(J)
             K=K+1
 1200        NSR(K)=NFNAM(NF)
      ENDIF
C
      CALL DABAS1(NN1,NN0,NN1,PLAN,NN0,KL)
C
      NFN=MAINF(KL)
      NRC=0
      IF (K .GT. 0)CALL WRIT22(NRC,NFN,K,KL,NSR(1))
C
      RETURN
      END
C
C*********************************************************************
C     THIS SUBROUTINE IDENTIFY THE SECONDARY FEATURES: GROOVE,       *
C     RECTANGULAR THROUGH HOLE, AND T-SLOT                           *
C*********************************************************************
      SUBROUTINE GROVBRDG
C
      DIMENSION NETYP(99),NOEG(99),DATA(6),DDTT(12),NFTYP(99)
```

```
      INTEGER FEATR(20),CONER(4),EDGES(9),EDGEL(9),PATERN(50),
      INTEGER RANKN(99),RANKFL(99),NFNAM(99),FEATURE(20),BLEDG(99)
      DIMENSION NRANK(99),SRANK(99),IRANK(99),PRANK(99),LRANK(99),
     1          ERANK(99),LFRANK(99),KKNER(4),IDO(99),NFAM(99),
     2          NEYP(99),NAME(2),
     3          MAINF(99),MSFRL(99),MF12(2),NFEAT(20),NODE(300),
     4          LINK(300),EXAMF(99),NSR(9),NMANF(20),XPOS(99)
      INTEGER BOR1,BOR2,BOR3,BOR4,BOR5,BOSS,CBOR,HOL1,HOL6,
     1NOC2,PAD1,POK1,POK2,SLO1,SLO2,GROV,STEP,PLAN,BOS1,BOS2,
     2GRV1,BRDG
      COMMON /INOUT/IW,NFTYP,NETYP,XPOS
      COMMON /FETUR/BOR1,BOR2,BOR3,BOR4,BOR5,BOSS,CBOR,HOL1,HOL6,
     1NOC2,PAD1,POK1,POK2,SLO1,SLO2,GROV,STEP,PLAN,BOS1,
     2BOS2,BOS3,GRV1,BRDG,BLOC,CBLO
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
      DATA NM1/1/,NM2/2/,NM3/3/,NM4/4/,NM5/5/,NM6/6/,NM7/7/,NM8/8/
      DATA N0/0/,NPR/0/
      CALL READ20(NM1,NNN)
  100 FORMAT(6I4)
C
      DO 1000 II=1,NNN
      N=MAINF(II)
      M0=II
      NFYP=NFEAT(N)
      IF (NFYP .NE. GROV)GO TO 1000
      M=MSFRL(II)
C
C**** M=0 THIS FEATURE CURRENTLY IS A GROOVE
C
      IF (M .NE. 0)GO TO 130
      CALL READ21(N,NP,NN,NSR(1))
  110 FORMAT(15I4)
      CALL DABAS1(NM1,NM2,NM5,GROV,N,NSR(1))
      WRITE(6,*)'A GROVE',(NSR(I),I=1,3)
      DO 125 I=1,3
      IK=NSR(I)
  125 EXAMF(IK)=1
      GO TO 1000
  130 M1=LINK(M)
      IF (M1 .NE. 0)GO TO 600
C
C**** CHANGE THE BASE FACE
C
      M0=NODE(M)
      CALL CHEKGROV(M,MRT)
      IF (MRT .NE. GROV) THEN
         WRITE(6,150)
  150    FORMAT(/1X,'SYSTEM ERROR IN GROVBRDG 150')
         STOP
      ENDIF
      M=MSFRL(M0)
      M1=LINK(M)
      IF (M .EQ. 0) THEN
         WRITE(6,200)
  200    FORMAT(/1X,'UNCLASSIFIED FEATURE IN GROVBRDG 100')
         STOP
      END IF
C
  600 N3=NODE(M)
      N31=MSFRL(N3)
      M31=NODE(N31)
      N32=LINK(N31)
      M32=NODE(N32)
C
      N4=NODE(M1)
      N41=MSFRL(N4)
      M41=NODE(N41)
      N42=LINK(N41)
      M42=NODE(N42)
C
      M33=M31
      IF (M31 .EQ. M0)M33=M32
```

```
      M44=M41
      IF (M41 .EQ. M0)M44=M42
C
      CALL CHEKGROV(M33,MRT)
      IF (MRT .LT. GROV)GO TO 700
      IF (M33 .NE. M44) THEN
          WRITE(6,*)'UNCLASSIFIED FEATURE',M33,M44
          STOP
      ENDIF
      WRITE(6,650)M0,N3,N4,M33
  650 FORMAT(/1X,'FACE',3I3,' AND',I3,' ARE THROUGH HOLES')
      CALL DABASE(NM4,N0,NM4,HOL2,NPR,M0,N3,N4,M33,NM5,NM6,NM7,NM8)
      GO TO 1000
C
C
  700 IF (M33 .EQ. M44) THEN
         WRITE(6,870)M0,N3,N4,M44
         CALL DABASE(NM3,N0,NM4,BRDG,NPR,M0,N3,N4,M44,NM5,NM6,
     1               NM7,NM8)
  870    FORMAT(/1X,3I3,' ARE A BRIDGE OVER FACE',I3)
      ELSE
          MA1=NODE(MSFRL(M33))
          MA2=NODE(MSFRL(M44))
          IF (XPOS(MA1) .NE. XPOS(MA2))GO TO 950
          NF1=NFEAT(MAINF(MA1))
          NF2=NFEAT(MAINF(MA2))
C
C**** A T-SLOT FEATURE
C
          IF (NF1 .EQ. NF2 .AND. NF1 .EQ. BLOC) THEN
             WRITE(6,900)M0,N3,N4,M33,M44,MF12(1),MF12(2)
  900        FORMAT(/1X,'A T-SLOT CREATED BY FACES:'/1X,7I3)
             CALL DABASE(NM7,N0,NM7,SLO2,NPR,M0,N3,N4,M33,M44,
     1                MF12(1),MF12(2),NM8)
          GO TO 1000
          ENDIF
  950     WRITE(6,*)'UNCLASSIFIED FEATURE',M33,M44
      ENDIF
 1000 CONTINUE
      RETURN
      END
C
C*****************************************************************
C     SUBROUTINE CHEKGROV EXAMINES IF A FEATURE IS A GROOVE      *
C*****************************************************************
      SUBROUTINE CHEKGROV(M,MRT)
      DIMENSION MAINF(99),MSFRL(99),NFEAT(20)
      COMMON /INFOR/MAINF,MSFRL,NFEAT
      GROV=16
      M1F=MAINF(M)
      NTYP=NFEAT(M1F)
      IF (NTYP .EQ. GROV) THEN
         MRT=GROV
         NFEAT(M1F)=-NTYP
      ELSE
         MRT=0
      ENDIF
      RETURN
      END
C
C*****************************************************
C**** SUBROUTINE ASINF ASSIGN FEATURE NUMBER TO MAINF *
C*****************************************************
C
      SUBROUTINE ASINF(NM,NS,NFN,NP,NSR)
      DIMENSION NSR(9),MAINF(99),MSFRL(99),NODE(300),LINK(300)
      DIMENSION NFEAT(20),EXAMF(99),NMANF(20)
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
C
C**** NM IS A THE NUMBER OF SURFACES
C**** NS IS A THE NUMBER OF SIDE SURFACES
C**** NP IS A FLAG FOR UPDATING THE MAINF, NP=1 FOR SECONDARY
```

```
C**** FEATURES
C**** NFN IS THE FEATURE NUMBER
C
      DO 100 I=1,NM
      M=NSR(I)
      ND=MAINF(M)
      IF (NP .EQ. 0) THEN
          IF (ND .EQ. 0) THEN
              MAINF(M)=NFN
          ELSE
              NFEAT(ND)=-NFEAT(ND)
          ENDIF
      ENDIF
      MAINF(M)=NFN
  100 EXAMF(M)=1
      IF (NS .EQ. 0)RETURN
      DO 200 I=2,NS+1
      CALL GETNOD(N)
      M=NSR(I)
      ND=MSFRL(M)
      IF (ND .EQ. 0) THEN
          MSFRL(M)=N
          NODE(N)=NSR(1)
      ELSE
  150     IF (ND .EQ. 0) THEN
              LINK(NND)=N
              NODE(N)=NSR(1)
          ELSE
              NND=ND
              ND=LINK(NND)
              GO TO 150
          ENDIF
      ENDIF
  200 CONTINUE
      RETURN
      END
C
C*************************************************************
C**** SUBROUTINE GETNOD RETURN AN AVAILABLE NODE NUMBER      *
C*************************************************************
C
      SUBROUTINE GETNOD(N)
      INTEGER AVAIL
      DIMENSION MAINF(99),MSFRL(99),NODE(300),LINK(300),NFEAT(20)
      DIMENSION EXAMF(99),NMANF(20)
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
C
      AVAIL=AVAIL+1
      IF (AVAIL .GT. 300) THEN
          WRITE(6,100)
  100     FORMAT(1X,'NO AVAILABLE NODE')
          STOP
      ENDIF
      N=AVAIL
      NODE(AVAIL)=0
      LINK(AVAIL)=0
      RETURN
      END
C
C*********************************************************************
C**   THIS SUBROUTINE DISTINGUISH PAD FEATURE FROM POST FEATURE.    *
C**   IF THE DISTANCE BETWEEN THE KEY SURFACE OF A BLOCK TO THE     *
C**   PLANE SURFACE (KL) IS LESS THAN ONE INCH, THE BLOC IS         *
C**   IDENTIFIED AS A PAD FEATURE.                                  *
C*********************************************************************
      SUBROUTINE PAD(KL,N,KP,BG,XPOS)
      INTEGER BLOC,PAD1,BG(3),BF(3),POST
      DIMENSION MAINF(99),MSFRL(99),NODE(300),LINK(300),NFEAT(20)
      DIMENSION EXAMF(99),NMANF(20),XPOS(99),NSR(20)
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
      DATA BLOC/23/,PAD1/11/,POST/25/
C
```

```
      DO 100 I=1,N
      M=BG(I)
      CALL FACEN(KL,KFS,M,DDD)
      BF(I)=KFS
      NG=MSFRL(KFS)
      MF=NODE(NG)
      NFN=MAINF(MF)
      IF (NFEAT(NFN) .EQ. BLOC)GO TO 50
      KP=0
      GO TO 300
   50 IF (I .EQ. 1) THEN
         PREVF=MF
      ELSE
         IF (PREVF .NE. MF) THEN
            KP=0
            RETURN
         ENDIF
      ENDIF
  100 CONTINUE
C
C**   FIND THE DISTANCE BETWEEN KEY SURFACE AND THE TOP OF THE BLOC
C
      A=XPOS(KL)
      B=XPOS(MF)
      C=ABS(A-B)
      IF (C .LE. 1) THEN
          NFEAT(NFN)=PAD1
          CALL READ21(NFN,NUNUM,NN,NSR(1))
          CALL WRIT21(NFN,PAD1,NN,NSR(1))
      ELSE
          NFEAT(NFN)=POST
          CALL READ21(NFN,NFNUM,NN,NSR(1))
          CALL WRIT21(NFN,POST,NN,NSR(1))
      ENDIF
      DO 200 I=1,N
         NF=BF(I)
  200 EXAMF(NF)=1
      KP=KFS
  300 RETURN
      END
C*****************************************************
C**** SUBROUTINE FACEN FINDS THE SHARING FACE NUMBER  *
C*****************************************************
C
      SUBROUTINE FACEN(JJ,KK,NEG,DDD)
C
C**   JJ  : KEY FACE NUMBER
C**   KK  : SHARING FACE NUMBER
C**   NEG : THE EDGE NAME
C**   DDD : THE RADIUS, IF USED
C
      INTEGER XFACE(35),YFACE(35),ZFACE(35),NFTYP(99),NETYP(99)
      INTEGER XCYL(35),YCYL(35),ZCYL(35)
      DIMENSION NOEG(99),DATA(99)
      COMMON /INOUT/IW,NFTYP,NETYP
      COMMON /INFOF/XFACE,YFACE,ZFACE,XCYL,YCYL,ZCYL
      IF (NEG .GT. 0)GO TO 150
      MM=JJ
      READ(9,REC=MM,FMT=50) (NOEG(I),I=1,12)
   50 FORMAT(55X,12I3)
      DO 100 I=1,12
      K=NOEG(I)
      IF (NETYP(K) .GT. 2000)GO TO 120
  100 CONTINUE
  120 NEG=K
  150 READ(10,REC=NEG,FMT=200)IF1,IF2,ILIN,(J,DATA(J),J=1,6)
  200 FORMAT(18X,2I3,I6,6(I2,F9.5))
      IF (JJ .EQ. IF1) THEN
         KK=IF2
      ELSE
         KK=IF1
      END IF
```

```
      READ(9,REC=KK,FMT=300)DDD
  300 FORMAT(40X,F9.5)
      RETURN
      END
C*****************************************************************
C**** SUBROUTINE DABASE TRANSFORM DATA FOR SUBROUTINE DABASE1    *
C*****************************************************************
C
      SUBROUTINE DABASE(NM,NS,NN,NP,NR,N1,N2,N3,N4,N5,N6,N7,N8,N9)
C
      DIMENSION NSR(9)
C
C**** NM IS THE MAIN FACE NUMBER TO BE UPDATED
C**** NS IS THE NUBER OF SIDE FACE TO BE MARKED
C**** NN IS THE NUMBER OF FACES ASSOCIATED WITH THIS FEATURE
C**** NP FEATURE TYPE NUMBER
C**** NR FLAG FOR UPDATING THE MAIN FEATURE
C
C
      NSR(1)=N1
      NSR(2)=N2
      NSR(3)=N3
      NSR(4)=N4
      NSR(5)=N5
      NSR(6)=N6
      NSR(7)=N7
      NSR(8)=N8
      IF (NP .EQ. 3)NSR(9)=N9
C
      CALL DABAS1(NM,NS,NN,NP,NR,NSR(1))
C
      RETURN
      END
C
C*****************************************************************
C**** SUBROUTINE DABAS1 UPDATE THE FEATURE DATA BASE             *
C*****************************************************************
C
      SUBROUTINE DABAS1(NM,NS,NN,NP,NR,NSR)
C
      DIMENSION MAINF(99),MSFRL(99),NFEAT(20),NODE(300),LINK(300)
      DIMENSION EXAMF(99),NSR(20),NMANF(20)
C
C**** NM IS THE MAIN FACE NUMBER TO BE UPDATED
C**** NS IS THE NUBER OF SIDE FACE TO BE MARKED
C**** NN IS THE NUMBER OF FACES ASSOCIATED WITH THIS FEATURE
C**** NP FEATURE TYPE NUMBER
C**** NR FLAG FOR UPDATING THE MAIN FEATURE
C
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
C
C
      IF (NR .NE. 0) THEN
         NFNUM=NR
      ELSE
         CALL GETBLN(NP,NFNUM)
      ENDIF
C
C**** UPDATE THE FEATURE FILE
C
      CALL WRIT21(NFNUM,NP,NN,NSR(1))
C
C**** NFUM IS THE FEATURE NUMBER
C**** NFEAT(NFNUM) STORES THE FEATURE TYPE OF NFNUM
C**** NMANF(NFNUM) FIND THE MAIN FACE OF FEATURE NFNUM
C
  200 CALL ASINF(NM,NS,NFNUM,NR,NSR(1))
      NFEAT(NFNUM)=NP
      NMANF(NFNUM)=N1
C
      RETURN
      END
```

```
C
C**** SUBROUTINE GETBLN UPDATE THE FEATURE NUMBER
C**** AND THE NUMBER OF FEATURE IN A FEATURE TYPE
C
      SUBROUTINE GETBLN(NP,NFNUM)
C
      INTEGER DIR(100),DATA(100)
      DATA NN3/3/,NN4/4/,NN1/1/
C
      CALL READ20(NN3,NFNUM)
      CALL READ20(NN4,NFD)
C
C**   NFD IS THE AVAILBLE FIELD NUBER IN RECORD 3
C
  100 FORMAT(6I4)
      READ(20,REC=2,FMT=200) (DIR(J),J=1,100)
      READ(20,REC=3,FMT=200) (DATA(J),J=1,100)
C
C**   DIR(I) STORES THE FIELD NUMBER OF RECORD 3,
C**   WHERE I IS THE FEATURE TYPE NUMBER.
C**   DATA(2N-1) STORES THE FEATURE NUMBER AND
C**   DATA(2N) STORES THE NEXT FIELD NUMBER WHICH STORES
C**   A FEATURE NUMBER OF THE SAME FEATURE TYPE.
C
  200 FORMAT(100I2)
      N=DIR(NP)
      IF (N .GT. 0) THEN
  250    M=N+1
         N=DATA(M)
         IF (N .GT. 0)GO TO 250
         DATA(M)=NFD
         DATA(NFD)=NFNUM
      ELSE
         DIR(NP)=NFD
         DATA(NFD)=NFNUM
      ENDIF
      DATA(NFD+1)=0
      NFM=NFNUM+1
      NFD=NFD+2
      CALL WRIT20(NN3,NFM)
      CALL WRIT20(NN4,NFD)
      WRITE(20,REC=2,FMT=200) (DIR(J),J=1,100)
      WRITE(20,REC=3,FMT=200) (DATA(J),J=1,100)
      RETURN
C
  300 WRITE(6,400)
  400 FORMAT(1X,'FILE DATA ERROR IN SUBROUTINE GETBLN')
      STOP
      END
C*****************************************************************
C     THIS SUBROUTINE BUILDS THE INTERNAL EXTERNAL RELATION FILE*
C*****************************************************************
      SUBROUTINE INTERF
      INTEGER FET,BLOC,PAD,POST,PLAN
      DIMENSION MAINF(99),MSFRL(99),NFEAT(20),NODE(300),LINK(300)
      DIMENSION EXAMF(99),NMANF(20),XPOS(99)
      DIMENSION NSR(20),NFS(20),NOF(20),INFOF(50),LL(4)
      COMMON /INOUT/IW,NFTYP,NETYP,XPOS
      COMMON /INFOF/XFACE,YFACE,ZFACE,XCYL,YCYL,ZCYL
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
C
      DATA NN5/5/,NN0/0/,NN3/3/,PAD/11/,BLOC/23/,PLAN/18/
      DATA POST/25/
C
C**   INITIALIZE INFOF
C
      DO 100 I=1,50
  100 INFOF(I)=-1
C
C**  GET THE NUMBER OF EXTERNAL RECORDS (NN)
C
      CALL READ20(NN5,NRC)
```

```
      NN=NRC-1
      IOF=1
C
      DO 400 JJ=1,NN
          CALL READ22(JJ,NFN,NK,KL,NSR(1))
C
C     JJ : INTERNAL RECORD NUMBER
C     NFN: FEATURE NAME
C     NK : NUMBER OF SURFACE
C     KL : KEY SURFACE
C     NSR: SIDE SURFACE
C
          DO 300 KK=1,NK
             NM=NSR(KK)
             IF (MAINF(NM) .LT. 1) THEN
                 NODN=MSFRL(NM)
                 IF (NODN .LT. 1) THEN
                     WRITE(6,*)'UNRECOGNIZE FEATURE IN INTERF'
                 ELSE
                     NM=NODE(NODN)
                 ENDIF
             ENDIF
C
C**          A PAD OR A POST (INSIDE A LOOP)
C
             NF=MAINF(NM)
             IF (NFEAT(NF) .EQ. BLOC) THEN
                 A=ABS(XPOS(KL)-XPOS(NM))
                 IF (A .GT. 1.0) THEN
                     NFEAT(NF)=POST
                 ELSE
                     NFEAT(NF)=PAD
                 ENDIF
                 CALL READ21(NF,NFNUM,NN,NFS(1))
                 CALL WRIT21(NF,NFEAT(NF),NN,NFS(1))

             ENDIF
C
C**   REPLACE SIDESURFACE NUMBER WITH INTERNAL FEATURE NUMBER
C
             NSR(KK)=NF
             CALL INFLIN(NFN,NF,IOF,INFOF(1))
  300     CONTINUE
          CALL WRIT22(JJ,NFN,NK,KL,NSR(1))
C
C     NOF STORE EXTERNAL SURFACE NAMES
C
          NOF(JJ)=KL
  400 CONTINUE
C
C**   STORING EXTERNAL FEATURESG
C
      IOF=0
      DO 500 JJ=1,NN
         NM=NOF(JJ)
         NF=MAINF(NM)
         CALL INFLIN(JJ,NF,IOF,INFOF(1))
  500 CONTINUE
C
C**   WRITE OUT THE DATA TO PERMANENT FILE
C
      CALL READ20(NN3,NFT)
      NFT=NFT-1
      DO 700 NK=1,NFT
         DO 550 I=1,4
  550    LL(I)=0
         LL1=INFOF(NK)
         IF (NFEAT(NK) .EQ. BLOC) THEN
             NFEAT(NK)=PLAN
             CALL READ21(NK,NFNUM,NN,NFS(1))
             CALL WRIT21(NK,NFEAT(NK),NN,NFS(1))
         ENDIF
```

```
         IF (LL1 .LT. 0) THEN
               IF (NFEAT(NK) .GT. 49 .OR. NFEAT(NK) .LT. 0) THEN
                     LL(2)=-1
                     LL(1)=-1
               ELSE
C**                  SET THE FEATURE AS AN EXTERNAL FEATURE
                     INFOF(NK)=0
               ENDIF
         ELSE
            LL(1)=LL1
            LL(2)=NODE(LL1)
            LL2=LINK(LL1)
            LL(3)=NODE(LL2)
            IF (LINK(LL2) .GT. 0) THEN
               LL3=LINK(LL2)
               LL(4)=NODE(LL3)
            ENDIF
         ENDIF
C
C**   WRITE INFORMATION ON INTERNAL-EXTERNAL RELATION FILE
C
         WRITE(24,REC=NK,FMT=600)LL(2),LL(3),LL(4)
  600    FORMAT(I6,I7,I7)
C
C**   WRITE THE FINAL RESULT ON THE SCREEN
C
         IF (LL(1) .GT. 0) THEN
            IF (LL(2) .EQ. 1) THEN
               FET=NFEAT(NK)
               WRITE(6,610)NK,FET,LL(3),LL(4)
  610          FORMAT(/1X,'FEATURE #',I3,' HAS THE FEATURE TYPE',
     1         I3,' AND IS AN INTERNAL FEATURE OF',I3,' AND',I3)
            ELSE IF (LL(2) .EQ. 2) THEN
               FET=NFEAT(NK)
               WRITE(6,610)NK,FET,LL(4)
               NRC=LL(3)
               CALL READ22(NRC,NFN,KK,KL,NFS(1))
               FET=NFEAT(NFN)
               WRITE(6,620)NK,FET,(NFS(I),I=1,KK)
  620          FORMAT(/1X,'FEATURE #',I3,' WITH THE FEATURE TYPE',
     1         /I3,1X,'IS AN EXTERNAL FEATURE AND CONTAINS THE',
     2         ' INTERNAL FEATURES',10I3
            ELSE IF (LL(2) .EQ. 0) THEN
               NRC=LL(3)
               CALL READ22(NRC,NFN,KK,KL,NFS(1))
               FET=NFEAT(NK)
               WRITE(6,620)NK,FET,(NFS(I),I=1,KK)
            ELSE
            ENDIF
         ELSE
            IF (LL(2) .GT. -1)WRITE(6,630)NK,NFEAT(NK)
  630       FORMAT(/1X,'FEATURE #',I3,' IS AN EXTERNAL FEATURE',
     1      1X,'HAVING THE FEATURE TYPE',I3)
         ENDIF
  700 CONTINUE
      RETURN
      END
C*************************************************************
C**   THIS SUBROUTINE SETS UP THE INTERNAL RELATION AMONG    *
C**   FEATURES.                                              *
C*************************************************************
      SUBROUTINE INFLIN(JJ,NF,IOF,INFOF)
C
C**   JJ  : EXTERNAL FEATURE NUMBER IF IOF EQUALS TO 1
C**         INTERNAL FEATURE FILE RECORD NUMBER IF IOF IS 0
C**   NF  : FEATURE NUMBER
C**   IOF : FLAG, 1 STANDS FOR INTERNAL FEATURE AND
C**               0 STANDS FOR EXTERNAL FEATURE
C**   INFOF: STARTING NODES OF THE ON CORE RELATION FILE
C
      DIMENSION MAINF(99),MSFRL(99),NFEAT(20),NODE(300),LINK(300)
      DIMENSION EXAMF(99),NSR(20),NMANF(20),INFOF(50)
```

```
      COMMON /INFOR/MAINF,MSFRL,NFEAT,NODE,LINK,EXAMF,NMANF
C
      CALL GETNOD(N)
      NY=INFOF(NF)
C
C**   STORE THE FEATURE RELATION INFORMATION
C
      IF (NY .LT. 0) THEN
          INFOF(NF)=N
          NODE(N)=IOF
          CALL GETNOD(M)
          LINK(N)=M
          NODE(M)=JJ
          LINK(M)=0
      ELSE
C
C**   INTERNAL OF TWO EXTERNAL FEATURES OR
C**   INTERNAL OF ANOTHER INTERNAL FEATURE
C
        NY1=LINK(NY)
        NY2=LINK(NY1)
        IF (NY2 .GT. 0) THEN
          WRITE(6,*)'IMPOSSIBLE RELATION'
        ELSE
          IF (IOF .EQ. 1) THEN
             LINK(NY1)=N
          ELSE
             NODE(NY)=2
             LINK(N)=LINK(NY)
             LINK(NY)=N
             NODE(N)=JJ
          ENDIF
        NODE(N)=JJ
        ENDIF
      ENDIF
      RETURN
      END
```

VITA

Hsiang-Kuan Kung

Candidate for the Degree of

Doctor of Philosophy

Thesis: AN INVESTIGATION INTO THE DEVELOPMENT OF PROCESS PLANS FROM SOLID GEOMETRIC MODELING REPRESENTATION

Major Field: Industrial Engineering and Management

Biographical:

Personal Data: Born in Taipei, Taiwan, R.O.C., May 8, 1949, the son of Mr. and Mrs. Li-Hwa Kung.

Education: Graduated from Hsin-Tsu High School, Taiwan, R.O.C., in May 1967; received Bachelor of Science degree in Industrial Engineering from Chung-Yuan Christian College of Science and Engineering, Taiwan, Republic of China, in June 1972; received Master of Science degree in Industrial Engineering from University of Rhode Island, in December 1978; completed requirements for the Doctor of Philosophy degree at Oklahoma State University in December, 1984.

Professional Experience: Production Planning Technician, Taiwan Material Agency, Taichung, Taiwan, May, 1974, to August, 1975; Quality Control Director, Sam-Min Food Industry Inc., September, 1975, to June, 1976; Graduate Assistant, University of Rhode Island, September, 1976, to June, 1978; Graduate Teaching and Research Assistant, Oklahoma State University, September, 1978, to May, 1984.

Professional Membership: American Institute of Industrial Engineering, Alpha Pi Mu.