Spring 1995

An investigation into artificial intelligence based generative computer process planning

William Tereshkovich
New Jersey Institute of Technology

ABSTRACT

A INVESTIGATION INTO ARTIFICIAL INTELLIGENCE BASED GENERATIVE COMPUTER PROCESS PLANNING

by
William Tereshkovich

The process planning function can be further optimized with the introduction of an intelligent software package such as an Artificial Intelligence based Generative Computer Process Planning System. An AI based CAPP system, also categorized as an expert system, provides the vital link between design and manufacture, a "bridge" of knowledge in the information chain of manufacturing engineering. With the increasing role of computers in the process planning function, planning has become easier, faster, and more efficient.

This thesis provides an explanation of the process planning function and how artificial intelligence will improve it. Topics in AI techniques and procedures, knowledge engineering, Computer Aided Design, and Group Technology (GT) are discussed. Utilizing the topics presented, a complete interactive software system is presented to illustrate the flexibility of an AI based CAPP system.

A INVESTIGATION INTO ARTIFICIAL INTELLIGENCE BASED GENERATIVE COMPUTER PROCESS PLANNING

by
William Tereshkovich

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Manufacturing Systems Engineering

Manufacturing Engineering Division

May 1995

APPROVAL PAGE

A INVESTIGATION INTO ARTIFICIAL INTELLIGENCE BASED GENERATIVE COMPUTER PROCESS PLANNING

William Tereshkovich

Steve Kotefski, Thesis Advisor — Date
Professor, Department of Manufacturing
Engineering Technology, NJIT

Dr. Samir Billatos, Committee Member — Date
Professor of Manufacturing Engineering and
Director of Manufacturing Engineering, NJIT

Dr. Nouri Levy, Committee Member — Date
Associate Professor, Department of Mechanical
and Industrial Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: William Tereshkovich

Degree: Master of Science in Manufacturing Systems Engineering

Date: May 1995

Undergraduate and Graduate Education:

- Master of Science in Manufacturing Systems Engineering
  New Jersey Institute of Technology, Newark, NJ, 1995

- Bachelor of Science in Engineering Technology
  New Jersey Institute of Technology, Newark, NJ, 1994

- Associate in Applied Science
  Brookdale Community College, Lincroft, NJ, 1992

Major: Manufacturing Systems Engineering

This thesis is dedicated to my parents, William and Eleanor Tereshkovich, and to my uncle, Roy Pautz, who have supported and encouraged my academic endeavors in every possible way and have been an inspiration for me in pursuing my goals and dreams as a graduate student.

ACKNOWLEDGMENT

The author wishes to express his sincere gratitude to his thesis advisor, Professor Steve Kotefski of the Manufacturing Engineering Technology Department at the New Jersey Institute of Technology for his timely and expert assistance, guidance and encouragement, friendship and support, during this research.

Special thanks to Dr. Samir Billatos and Dr. Nouri Levy for serving as members of the thesis committee.

The author would also like to thank James and Velma Emmi of the EMMI Foundation for financial assistance during this research.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

CHAPTER 1

THE PROCESS PLANNING FUNCTION

One of the most widely used manufacturing techniques in industry today is process planning. The process planning function represents the vital link between design and manufacture, a "bridge" of knowledge in the information chain of manufacturing engineering. It involves the preparation and documentation of the minimum manufacturing sequences to be completed by work centers, and describes specific designs and cycle times of resource usage for a product to be manufactured. In addition, process planning (as with the planning of operations layouts and numerical control systems) often utilizes detailed information for a specific purpose. Therefore, process planning could be described as the preparation of manufacturing instructions, sequences, and engineering drawings required for production.

The process plan for a particular work-part is not unique since there is usually more than one way of manufacturing it. A process plan simply details the sequence of operations to produce a work-part. Different process plans may also be required for different lot sizes. The process plan usually consists of several sections. These sections include general information, operation sequences, and assembly instructions.

The first section contains general information that defines the work-part, its version, and any other special conditions relating to its production. For example, engineering drawings describing the work-part or the lot size for which the plan was developed.

The second section of the process plan is made up of the sequences defining the operations required for manufacture.

The final section of the process plan is only required if the work-part is to be unloaded from one machine and loaded onto another for further processing. The third section may also be made up of assembly instructions or special machining instructions.

Without process planning, there can be no performance analysis and control, no effective production scheduling, and overall, no design for manufacture. The preparation of a process plan is the translation of engineering requirements (contained in engineering drawings and design specifications) into detailed technical manufacturing requirements of material labor, design for manufacture, and equipment. The process plan carries information that is traditionally recorded and distributed to other activities within an organization using laborious technical and clerical techniques. These laborious and time consuming techniques are usually applied by professional process planners within an organization. Upon the use of process plans, receiving departments have to manipulate the data as input into their own systems, thus the process planning function requires highly skilled planners and a large amount of costly time.

Automating the process planning function is an obvious alternative to alleviate the amount of time and experience it requires.

1.1 Traditional/Manual Process Planning

Traditionally, the process planning function has been carried out by specialized technicians with exceptional knowledge in manufacturing techniques, machine design capability, company machinery and equipment, and level of plant skills among workers. Manual process planning involves the visualization and analysis of production sequences and alternative operations. Clerical techniques such as drafting and calculation also play an important role. To be effective, process planners must utilize various types of aids to achieve the lowest unit cost consistent with all engineering requirements. Personally acquired manufacturing logic must be applied manually when data is received from all engineering specifications. This is usually dependent upon the process planners ability to recall past experiences and current planning techniques to apply knowledge efficiently and effectively.

Standard data is applied by process planners to describe cost comparison of alternative methods and operations to arrive at target manufacturing deadlines. The standards may be information relating to estimates based on past data through to complex manufacturing standards for both design and processes.

With manual process planning, the process planner is faced with considerable decision making, data retrieval, and calculation during the planning cycle. Valuable time is wasted due to laborious standard clerical techniques. Costly errors can also occur when design data is interpreted incorrectly.

Manual process planning involves a time consuming routine in document preparation. A process planner must first study and evaluate the overall geometry of a work-part. This information will be utilized to classify the part in order to determine the proper production sequence. The planner must then study the engineering drawing to identify all the production features. The production features will be used to determine the best raw material shape. Finally, the planner must be able to identify the datum surfaces, off the engineering drawing, and use the information to determine setups. With this data collected, the planners sequence could be to:

- Determine the work-centers to be used in a particular operation.
- Determine a rough sequence of operations for production.
- Determine if any modification is to be made with the operations sequence.
- Select tooling.
- Select fixtures, jigs, etc.

- Determine the cutting parameters for each operation.
- Provide a process plan in finished form.

The time spent during the process planning cycle can be broken down into percentages. The percentages illustrate the time a standard set of plans may take by an experienced planner. The percentage of time to prepare a set of process plans is:

- Fifteen percent for technical decision making and planning strategy.
- Forty percent for data retrieval and calculation.
- Forty-five percent for text and other document preparation. This constitutes engineering drawing, bill of materials, production planning, etc.

These percentages are typical across all types of engineering organizations, from batch production to high production volume. The percentages also indicate that traditional process planners are continually attempting to create and update manufacturing information. Although a planner may be able to make quick decisions, data retrieval, document preparation, and calculation, manual process planning is time consuming and is usually prone to error. This presents the planner with considerable communication problems within the organization.

1.1.1 Disadvantages of Manual Process Planning

Most of the disadvantages stemming from the traditional approach are caused primarily from the very high dependence upon technical and clerical activities. The activities include both preparation of the process plans to planning the engineers need to interact with related activities. This is compounded by the lack of communication between planning departments. More specifically, the main disadvantages of manual process planning include:

- Manufacturing logic is individual - meaning it usually resides in the process planners mind. This has a percentage of error.
- The individual process logic has to be recalled and re-processed for every plan. This becomes a laborious and time consuming process that is often relied on by guess-work.
- The process plan results are usually incomplete and inconsistent, resulting in a high margin of error.
- There is a extension of pre-production leads times due to data retrieval and update.
- The estimation for new products or processes are suspect until proven correct.
- Manufacturing skills are locked in clerical planning routines rather than used on methods of improvement and cost reduction programs.
- Supervision and other operatives have little respect for the targets set by the process plans due to inconsistency and inaccuracy.
- The process planning information is often out of date. The entire planning cycle is subject to frequent

information updates such as engineering changes to product design, alternative materials and processes, improved manufacturing processes, and changes to quantities.

With manual process planning, there is a present structure that can only operate with highly skilled planners which have superior skills as an operator and a "up to date" knowledge of manufacturing design and processes.

The efficient and effective way to process the design and planning activities is through automation. Using an artificial intelligence (AI) based expert system, planning activities will become more accurate and timely. Engineering data can be quickly retrieved and updated. Data storage will become easier and communication will be optimized.

1.2 Automated Process Planning

Automated process planning can be broadly categorized as being variant or generative. These two approaches are used for the design of expert systems and utilize computer processing abilities. The generic term, computer assisted process planning (CAPP), is also used to describe automated process planning. Automated process planning systems do not require the planner to have expert knowledge of manufacturing design and processes. CAPP systems use pre-programmed engineering knowledge, utilizing artificial intelligence, to automatically generate accurate and effective process plans from the preliminary data that is given.

1.2.1 Variant Computer Process Planning

A variant process planning system could be referred to as a data retrieval system. With this approach, a set of standard plans is established and maintained for a particular work-piece. Expert knowledge is pre-programmed into a computer system using a database to store all information. The information may be retrieved and updated to accommodate design changes, manufacturing changes, or new developments in technology, (See figure 1 on page 9). Retrieval of information is maintained using a classification and coding scheme as used in group technology (GT). The general specifications for data modification and retrieval are:

- Establishment of a coding scheme using group technology concepts.
- The formation of part families using group technology concepts.
- The development of standard process plans. Process plans are generally designed to a specific operation.
- The retrieval and modification of standard process plans for new work-parts.

In order to utilize these specifications, some basic functional modules are required in the development of a variant CAPP system.

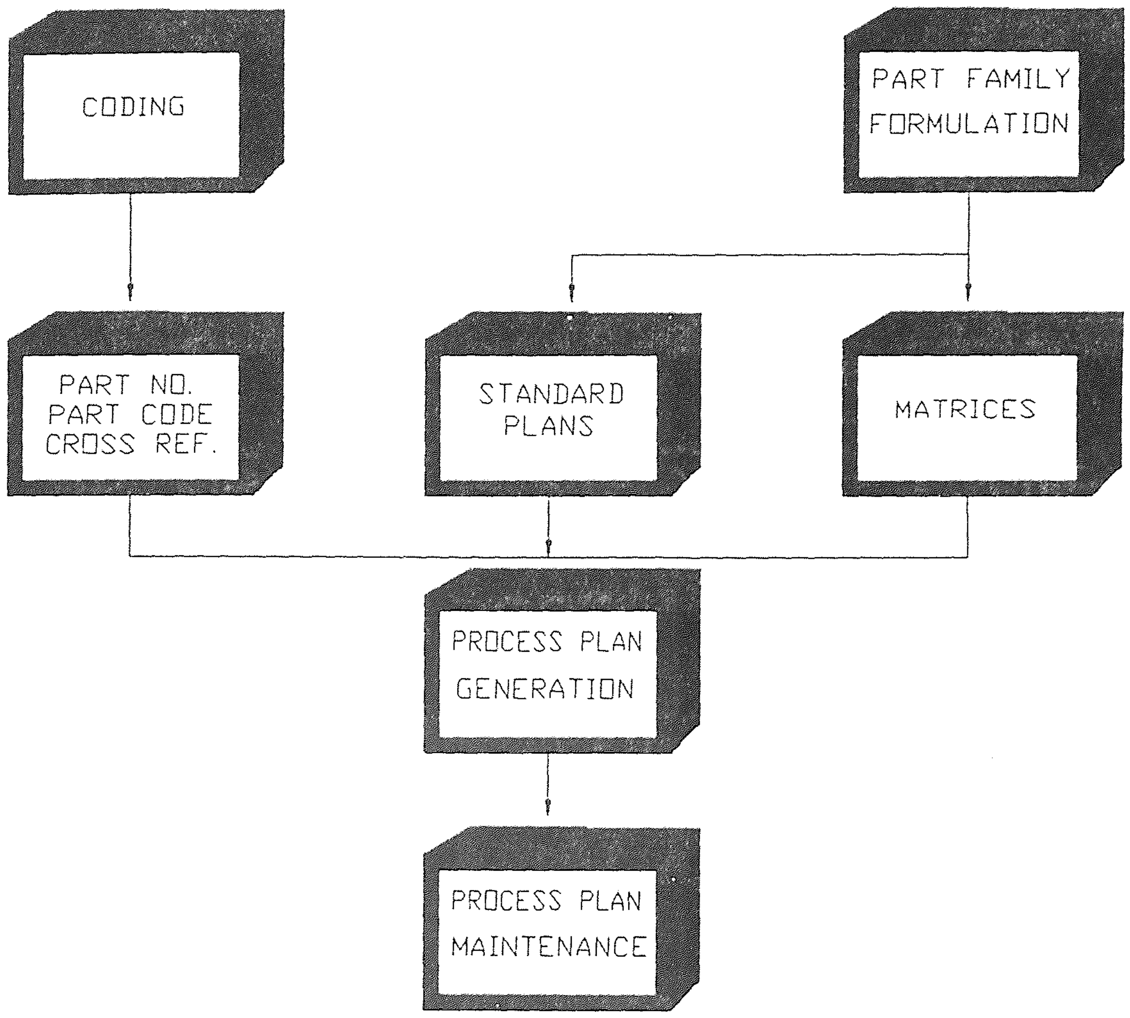


Figure 1. Variant Process Planning

1.2.1.1 Establishment of a Coding Scheme

A major step in developing a variant CAPP system is to select and establish a coding scheme. The selection relies heavily on the product that the company manufactures. The coding scheme could either be unique or general.

1.2.1.2 Search Procedure for Variant Systems

The main function of a variant CAPP system is to retrieve process plans for similar components to be put into production. In doing this, a search procedure must take place where parts can be identified in relation to a particular family. When the part family is located, the process plan can be easily retrieved.

A family matrix search may be used to match entered code with information in the systems database. Family matrices are also known as masks. When the entered code passes through a mask, a particular family is identified. A search procedure as described by Chang, Tien-Chen, Wysk, A. Richard and Hsu-Pin Wang, for variant systems can be evaluated as the following:

- Let C_j be a value of code position j for the given component.
- $P^l{}_n$ is a pointer for family matrix l, which links to the next family matrix.
- $P^l{}_s$ is a pointer for family matrix l, which links to the directory of the standard plan.
- $P^l{}_{ij}$ is the content of family matrix l; when it equals

one, code position j is allowed to have the value i.
With having all the variables defined, standard process plans may be located using the following algorithm.

```
Step 1. For all, do step 2. End stop.
Step 2. For j = 1 to J, do step 3; end, goto step 5.
Step 3. i = $C_j$; if $P^l_{ij}$ <> 0, end step; otherwise,
Step 4. l = $P^l_n$; goto step 2.
Step 5. Standard process plan found; $P^l_s$ is the
        pointer to the standard process plan. End
        search process.
```

Another way to perform a matrix search, let C*j equal a range of values instead of just a single value. The algorithm above can then be changed in step 3. Step 3 will look like this:

$\sum P^l_{ij} \neq 0$, end step; otherwise, next j.

1.2.1.3 The Formation of Part Families

Collecting similar parts by geometry, surface features, or fabrication process similarities is the backbone of variant process planning. This insures the proper formation of a part family. Since all similar parts are assigned to a specific part family, a database may be used to store all necessary information by utilizing a random access data structure, thus creating a variant system for retrieving data for a process plan.

General Methods for forming part families include:

- Production flow analysis.
- Cluster analysis.
- Machine loading analysis.

1.2.1.4 Development of Standard Plans

A standard plan in a part family is a collection of common operations used to fabricate work-parts belonging to a particular part family. The capabilities of generating, retrieving, and updating standard plans are the basic requirements of a variant CAPP system.

In order to make a variant CAPP system functional, part data for a standard plan should be retrievable in several ways. These include:

- The part data may be retrieved using part codes and specifications, part numbers, and part family numbers.
- A cross reference module used to obtain members in tabular format.
- A relational database structure.

1.2.1.5 Retrieval and Modification

Since variant process planning is primarily a retrieval/update method, there is the need to design a procedure through which the system planner can retrieve fixed data and modify it for a standard work-part. This type of design should be informative, useful, and user-friendly.

1.2.2 Generative Computer Process Planning

Generative computer process planning is a system designed to automatically synthesize process information to create design specifications and process plans for a new component. Decision logic and optimization are encoded into the system to create a type of artificial intelligence. Using all data that is entered, a generative system will produce a complete process plan beginning with design data, in the form of an engineering drawing, to the requirements of the manufacturing process. A generative system also requires detailed mathematical techniques for the combinational programming aspects, (See figure 2 on page 14).

1.2.2.1 Part Descriptions

A central requirement for a generative CAPP system is a part description. Geometry, dimensioning, and surface quality requirements are to be defined by the planner. The entered data will be stored into memory, where it will be processed and utilized to generate useful process information.

1.2.2.2 Coding Methods

As with variant systems, some generative systems utilize coding methods for part description. Various group technology concepts are used to create a database where information can be stored and retrieved.

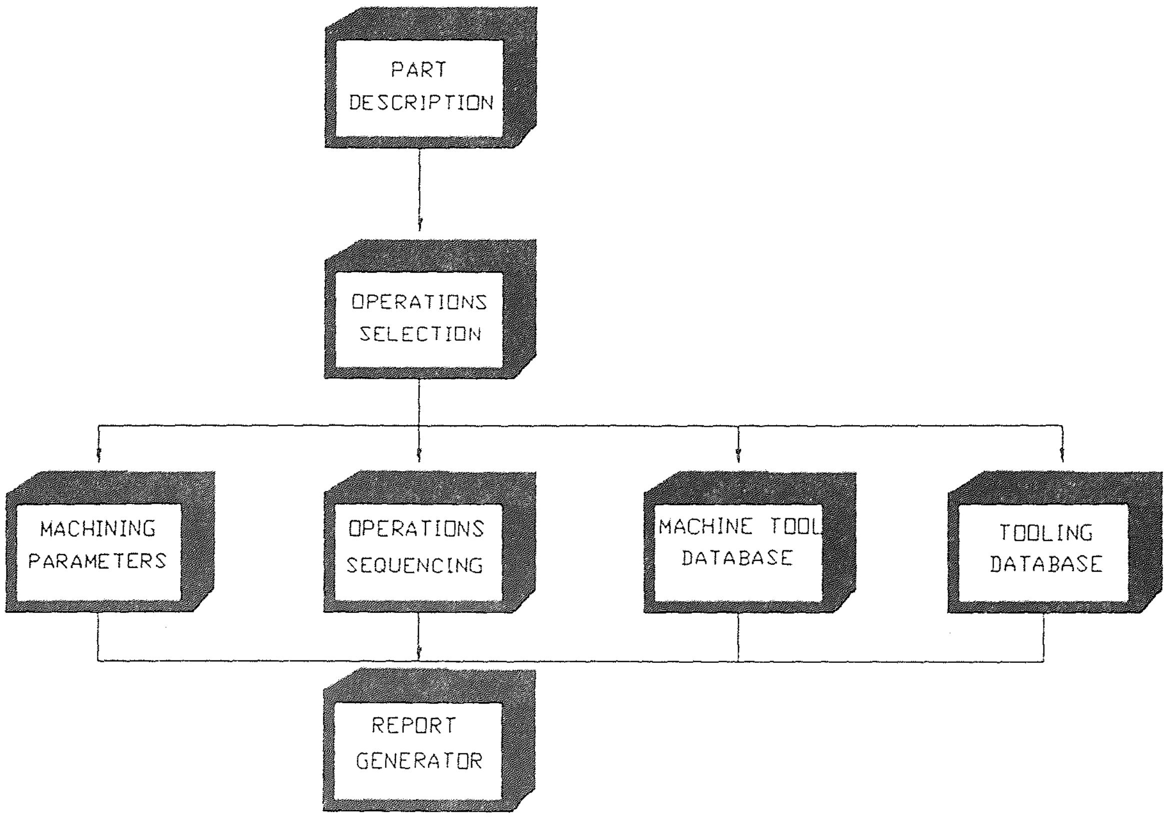


Figure 2. Generative Process Planning

1.2.2.3 Language Methods

Most generative systems utilize a specially designed language syntax to describe components and specifications. For this type of system, the syntax and semantics of commands are defined within a database structure in order for a component description to be given by the planner. Language methods are a preferred alternative to group technology.

1.2.2.4 Sequencing

Sequencing plays a vital role in generative CAPP systems since expert knowledge is programmed as a decision tree. An IF-THEN sequence usually simulates the decision process, thus sequencing routines can simulate an expert decision.

1.2.2.5 Tooling Database

A generative system generally contains a database of tooling specifications, such as jigs, fixtures, and clamping devices. The database can either be utilized by the operator or the program itself. Machine tools, used to complete a particular operation, must also be encoded into a generative system.

1.2.2.6 Generation of a Report

A main requirement of any CAPP system is the generation of a process plan in hard copy format. The generated report must be legible and easy to understand to manufacturing personnel. The report should be informative, containing all the basic data such as part identification, machining operations, machine identification, and tooling. Another basic requirement of a report generator is the generation of an engineering drawing. The drawing will represent the design specifications that are required and can be created using a plotter or a laser printer.

1.2.2.7 Specifications of a Generative Computer Process Planning System

Considering the importance of the process planning function, there has been little research in the way of integrating artificial intelligence with process planning. However, in recent years, the utilization of generative CAPP systems has been growing. A particular system could include an expert system being designed to produce process plans for cylindrical components. The system also processes the size of the sequencing and machining operations. An example of the amount of sequences this type of computer system must make could be:

> A component to be manufactured consists of X number of independent and different features such as surface texture, holes, and fillets. Each of these features may be machined in any sequence and each may be machined on any of Y machines. The total number of different sequencing possibilities, defined by Z, for manufacturing the component can be given by:

$$Z = (X!)(Y^X) \qquad (1)$$

The different number of possibilities equals the number of different features factorial multiplied by the number of machines to the power of different machines. Thus, for a simple part of 8 holes which is to be manufactured in a shop with 6 drill presses, there will be (40320)(68) or 6.7×10^{10} alternatives.

In utilizing particular specifications for a generative system, decision alternatives such as the previous example must be taken into consideration. Since today's computers can provide quick and accurate processing, utilizing computer processing is almost a must to calculate the best alternative.

A generative expert system should be designed to allow the operator to enter specific data requirements into a computer without having to program it. The major objectives in designing a generative expert system would be to:

- Free the process planner from time consuming clerical activities such as calculation, report generation, and engineering drawing.
- Use the existing company data. Previous data once used in design or processing may be utilized and updated within a generative system.
- Perform all calculations. The generative system should be pre-programmed with all the necessary formulas in order to generate the best possible solution.
- Be applicable to all types of engineering.

- Generate user defined documentation and data files. This would include a graphical representation of a particular geometry, process sheets, or the combination of both. The files must be retrievable for updating.
- Be capable of extension or update by engineers, to reflect changing products.
- Have the ability to interface with other manufacturing systems. This requires that the system can output design files and other data to computer aided design (CAD) systems, computer aided manufacturing (CAM) systems, etc.

The generative method of computer process planning involves the generation of unique plans for a particular process or design. Using this method does not require storage of standard routines and information (as with variant systems). Instead, new plans are generated and tested automatically for feasibility, and then the best alternative is chosen by the expert system's design criteria (such as geometric coding and decision tree logic).

1.2.2.8 Benefits of a Generative Computer Process Planning System

The objective of a generative computer process planning system is to improve the output level of an organization by simplifying the planning and design process. The major benefits of this type of system include:

- Releases an engineer from clerical routines and methods such as calculation and documentation.
- Improves the consistency in development of operation times over a wide product range. Designs become more accurate and process plans become more efficient.
- Provides the speed of response to engineering changes. Since files are stored in the system, retrieval and update can easily be accomplished.
- Provides acceptability of planned targets. Since plans are more accurate, planned manufacturing deadlines can be achieved.
- Makes accurate responses and decisions.
- Reduces pre-production lead times. This is due to the ability of data storage and retrieval. Updating a current design to meet future requirements provides quick pre-production lead times.
- Provides easy access planning data. Data may be used by simply accessing the systems database. If there is a need, the expert data can be used for manual process planning.

CHAPTER 2

KNOWLEDGE ENGINEERING IN GENERATIVE COMPUTER PROCESS PLANNING

There is an extensive amount of knowledge relating to the planning of manufacturing processes that one must utilize in order to make a CAPP system successful. The extraction, classification, refinement, and formalization of information is known as knowledge engineering. Knowledge engineering is not only faced with the extraction of relevant knowledge from the expert, but involves applying the knowledge into practical form.

In the case of an AI based CAPP system, knowledge must be extracted from the expert, such as a engineer or a process planner. The knowledge must be relevant to a particular subject and must convey problem solving techniques. The extracted knowledge is to be encoded into a program where expert simulation occurs.

Knowledge bases are developed by extracting the rules and procedures human experts use in solving problems. Knowledge bases are usually broken down into a IF-THEN format since the program will make pattern matches to solve problems. Because knowledge bases are separate entities from inference engines (decision-making algorithms), the knowledge contained in the knowledge base is easy to modify. Just as in a spreadsheet, where numbers can be changed independently, expert system knowledge bases can be changed just as easily. Changes within the knowledge base simply updates and deletes rules in any order.

2.1 Knowledge Defined

A definition of knowledge could be, as described Frenzel (11), an understanding of a particular field of interest that has been obtained through experience and education. Knowledge is made up of ideas concepts, facts and figures. Knowledge implies learning, awareness, and familiarity with one or more subjects. Knowledge is the key to solving basic to complex problems.

2.2 Knowledge Representation

There are two forms of knowledge that can be broken down and put into a knowledge base. They are declarative knowledge and procedural knowledge. Most artificial intelligence systems will contain both.

2.2.1 Declarative Knowledge

Declarative knowledge is primarily a statement of fact about a certain thing. Declarative knowledge permits the statement of information, deduction of relationships, and the classification of objects. Group technology concepts utilizes declarative knowledge for classification of manufactured parts. In expert systems, declarative knowledge representation schemes include semantic networks, frames and production rules.

2.2.1.1 Semantic Networks

The use of a semantic network is one of the most basic methods to represent knowledge. A semantic network consists of graphically depicting the relationships between objects, events, concepts, situations or actions by a directed graphical representation consisting of nodes and labeled edges. A simple semantic network may look like:

```
MACAW--------------->WINGS
         (HAS-PART)
```

Where MACAW and WINGS are nodes representing sets or concepts and HAS-PART is the name of the link specifying their relationship.

Semantic networks are primarily used to illustrate relationships utilizing declarative knowledge. It is primarily a representation of world aspects by naming and referring objects in a given domain and describing the relationship between them.

2.2.1.2 Frames

A declarative knowledge representation scheme known as a frame is used to describe, in detail, information about a particular object. Frames usually involve the gathering of well known or generalized data. This data is collected and can be placed in discrete elements known as slots. The slots, which describes attributes of an object, can be subdivided into facets where a description of the attribute may be given.

A frame "system" is mainly composed of interrelated frames that are required to represent a specific domain. It relies heavily on the concept of inheritance between one frame to another.

2.2.1.3 Production Rules

In expert systems, a common way of representing heuristic knowledge is with production rules. The rules usually consist of two to three part statements that contain a small increment of knowledge.

The format of a production rule usually is written as an IF-THEN statement or an OR statement. IF usually refers to a situation and THEN refers to an action or a conclusion. OR represents an alternative. A few examples of a rule format are as follows:

1. IF it starts getting dark,
   THEN turn on the light.
2. IF the cube is hard,
   AND is cold,
   AND used to be in the form of water,
   THEN it is an ice cube.
3. IF the test is tomorrow,
   OR the test is the next day,
   THEN I will study tonight.

The major benefits of rules in this type of format is that knowledge is represented in a compact way. Production rules can be easily updated or deleted from an expert system, therefore, changes in a system can be made quickly.

Production systems will provide a model for encoding human expertise in the form of rules and designing pattern-driven search algorithms. Artificial intelligence systems will utilize production rules as a potential model for solving specific problems. This is an ideal method for computer aided process planning systems.

2.2.2 Procedural Knowledge

Procedural knowledge provides a way to apply declarative knowledge. Procedural knowledge recommends what to do and how to do it. A step by step sequence on how to build a computer or instructions for building a model airplane are examples of procedural knowledge. In expert systems, procedural knowledge is represented as scripts and production rules.

2.3 Information Versus Knowledge

There is a major difference between information and knowledge. Information is based primarily on facts and figures in that raw data has not been interpreted. Information can be in the form of equations, engineering drawings, and random data.

Knowledge, on the other hand, is an understanding of information based on analysis and application. Raw data is interpreted and put into useful form.

The difference between knowledge and information is not always obvious. Knowledge involves a wider range than information. Knowledge not only includes information but includes skills, perception, imagination, intuition, and common sense due to experience. It is the sum of perceptive processes organized in such a way that conclusion and solutions may be drawn.

2.4 Knowledge Applied to Artificial Intelligence Systems

Knowledge for an expert system can be obtained from textbook type information. Knowledge used in expert systems can be a fact or a figure and is usually obtained directly from an expert. The type of knowledge incorporated into an expert system could be a policy and procedures manual, a set structure of equations, or a given data base that can be retrieved or updated.

However, in most applications, the kind of knowledge that is best applied to expert systems is heuristic knowledge. Heuristic knowledge is practical real-world understanding. It includes all the strategies and techniques that an expert might use to solve a particular problem. Heuristic knowledge is not textbook oriented, rather it is knowledge that has been obtained through years of experience and exposure to a wide variety of problems. Through heuristic knowledge techniques, an expert can quickly solve problems since there is a set format of procedures. Programmed into an expert system, these procedures and techniques can be utilized to solve a variety of problems. The knowledge format can be automatically applied through given data to simulate the decision making process.

In developing a manufacturing expert system, heuristic knowledge is encoded into a program where it is applied to everyday problems. Encoding is the "packaging" of heuristic knowledge to be used during system operation. The encoded knowledge becomes expert information for a computer to work with. Information such as spreadsheets, data bases, process formulas, and mathematics, are required if the system is to perform "number crunching".

2.5 Knowledge Representation Schemes

In manufacturing, information to be encoded into a knowledge base is usually difficult by using simple structures like arrays or sets of numbers. Representation schemes such as First Order Logic, Semantic Networks, and Frames can be used to represent statements and to indicate how to carry out a variety of actions, such as information retrieval, computation, etc.

First Order Logic involves the relationship between assumptions and conclusions. First Order Logic sentences can be expressed as a collection of clauses. Clauses may be defined as an expression in the following form:

$$D_j, \ldots., D_n \leftarrow C_j, \ldots., C_o, \quad n,o >=0$$

where $C_j, \ldots., C_o$ are conditions of the clause and $D_j, \ldots., D_n$ are alternative conclusions of the clause.

Both conditions and conclusions, called atoms, are expressions of the form:

$$Z(d_j, \ldots., d_l)$$

where Z is a l-argument predicate symbol and $d_j, \ldots., d_l$ are terms.

A term is a variable that is an expression of the form:

$$g(d_j, \ldots., d_m)$$

where g is a m-argument function symbol and $d_j, \ldots., d_m$ are terms.

If a clause contains the variables $y_j, \ldots., y_l$, then it can be concluded as the following statement:

for all $y_j, \ldots., y_l$

D_j or .... or D_n if C_j and .... and C_o

if o=0, then the clause can conclude that:

for all $y_j, \ldots., y_l$

D_j or .... or D_n

if n=0, then the clause can conclude that:

for no $y_j, \ldots., y_l$

C_j and .... and C_o

Using the First Order Logic as defined above, manufacturing schemes can be developed. A few examples of applying this type of knowledge representation scheme could include:

MACH-OP(M3, OP5), meaning that machine three can perform operation five.

IDLE(M3, t3), meaning that machine three is idle at time three.

JOB-LAST-OP(OP3, J5), meaning that operation three is the last operation to be performed on job five.

Semantic Networks, sometimes are known as structured objects because the major emphasis is on the structure of the representation, also plays a vital role in knowledge representation. A semantic network is a graph whose nodes represent individual objects within a system, as explained earlier. Directed arcs, connecting the nodes, represent binary relationships.

The semantic network representation of knowledge is better known as a graphical sequence of events. An example of this could be:

$$\begin{tikzcd} a \arrow[r, "X"'] & b \end{tikzcd}$$

This is a graphical representation of:

X(a, b) <----.

Frames, on the other hand, is a generalization of a property list which provides a structured representation of application in a given domain. A frame provides a mechanism that guides description movement and allows for specification of procedures for computing purposes.

In manufacturing, frames are usually applied to job shop scheduling and project management. In the production planning context, frames represent knowledge about jobs, tasks, and resources, such as machines, tools, pallets, etc.

CHAPTER 3

ARTIFICIAL INTELLIGENCE AND GENERATIVE COMPUTER PROCESS PLANNING

Artificial intelligence is a broad field of computing that involves making computers duplicate the human decision making process. Artificial intelligence could be described as a collection of techniques that allow a computer system to mimic the human thinking process. An AI computer system is a tool to be utilized in solving simple to complex problems. This is an attempt to make machines do specific tasks that require some level of intelligence. Artificial intelligence is simply a useful tool that can be used to make computers more efficient and productive, which in turn makes human jobs more efficient and productive.

The ability to duplicate the human thinking process is desirable since decisions can be made accurately and effectively and in a more timely fashion. Many manual manufacturing operations can be performed faster and more efficient with the use of AI systems. Artificial intelligence takes process planning a step further, the result is a better equipped computer system that can be applied to many different scenarios.

3.1 Types Of Computing

The two basic forms of computing are conventional and artificially intelligent. Both represent programming techniques that require expert knowledge, but process the knowledge in a very different way.

3.1.1 Conventional Computing

In conventional computing, a step-by-step list of instructions, called an algorithm, is given to the computer to perform a particular operation, (See figure 3 on page 32). Conventional computing tends to share two basic ideas. These are:

- The application programmer will develop a step-by-step numbered sequence algorithm in order to solve a problem.
- The data required in conventional computing execution is stored in a data base. Data can then be called by the program as it is needed when each step is executed.

Conventional software relies on a conventional language such as COBOL, FORTRAN, and PASCAL, however, the C++ language is a conventional language being used in the development of artificial intelligence based systems.

Conventional software programs, as described by Frenzel, processes data within nine parameters. These are:

- To store data. The data may be a fact, figure, or formula.
- To retrieve data from files. This is accomplished by coding schemes within a database or file location. The

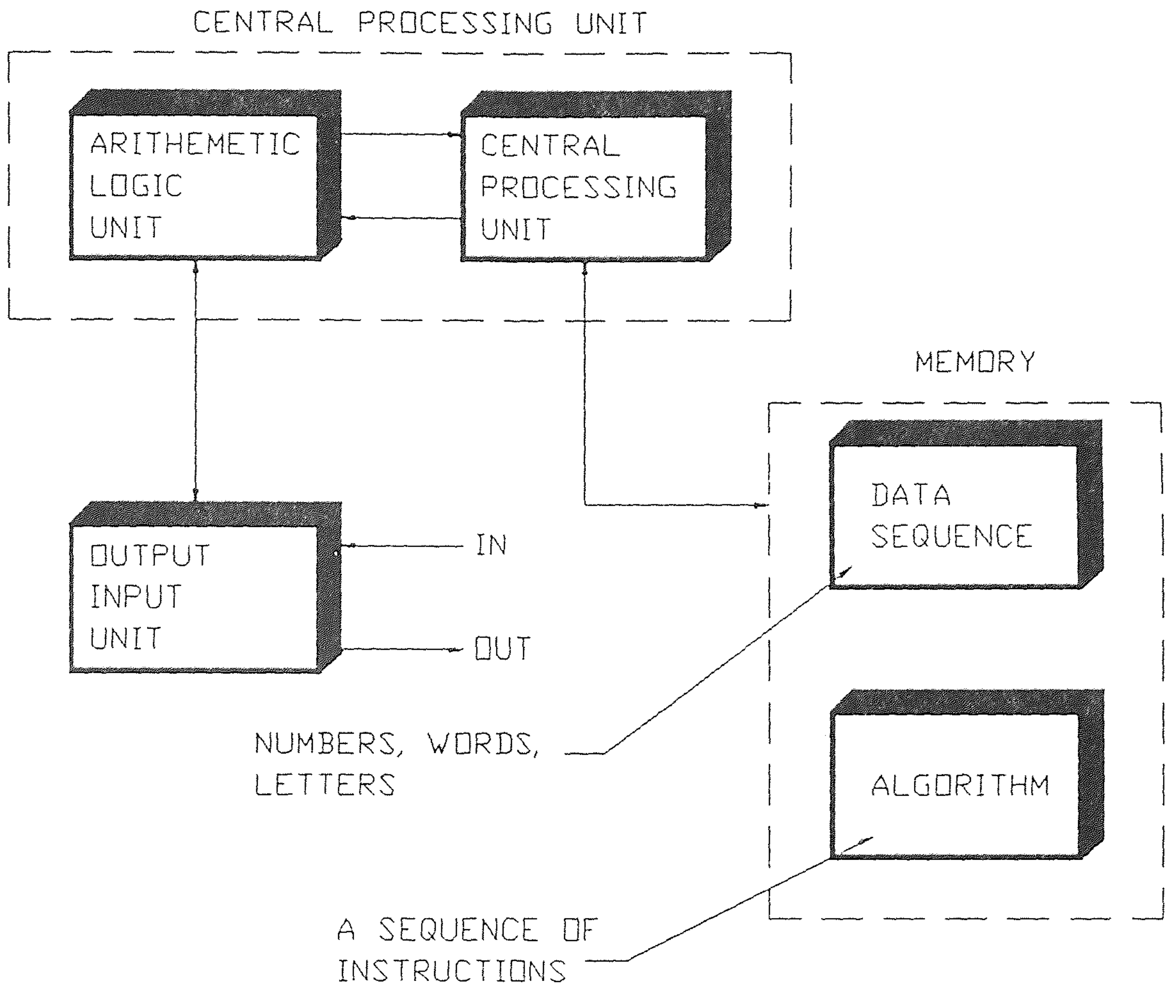


Figure 3. Conventional Computer Processing

data will be retrieved as much as the application program needs it.

- The translation of data from one form to another. An example is utilizing raw data, in numerical form, and translating it to text.
- Sorting data to obtain the desired file.
- To edit data, so files may be updated or deleted.
- The ability for decision making using IF-THEN production rules.
- To monitor specific events such as calculation, data input, and data output.
- The control over internal and external devices such as printers, disk drives, machine tools, and robotics.

To solve a particular problem with conventional computing, the programmer must first analyze the problem and realize how to encode it into the computer. An incremental procedure, using a specific computer language, is usually taken that will produce binary code into the computer's main memory. When the program is executed, it should follow the exact sequence of events that were specified.

3.1.2 Artificial Intelligence Computing

AI computing, as compared to conventional computing, is quite different. AI involves the simulation of the human decision process and, in a way, it mimics the way humans think, (See figure 4 on page 34).

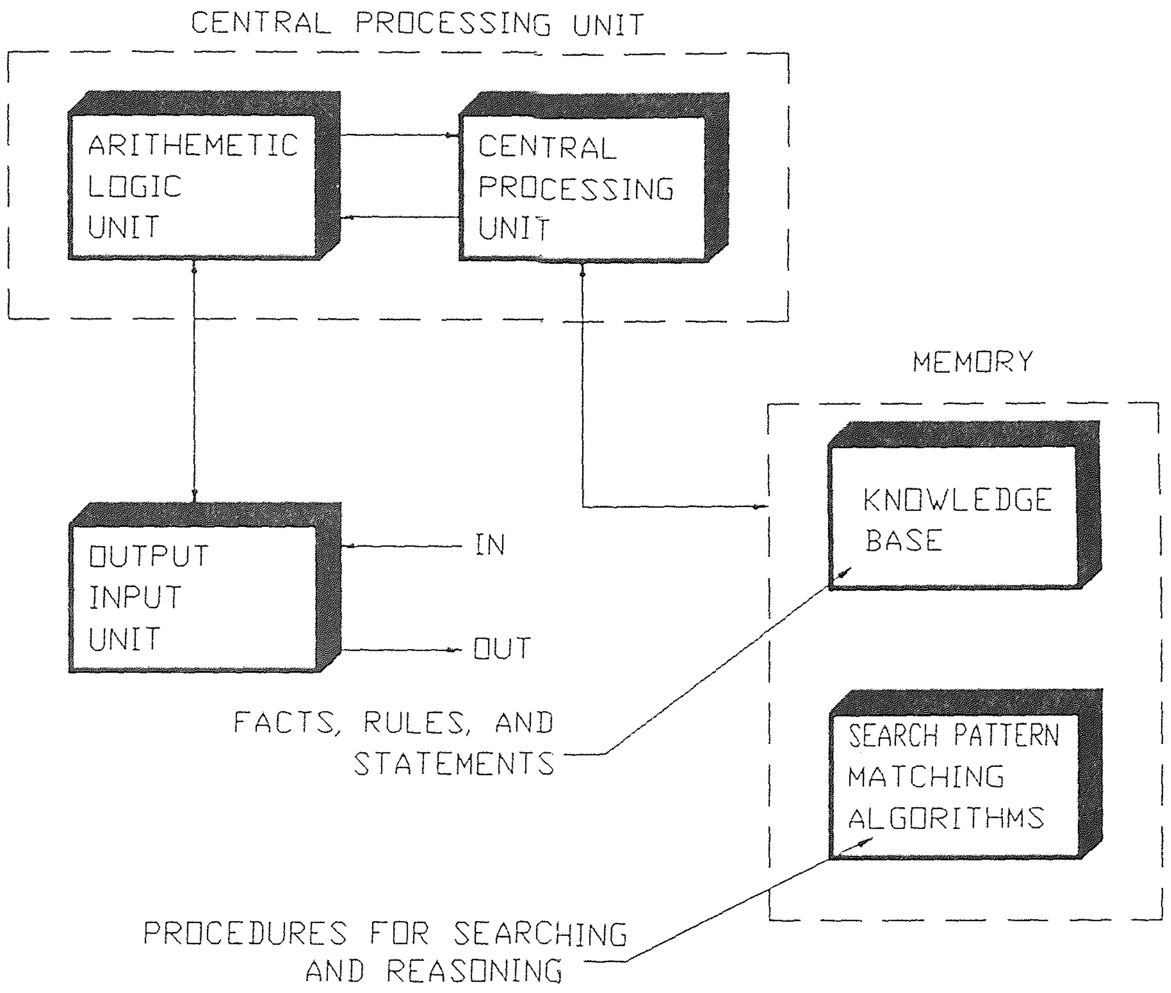


Figure 4. Artificial Intelligence Based Processing

Artificial intelligence computing can be broken down into six general areas, (See figure 5 on page 36). These areas include:

- Natural Language/Speech Recognition: This area of AI study concentrates on hardware/software development that enables humans to interact using spoken commands. This is better known as speech recognition. Speech recognition research involves the differentiation of similar sounding words with different meanings, such as write, Wright, and right.
- Robotics: This is an area of artificial intelligence computing that uses a combination of techniques to develop intelligent robots that can see, move, and manipulate objects on their own in response to changing environmental conditions. An example would include robot vision. Robot vision is usually held to inspection tasks.
- Expert Systems: The most commercially successful area of artificial intelligence, expert systems utilize artificial intelligence concepts to enable computers to function in decision roles as advisors that provide decision making ability.
- Other areas of AI computing include exploratory programming and enhanced human interfaces. Exploratory programming deals with computers programming themselves. Enhanced human interfaces involves utilizing artificial intelligence in the design of user-friendly human

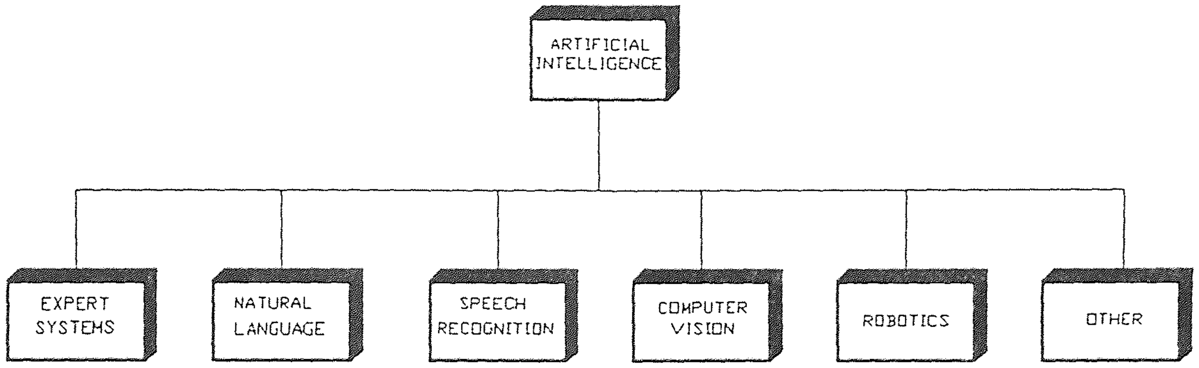


Figure 5. Generic Areas of Artificial Intelligence

interfaces, such as input/output devices.

AI based systems use a pattern matching approach. If sufficient information is provided, the system may not arrive at the conclusion or solution. Instead of following a data sequence or algorithm, AI computing involves the use of heuristics, logic reasoning, and search techniques. AI based systems use this type of pattern matching approach.

3.1.2.1 Use Of Heuristics

As mentioned earlier, heuristics is knowledge that is practical and known. It is "real world", meaning it contains all the tricks of the trade that has been developed through experience.

Heuristics could also be applied to a search technique, in that the search process is made faster and more efficient. It is a way of focusing on only a percentage of a search tree that will provide the optimal solution.

There are two basic types of heuristics used in artificial intelligence systems. These include general-purpose heuristics and domain-specific heuristics.

- General-purpose heuristics are used in search techniques that limit the depth of searching in a network of branches. This is generally known as a depth-bound search technique.
- Domain-specific heuristics are applicable to certain types of problems, such as limiting a search to a small subset of knowledge within a branch or group.

3.1.2.2 Logic Reasoning

Most AI systems apply deductive reasoning to provide an optimal solution to a particular problem. AI programs usually contain a collection of logical facts in their databases that can be deleted or updated. Given data supplied by an operator, the AI program will use logical theorems to prove or dis-prove a particular assumption. Consider the following example:

- ALL TIRES ARE ROUND:

  A--->B (A: TIRES, B: ARE ROUND).
- IF AN OBJECT IS ROUND THEN IT CAN ROLL:

  B--->C (C: TIRES CAN ROLL).

This form of logical reasoning, called a transitivity relationship, concludes that all tires are round and therefore they can roll.

In AI systems, logical reasoning is typically a multi step process. The reasoning represented as rules and principles may be used to derive conclusions about a particular subject.

3.1.2.3 Search Techniques - Decision Trees

The heart of most AI systems is the ability to search for the most feasible solution. Utilization of a information tree, or search tree, is the most common technique. This starts at a root node with children following each node. The number of nodes increases exponentially with the number of levels and with the number of alternate choices at each search level. Two common forms of search techniques are known as backtracking and graph searching.

In backtracking, a point of return is selected when a rule is applied. Should any difficulty occur during processing, the program can revert back to the return point and another rule can be applied.

In graph searching, two basic procedures which are extensively used, include blind searching and informed searching.

Blind searching involves a control procedure ignoring problem related information to guide the search. In a way, blind searching is a random way to access information.

Informed strategy is quite different. It involves restricting the search with problem related knowledge.

Searching is broken down into a tree where information can be evaluated. The terms used in a simple tree search, as described by Soundar R. T. Kumara, Rangasami L. Kashyap, and Allen L. Soyster, include:

- Root Node: The starting state or initial configuration, which represents the beginning of the search tree, since a tree only has one root.
- Leaf Node: A node of the tree which does not have any children (successors).
- Goal Node: The node which represents the configuration satisfying the goal state or solution.
- Branch: The link that connects any two nodes (i and j) together; node j is generated by applying an operator or

rule k to the node i.

- Expansion: A node i is said to be expanded if an operator(s) is(are) applied to it and its successors are generated.
- Termination: The process of stopping the search procedure, normally when the goal is reached.
- Level: The root node in a tree is said to be at a depth level 0. Any other node's depth is given by LEVEL = level of its parent node + 1.

3.2 Artificial Intelligence and Process Planning

A productive computer process planning system must contain enormous amounts of expert information for it to be useful. The information programmed into a process planning system must be relative to a particular operation and should be factual about manufacturing operations. The process planning system should be flexible since facts and rules, contained in the database, have to be updated or deleted. Updating into the AI system is very important in a manufacturing scenario since technology is continuously changing and improving. The ability to update database information and to be flexible is an important requirement because the system must change with the technology.

In a traditional CAPP system, manufacturing knowledge is encoded line by line into the program's statements. Any modification to these statements would result in rewriting the entire program. In other words, the traditional CAPP system cannot update the information unless it is re-encoded into the program. The rigidity of this type of system endangers the ability for linking information from one design package to another. In particular, linking drawing files from a CAPP system to a CAD (computer aided design) system, which is an important factor in integrating CAD/CAM.

An expert system, such as generative computer process planning, will store information so it is possible to add, delete, and modify facts about manufacturing concepts. They optimize the production process by helping to identify critical parameters in order to solve a particular problem.

3.3 Expert Generative Computer Process Planning Systems

A commercially successful branch of artificial intelligence, expert GCPP systems are an attempt to capture the heuristics used by experts to solve problems through computer programming. These types of systems attempt to utilize expert knowledge and apply it to manufacturing scenarios, (See figure 6 on page 42). Design of an expert system requires an enormous amount of expert knowledge. The process of extracting information and encoding it into a program is known as knowledge engineering. As explained earlier, knowledge engineering refers to the process of transferring an experts knowledge to the algorithms of an expert system.

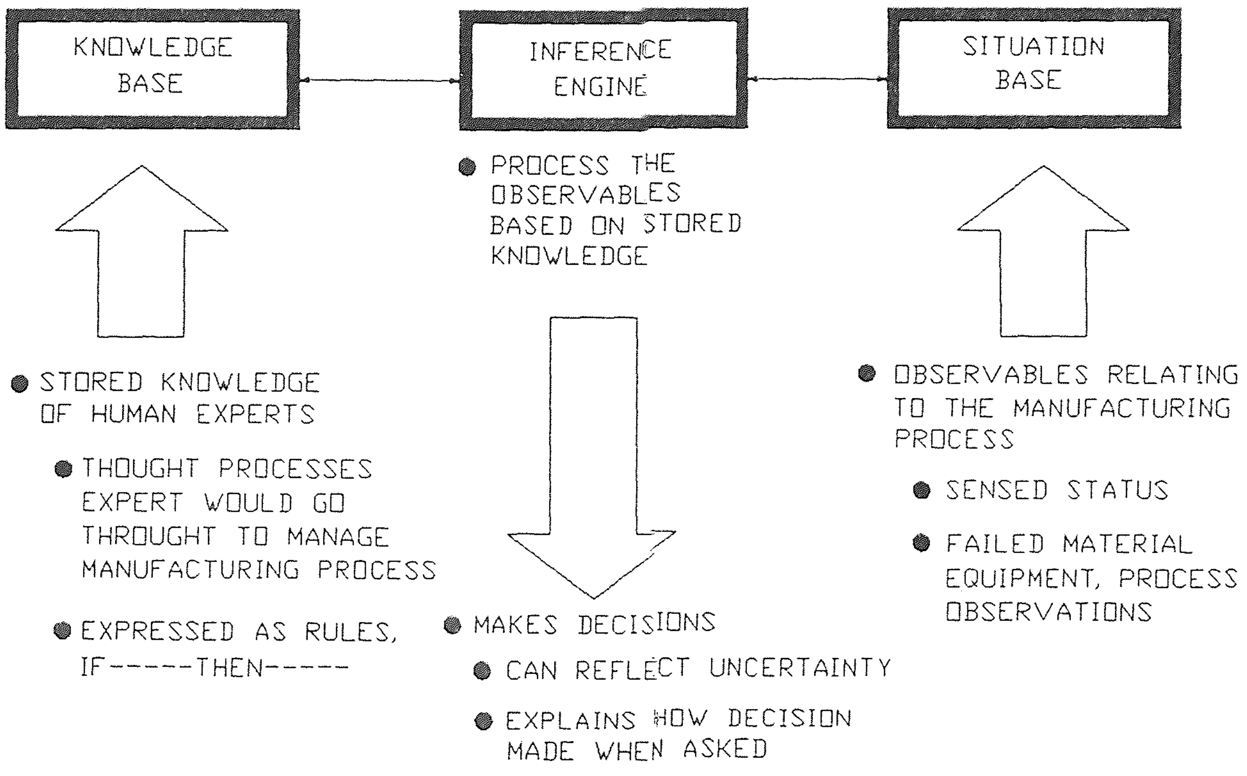

KNOWLEDGE BASE
INFERENCE ENGINE
SITUATION BASE
● PROCESS THE OBSERVABLES BASED ON STORED KNOWLEDGE
● STORED KNOWLEDGE OF HUMAN EXPERTS
● THOUGHT PROCESSES EXPERT WOULD GO THROUGHT TO MANAGE MANUFACTURING PROCESS
● EXPRESSED AS RULES, IF-----THEN-----
● MAKES DECISIONS
● CAN REFLECT UNCERTAINTY
● EXPLAINS HOW DECISION MADE WHEN ASKED
● OBSERVABLES RELATING TO THE MANUFACTURING PROCESS
● SENSED STATUS
● FAILED MATERIAL EQUIPMENT, PROCESS OBSERVATIONS

Figure 6. Elements of an Expert Generative Computer Processing System

Expert GCPP systems are designed as computer programs that have the ability to search for pre-programmed data, and to utilize the data to process the optimal solution for the given problem.

Generative computer process planning will create a process plan automatically for a specified type of component. The system will synthesize process information that is given by the user. With decision logic and optimization formulas encoded into the system, human input is reduced. A generative CAPP system will produce a complete process plan, from design data through a particular engineering drawing to providing manufacturing processes that are required. The software that is presented later in the thesis is designed using these principles. Generative systems are usually designed by identifying component characteristics first. The characteristics may include features, dimensions and tolerances, surface quality, and other specifications. An ideal situation, which was encoded into the thesis software, is the design of an interface between a generative CAPP system and a CAD system. All design characteristics is to be interpreted and processed in the CAPP system, and additional design and editing may be done by utilizing a CAD database. This may be done by placing design data into a common file exchange format. Formats that are being used by today's CAD software includes a DXF file format (Data Exchange Format) and IGES (Initial Graphics Exchange Specification)

Operation and sequencing is one component that is essential to AI based generative systems. Once component characteristics are input into the system, manipulation and calculation is necessary to generate routing and operating information.

With generated engineering drawings and process information, encoding report generation functions into the system is essential.

3.3.1 Generative Computer Process Planning Functions

Artificial intelligence based GCPP involves the application of computers to assist the standard process planning function. In its most basic form, the system will reduce the time and effort required to prepare process plans and will increase the accuracy of each design and plan. With the advancement of such a system, a major function is to provide a automated interface to a CAD and CAM system. This will allow complete integration within the manufacturing system.

Key research functionality areas of computer assisted process planning includes the development and understanding of transformation rules that humans may apply to the planning process, CAPP representation to CAD representation, CAD representation to CAM representation, and CAM representation to the development of a functional work-piece. The complete functionality of a generative based system, as described by Hsu-Pin Wang and Jian-kang Li, would include:

- Design input and output, such as the generation of engineering drawings and the ability to input/output them into another system.
- Material selection.
- Process selection.
- Machine and tool selection.
- Intermediate surface determination.
- Fixture selection.
- Machining parameter selection.
- Cost/time estimation.
- Plan preparation.
- Numerical Code (NC) image generation.

3.3.2 Definition of an Expert, a CAPP Relationship

An expert is a professional who has a masterful knowledge of a particular field. This "field" of knowledge is usually acquired through formal and informal learning experiences. This could include on-the-job experiences, advanced education, and the ability to solving complex problems. The expert is the person who can solve the problem quickly and decisively.

A collection of the experts knowledge may be formulated into computer software to accomplish similar problem solving ability. In a sense, the computer system becomes the expert, a machine to solve a particular problem quickly and decisively with no errors. A general idea is simply a software copy of the expert, in a particular field, that can be called upon at any given time.

3.3.3 Manufacturing Expert Systems

A primary goal in manufacturing is to produce high quality goods and services at the lowest cost. With this in mind, manufacturing expert systems improve productivity by making the process planning function easier, thus high quality goods and services may be provided with the least amount of effort.

Using an expert system to improve productivity begins with the design and setup of production lines capable of maximum output. Utilizing the ability for maximum output can be determined by the expert system with the given processing parameters. In existing manufacturing processes, expert systems can optimize the production process by identifying critical parameters in the proper order of importance. Manufacturing expert systems aid manufacturing engineers to evaluate alternative operations and processes. This helps improve production planning, process planning, and product design. Manufacturing expert systems will also assist the manufacturing manager in understanding and improving lead time, inventory control, and material handling.

Most of today's manufacturing expert systems are designed for specialized operations. Operations such as product design, process planning, and design for manufacture. Theses types of manufacturing systems are also designed for solving problems with plant productivity and plant layout.

3.3.3.1 Manufacturing Robotics, Kinematics and Design

A basic problem of robot kinematics is the movement from a desired chartering position and orientation to moving back of the joint angles required in the movement. A solution to this problem can be acquired with programming knowledge of straight line motion into an expert system. For a robotic arm, the forward function which moves the joint angles of the end-effector is nonlinear and transcendental.

Designing an expert system for solving problems with robot kinematics could prove extremely useful in the manufacturing industry. Manufacturing operations such as spray painting, spot welding, and cutting may be optimized for accuracy and repeatability as well as tool path configuration optimization. Such a manufacturing expert system could be designed to explore the space required for possible robot configurations and searching for designs which satisfy input criteria, such as work-space shape and size.

3.4 Benefits of Expert Generative Computer Processing Systems

The use of expert systems will increase productivity and reduce cost. The benefits of using an expert system includes the increase in productivity, a reduction in cost, the ability to store and update valuable information, and the improvement in communication and understanding.

- Implementing the use of an expert system will increase productivity. The major benefit of an expert system is that valuable information can be generated quickly and accurately, thus productivity will be increased. The expert knowledge can be instantly accessed using a computer, therefore a particular job will be completed in a short amount of time. Time savings will result in higher productivity since the expert system permits the operator to accomplish more work in the same amount of time.
- Expert systems will reduce cost. Cost reduction will result since complex problems can be solved in a short amount of time. The expert system also avoids making costly mistakes and bad decisions, decisions that can lead to waste. Expert systems will reduce cost, but at the same time, will improve the quality of manufacturing operations.
- Storage and retrieval of valuable information is accomplished with an expert system. Valuable manufacturing information can be retrieved and updated to further develop a design. Media such as floppy disks, hard disks, and printing devices provide storage and retrieval ability.
- Expert systems will improve communication and understanding between departments within a organization.

Information is clearly and accurately conveyed through networks, thus supplying important data to engineers, managers, and operators.

3.5 Expert Systems, a Survey

The following is a brief survey of expert systems to illustrate how they are applied to specific problems. They include:

- DENDRAL, the first expert system developed at Stanford University, was designed to determine the structure of unknown chemical compounds on the basis of mass spectrographic data.
- XCON, eXpert CONfigurer of VAX systems, (originally called R1) was developed by the Digital Equipment Corporation (DEC). It was designed to process a customer order, dealing with VAX computer systems, and to select the boards, slots, and cables required to produce a working computer system which meets customer specifications.
- PROSPECTOR, an expert system that was designed to predict the locations of ore deposits based on detailed geological data at a particular site.
- DELTA/CATS-1, An expert system used by General Electric to diagnose faults in diesel locomotives. The system prompts the user to supply instrument values and test conditions, then suggests alternative solutions for repair.

- MYCIN, designed to aid the medical profession in the diagnosis of blood diseases. The system was developed by Stanford University researchers using knowledge extracted from Stanford physicians.
- INTERNIST-1, similar to MYCIN, this generative type system is used for medical diagnosis for internal medicine.
- GARI, developed in France, was the first artificial intelligence CAPP system. GARI utilizes production rules for its knowledge representation and to generate process plans from the model of a part. Geometrical and technological data information is encoded within the program. The model will describe the part in terms of entities, such as holes, grooves, notches, and faces.
- TOM - Technostructure of Machining, written in PASCAL, was developed at the University of Tokyo in Japan. TOM is a production rule based CAPP system that translates design data from the COMPAC CAD system using the IGES (Initial Graphics Exchange Specification). The software that was developed for this thesis is similar to the TOM system.
- SIPP - Semi-Intelligent Process Planner, written in PROLOG, is a artificial intelligence based CAPP system for the creation of metal parts using chip metal removal methods. As defined earlier, SIPP uses frames as its knowledge representation scheme.
- Turbo-CAPP, developed using PROLOG, is a knowledge based

CAPP system that is capable of extracting and interpreting surface features from a CAD database and will perform intelligent reasoning for process planning.

- CAPE, Computer Aided Parts Estimating system, is a knowledge based system that generates, evaluates, and prices auto-part manufacturing plans, and provides current operation benefits.
- DEPICT, Digitized Expert Pictures, an expert system that was developed to assist IBM's memory and logic chip manufacturing facility. DEPICT collects and retrieves images as well as data to speed analysis and the identification of semiconductor defects.
- Virtual Geometry Environment - VGE, is a knowledge-based system that advises users on engineering and manufacturing applications.
- KLUE, Knowledge Legacy of the Unavailable Expert, is an expert system tool that provides an integrated system for development, maintenance, and operation of a diagnostic expert system in a manufacturing environment.
- TOLTEC, a system equipped with basic learning capability. TOLTEC will generate output in the form of operations and their sequences.

All the expert systems illustrated are useful in the following ways:

- debugging
- design
- diagnosis

- instruction
- interpretation
- monitoring
- prediction
- planning
- repair

Each system in the survey does not come equipped with a complete set of manufacturing procedures and knowledge. Most of the systems focus on only a small portion of the issues that are presented in process planning, design, etc.

CHAPTER 4

GROUP TECHNOLOGY FOR AI PROCESS PLANNING SYSTEMS

Group technology is an important topic since it is a main ingredient in computer process planning and other manufacturing activities. With the development of Group Technology (GT) in 1958 by S.P. Mitrofanov, a Russian engineer, parts can be classified and grouped into families.

A popular form of production is known as batch manufacturing. Batch manufacturing requires that part specifications be grouped together in order for efficient production to take place. A major approach to maximizing production in batch manufacturing is called Group Technology. Group Technology also plays a vital role in computer assisted process planning for batch manufacturing.

Group technology is a manufacturing concept where similar parts and components are identified and "grouped" together to form a family. Group Technology concepts will take full advantage of part similarities in manufacturing and design. All part families must posses similar design and manufacturing characteristics in order for coding and classification to take place, hence, the processing of each family member would also be similar.

Parts coding and classification is a major element in Group Technology. It is concerned with identifying the similarities among parts and relating these similarities to a coding system. The two major types of part families include:

- Design attributes; such as geometric size and shape, as well as surface features and chemical composition.
- Manufacturing attributes; The sequence of processing steps required to make the part is also a topic in classifying elements into families.

The topics of part families and classification and coding will be fully explained in the next two sections.

4.1 Classification and Coding

Classification and coding represents the heart of Group Technology. It represents the identification of the similarities among parts and the relation of the similarities to a coding system. Classification and coding systems do allow for certain differences with design and manufacturing schemes but is primarily concerned with producing a coding system. The coding system is designed to facilitate retrieval of part specifications such as design and manufacturing operations. Coding, which is a group technology technique, can be used to describe a part without in-depth detail. Coding and classification for a components representation may be broken down into four factors. These are:

- The total amount of components, such as rotational, prismatic, deep drawn, etc.

- The detail the code should represent.
- The type of code structure, such as hybrid, hierarchical, or chain.
- The digital representation, such as binary, octal, decimal, etc.

Coding, according to Chang, Tien-Chien, Wysk, A. Richard and Hsu-Pin Wang, (7) can be defined as a function of H that maps components from a population space P into a coded space C. A certain code can be defined (for component i) as:

$$i \in P \Rightarrow \exists \text{ only one } j \in C \Rightarrow j = H(i) \qquad (2)$$

Completeness can be defined as:

$$\forall i \in P \exists j \in C \Rightarrow j = H(i) \qquad (3)$$

With a coding system in place, a generative or variant CAPP system will be able to retrieve the required data for use and update.

4.1.1 Principles and Structures of Classification and Coding

Classification is a term that means to sort or place parts into groups that contain similar features. A code can be used, either in numerical form or letter form, as a means to identify a particular classification.

There are many classification and coding principles that are used in today's process planning systems that utilize Group Technology. Two main principles, according to Hsu-Pin Wang and Jian-kang Li, includes Similarity and Fuzzy Classification. Similarity involves grouping "like" parts together for ease of operation. In this case, a sample that consists of n parts and is composed of p features can be represented by a vector:

$$X_i = (x_{i1}, x_{i2}, \ldots\ldots\ldots\ldots, x_{ip}) \qquad (4)$$

The n parts under consideration can bee seen as n points in a p-dimensional Euclidean space. The similarity of the two parts can be measured by the distance $d(X_i, X_j)$ between two points in the Euclidean space.

In order to improve part formation, principles in Fuzzy Classification may be used. As an example, suppose that n parts are to be grouped into C families. The traditional way to classify the parts would be to set up a binary matrix such as:

$$T = \left|\begin{array}{c|ccccc} & X_1 & X_2 & X_3 & \ldots & X_n \\ 1 & 0 & 0 & 0 & \ldots & 1 \\ 2 & 1 & 0 & 0 & \ldots & 0 \\ 3 & 0 & 1 & 0 & \ldots & 0 \\ . & . & . & . & \ldots & . \\ c & 0 & 0 & 1 & \ldots & 0 \end{array}\right| \qquad (5)$$

Where U_{ij} = {1, the jth part}, {0, otherwise}.

In the Fuzzy Classification system U_{ij} is not restricted to binary numbers only. The Fuzzy Classification matrix can be illustrated as:

$$T = \left|\begin{array}{c|ccccc} & X_1 & X_2 & X_3 & \ldots & X_n \\ 1 & U_{11} & U_{12} & U_{13} & \ldots & U_{1n} \\ 2 & U_{21} & U_{22} & U_{23} & \ldots & U_{2n} \\ 3 & U_{31} & U_{32} & U_{33} & \ldots & U_{3n} \\ . & . & . & . & \ldots & . \\ c & U_{c1} & U_{c2} & U_{c3} & \ldots & U_{cn} \end{array}\right| \qquad (6)$$

Where a). $0 <= U_{ij} <= 1$, for i=1,2,...c; j=1,2,...n (7)

b). $\sum_{i=1}^{c} U_{ij} = 1$, for j=1,2,...n (8)

c). $\sum_{j=1}^{n} U_{ij} > 0$, for i=1,2,...c (9)

Each production flow can be represented by a reference pattern, V_i. In equation form:

$$V_i = (v_{i1}, v_{i2}, \ldots\ldots v_{ip}) \qquad (10)$$

Where,

$$V_{i,k} = \sum_{j=1}^{n} u_{ij}(u_k(x_{jk})) / \sum_{j=1}^{n} u_{ij} \qquad (11)$$

i=1,2,...c; k=1,2,......,p

The distance between part j and reference pattern i is:

$$\left\{ \sum_{k=1}^{p} (\mu_k(x_{jk}) - v_{ik})^2 \right\}^{1/2} \qquad (12)$$

and the weighted sum of squares of the distances from part j to all C reference patterns is:

$$\sum_{i=1}^{c} \mu_{ij} \sum_{k=1}^{p} (\mu_k(x_{jk}) - v_{ik})^2 \qquad (13)$$

The total weighted sum of squares of the distances from all parts to all reference patterns is:

$$J(U,V) = \sum_{j=1}^{n} \sum_{i=1}^{c} u_{ij} \sum_{k=1}^{p} (\mu_k(x_{jk}) - v_{ik})^2 \qquad (14)$$

A procedure for utilizing fuzzy classification is given as:

a. Define an initial classification matrix U_o satisfying equations 7, 8, 9.

b. Compute V_i by minimizing the total distance of the sum.

c. Compute the new classification matrix U.

In general, Fuzzy Classification systems provide optimal grouping, resulting in a more precise classification of parts.

4.1.2 Classification and Coding Systems

Generative computer process planning relies heavily on classification and coding. Design and manufacturing are two major topics of classification and coding systems, where both are concerned with part grouping. Parts classification falls into three areas. These areas are:

- Classification of part design attributes. Design attributes such as surface texture, tolerances, dimensions, material composition, length to diameter ratio, and internal/external shape all fall into this category.
- Classification of part manufacturing attributes. Design attributes such as fixtures required, cutting tools, production time, surface finish, type of process, batch size, and operation sequences fall into this category.
- Classification of both design and manufacturing

attributes. This type of overlapping classification system is concerned with the combination of both design and manufacturing specifications.

Utilization of numerical digits, arranged in sequence, code a work-part's manufacturing and design attributes. Three major forms of coding include:

- Hierarchical coding. This form of coding involves the interpretation of each symbol depending on the value of the proceeding symbol. Hierarchical coding is also known as monocoding. One advantage of a hierarchical structure is that it can represent a large amount of information with very few code positions. A disadvantage of hierarchical coding is that it is sometimes difficult to develop because of all the branches in the hierarchy that must be defined. For a coding system with N digits and M attributes for each digit, the number of possible combination of codes is given as:

$$R_{mono} = \sum_{n=1}^{N} M^{n} \qquad (15)$$

- Chain coding. This form of coding relies on each symbol, in sequence, being fixed. That is, it does not depend on the proceeding symbol. Chain coding, also called polycoding, is much easier to construct and use. The disadvantage of using chain coding is that it cannot be as detailed as hierarchical coding with the same number of coding digits. The number of possible codes the system may generate is given as:

$$R_{chain} = M(N) \qquad (16)$$

- Hybrid coding. Hybrid coding is a cross between chain coding and hierarchical coding. An illustration includes the number of digits of monocodes N_n and chain codes N_c and the number of combinations in the system is given as:

$$R_{hier} = \sum_{n=1}^{N_n} M^n + M(N_c) \qquad (17)$$

Five major coding and classification systems utilized in the design of a generative computer processing system includes the Opitz, MultiClass, Vuoso-Praha, KK-3, and DCLASS classification systems.

One of the first classification and coding schemes was developed by H. Opitz of the University of Aachen in West Germany. This system was originally developed for the classification and coding of mechanical parts.

The Opitz coding system utilizes a total of nine digits. The digits can be further expanded by adding an additional four digits. The Opitz system is in the form:

12345 6789 ABCD

The nine digits represent both manufacturing and design data, the first five, known as "form code" represent the design specifications of the part. The following four digits, known as supplementary code, represent the manufacturing aspects of the part. Secondary code, in the form of an extra four digits, are intended to code the production operation.

The MultiClass classification and coding system is a commercial product offered by the Organization for Industrial Research (OIR). The major benefit of using the MultiClass system is that it allows for the customization of the classification and coding scheme.

The MultiClass system can consist up to thirty digits, which are divided into two areas. The first area is a fixed system provided by the Organization for Industrial Research. The second area is reserved for the user defined classification and coding system, which is useful in customizing an organizations particular needs.

The Vuoso-Praha classification system utilizes a four digit system that labels a part by Kind, Class, Group, and Material. The Vuoso-Praha system is primarily used for rough part classification so as to identify the type of department that would produce the part.

The KK-3 classification system is a general purpose system for machining parts. It was developed by the Japan Society for the Promotion of Machine Industry (JSPMI).
The KK-3 system, based on a 21 digit decimal system, provides classification to metal cutting and grinding components. The major benefit of the KK-3 system is that more information can be represented, as compared to other classification and coding systems.

The DCLASS classification and coding system is a tree structured system that was designed to be a decision making classification system. The DCLASS system will generate codes for components, materials, processes, machines, and tools. For components, an eight digit code is used:

Digits 1 - 3	Basic shape
Digit 4	Form feature
Digit 5	Size
Digit 6	Precision
Digits 7 & 8	Material

In the DCLASS system, conditions are represented as branches where a code can be found at the junction of each branch. Multiple passes of the decision tree allow a complete code to be found.

4.1.3 Benefits of Classification and Coding

The major benefits of a well designed classification and coding system for use in a CAPP system is as follows:

- Classification and coding facilitates the formation of part families and machine cells.
- It permits for quick retrieval of manufacturing information such as design specifications, engineering drawings, and process plans.
- Design duplication is reduced since information is kept in a similar data structure with an individual coding scheme.
- Reliable work-piece statistics are provided with the

encoded information.

- Accurate estimation of machine tool requirements, speeds and feeds, and logical machine loading is facilitated.
- It permits rationalization of tooling setups, reduction of setup time, and the reduction of production throughput time.
- Process planning, production planning, and scheduling procedures are made easier and more effective.
- Cost estimation and facilities cost accounting procedures are improved.
- Better machine tool utilization and better use of tools, fixtures, and labor is provided.
- Most importantly, classification and coding facilitates numerical control programming.

4.2 Part Families

A part family is a collection of parts which are similar in geometric size or similar in the processing steps required for manufacture. The parts within the family may be different, but share the same characteristics.

The advantages of collecting similar parts into families are enormous. Machine tools can be arranged so that parts can be machined in the same family, design specifications can be grouped together, etc. By classifying parts into families, higher production will result.

Arranging parts into families can prove to be a difficult and time consuming effort. There are three general methods that aid this process. These methods include:

- A visual inspection of the part. This involves classifying parts by visually inspecting the work-part for specific geometries and surface features. Similar parts will be placed in the same part family.
- Examination of design and production data. This method involves the examination of individual design and manufacturing attributes of each individual part. Sampling procedures may be utilized to establish a particular part family.
- Production flow analysis or PFA. This method relies on the information contained on route sheets rather than engineering drawings/design specifications. Production flow analysis is used to analyze the operation sequence and machine routing for the parts produced. Obtaining this data will result in the specifications required for a part family.

4.3 Production Flow Analysis

A technique for identifying part families is known as production flow analysis. Although the data used in the analysis is obtained from production route sheets, PFA will analyze the operations sequence, resulting in grouping for parts with similar routings. An advantage of using PFA is that manufacturing information is used instead of design data. This results in similar process routing for parts with different geometries.

The PFA procedure, as stated by Mikell P. Groover, is organized in the following steps:

- The first step in production flow analysis is to collect and study all relevant data. Data collected will be used to determine the population. Using route sheets, further data such as part numbers and operation sequences, will be needed.
- The second step is to utilize the information from the process sheets to arrange all the work-parts into groups or families. This is done by finding the similarities of the parts process routings.
- The third step is to present the data collected graphically in a PFA chart. The PFA chart is simply a plot of the process codes.
- The final step in production flow analysis is to place all similar data, obtained from the PFA chart, into groups.

4.4 Design Retrieval and Group Technology

Design retrieval allows engineers to locate data about existing designs. Using group technology concepts, designs and design data can be updated or stored within a CAPP system. A design retrieval system plays a vital role in process planning and group technology.

The four components of a design retrieval system include:

- A classification and coding system.
- A base of existing designs and equations.
- A database linking existing designs to the classification and coding system.
- A user interface to allow the user to locate similar designs.

A database management system (DBMS) is a effective tool for implementing design retrieval within a CAPP system. A classification and coding scheme will identify the key design attributes that is required for searching a database. The database will relate the attributes with the design data such as solid models, engineering analysis, and process plans.

4.4.1 Benefits of Design Retrieval

If a design is reused and updated, it is beneficial to optimize the design itself along with the process plan for manufacturing it. An organization with an effective design retrieval system can produce a product with less design effort by retrieving a design and modifying it instead of creating a new design. This provides cost savings. Other benefits of this system include:

- Design retrieval provides corporate management with greater strategic flexibility.
- A product can be designed with less effort at a lower

cost.

- Design retrieval will dramatically reduce the product development cycle.
- Provides a more tailored product by modifying the original design.
- Design retrieval allows an organization to provide a broader range of product options.

CHAPTER 5

CAD/CAM AND GENERATIVE COMPUTER PROCESS PLANNING

Computer aided design (CAD) and computer aided manufacturing (CAM) are two separate entities that can be combined to form CAD/CAM systems. Integrating a CAD/CAM system with an expert system, such as artificial intelligence based generative computer process planning, can be extremely useful, in that, the best of both elements can work together to solve a common problem.

Computer aided design primarily is concerned with the analysis and optimization of a particular design. Drafting and other functions, such as finite element analysis (FEA) and animation, are usually utilized with this system.

On the other hand, Computer aided manufacturing focuses on design for manufacture. CAM is primarily used for production scheduling, inventory control, computer numerical code generation, and other manufacturing concerns.

The major benefit of combining CAD with CAM is that a particular design can be developed and the manufacturing process can be controlled from start to finish. A CAD/CAM system allows for the complete production of a product, from design to manufacture using one or more computer systems, resulting in increased productivity and planning effectiveness.

5.1 CAD/CAM Evolution

Initially, CAD systems were primarily drafting stations that produced engineering drawings and other documents. These systems were linked to computer controlled plotters to produce a hard copy of the drawing. The early CAD systems were later linked to graphic displays, such as a CRT, where geometric modeling could be achieved. Two and three dimensional objects could be manipulated and analyzed with complex graphical techniques.

Today, these same systems are based upon interactive graphics, that not only provide engineering drawings, but provide a wide array of tools. These tools usually consist of finite element analysis, three dimensional modeling, real-time simulation, and various kinematic techniques. Kinematic analysis combined with a CAD system allows the motion of mechanisms to be studied in "real-time". Real time referring to graphical simulation at normal time conditions. The simulation could also be sped up to analyze the design at a future period in time.

Along with the development of CAD systems, CAM systems were also being developed to meet manufacturing needs. These systems were primarily designed to generate computer numerical control (CNC) code. Manual CNC programming is proven to a laborious and time consuming effort that sometimes results in costly errors. The early CAM systems also had limited production planning features.

The integration of CAD/CAM systems with A CAPP system allows for the geometric modeling and analysis of a work-piece with the preparation of CNC code and process planning for manufacturing. The major benefit of CAD/CAM systems is that the system allows the engineer to go from an initial concept to a finished work-piece with one computer system. Integrating CAD/CAM with AI based CAPP systems contain three basic concepts: These are:

- Design and manufacturing data can be transferred automatically between different modules and user groups within one system. Shorter production time is obtained resulting in higher productivity and fewer costly mistakes.
- There is a standard entry to any part of the system. The system is controlled by an executive program in order for data flow between modules to be properly controlled. The benefit here is multiple design stations may be placed throughout the organization, such as a having workstations in a design office and on the shop floor.
- All modules are designed with a common interface. This includes software design such as pull down menus and a user-friendly environment.

5.2 CAD Fundamentals

Computer aided design involves design, development, and analysis of a product. CAD can be categorized as a rapid prototyping device that allows storage and retrieval of a particular design. It is a system designed for users who are skilled with computer hardware and software, in systems analysis and methodology.

The functions of CAD may be grouped together into four distinct categories. These include:

- Geometric modeling and design.
- Engineering analysis.
- Kinematics.
- Drafting.

Geometric modeling and design involves the use of three dimensional models, usually a wire mesh, to analyze a particular design. Although three-dimensional wire frames may be adequate to represent the solid nature of an object, it sometimes requires further development. A solution, usually built in the CAD system or provided by "add-on" modules, is a geometric modeling and shading module that contains elementary building blocks of elementary solid shapes. Theses shapes are better known as primitives. AutoCAD's Advanced Modeling Extension (AME) module provides this function. CAD geometric modeling can be closely tied to numerical control code generation of a CAM system. Sculptured surfaces on three-dimensional CAD models can be linked with CAM capabilities. An example would include importing an AutoCAD drawing file into a SmartCAM system. The SmartCAM system will then be able to generate NC code with the imported drawing file.

Engineering analysis and kinematics is another area of CAD. For example, today's CAD systems can move into analysis, calculation of weight, volume, surface area, moment of inertia, and center of gravity. This is possible on any type of geometric model. The CAD system is also capable of generating the finite element model of the wire frame representation.

Drafting and forms of animation are very important elements of CAD systems. Computerized drafting allows for the documentation of a particular design. It allows for easy update and retrieval of files and reduces drawing production time. Following geometric modeling and drafting, three-dimensional animation is possible. This type of analysis allows the engineer to be sure that the moving element does not impact on other parts of the structure.

The CAD design station configuration usually is broken up into parts, (See figure 7 on page 73). They consist of a central processing unit (CPU), a graphics display unit (CRT), input devices such as a keyboard, digitizer, or mouse, general output devices such as disk drives, and hard copy output devices such as printers and plotters.

Today's CAD systems can be categorized as one of four types of interactive graphics systems. These include:

- Local I/O systems. These systems provide the input and output devices locally.
- Intelligent terminal systems. This type of system provides the controllers for the input and output

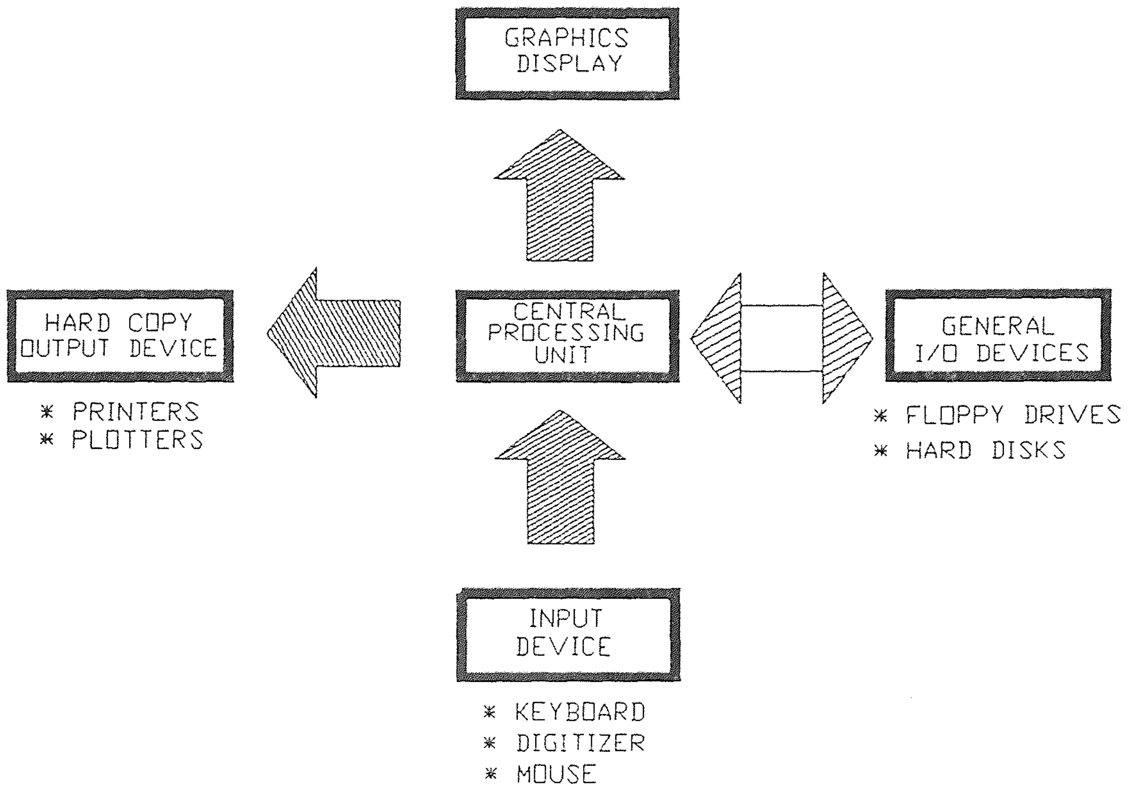


Figure 7. A Typical CAD System

themselves. They contain I/O processing facilities for providing I/O timing. They also contain a special microprocessor for executing device driver routines and small portions of application software.

- Intelligent satellite systems. Intelligent satellite systems provide the local holding of all the systems routines and all of the application program. This provides for quick processing.
- Local stand-alone systems. The local-stand alone system is primarily used for an intelligent satellite. This provides local processing power, storage, and hard copy capability.

5.3 CAD and Finite Element Analysis

Finite element techniques are widely utilized for the analysis of static, dynamic, and thermal stressing of structures. Using a CAD system to analyze finite elements of a structure has considerable advantages over manual methods. The advantages include the graphical representation of element connections and position, and the ability to change a wire mesh instantly. This is beneficial because the best mesh arrangement can be determined to solve a particular design problem.

5.4 The CAD/CAM Database

CAD/CAM systems are primarily designed to control various manufacturing activities, such as production planning, analysis and synthesis, and product design. Utilizing such features requires an enormous database filled with expert knowledge in manufacturing and design. The database requirements for these types of abilities include:

- Various forms of engineering data, such as engineering drawings, and machine data. Machine data such as fixtures, tooling, and jigs, must be managed by the CAD/CAM system. CAD/CAM systems must also manage design analysis data, process organization, and bills of materials.
- The CAD/CAM database must be extremely large. This is necessary since engineering drawings, shape descriptions, and design data require large amounts of memory.
- The CAD/CAM operator must not have to keep required syntax that may be needed at a later time. The required information must be stored in the database to provide easy retrieval and update.
- Tentative and iterative design processes must be supported.
- Until an actual design is carried out, some parts of data structures in the CAD/CAM application fields may be unable to be defined.
- Management of an integrated CAD/CAM database by a

distributed database management mechanism must be taken into account.

- A dynamic data structure control mechanism must be taken into consideration. This provides the capability for engineers to define the data structure at any time.

5.5 CAD/CAM with CAPP Improving Productivity

Using a CAD/CAM with a AI based CAPP system provides numerous benefits. Combining each system will improve productivity by providing automatic design and analysis, quick data retrieval, and various other functions. The complete design and manufacturing planning process is automated from start to finish. This allows for more accurate and efficient designs resulting in cost savings.

5.5.1 Engineering Drawing

Engineering drawings with recurring features and documents, that are frequently updated, are efficiently completed with a CAD system. Initial designs may be imported from a CAPP system using the Data Exchange Format. Retrieval is automatic, so there is no wait to locate a set of documents. Updating is easy since a portion of the document can be edited. Once the change is made, the document can be saved in the system.

5.5.2 Other Forms of Documentation

Bills of materials and technical illustrations are quickly produced from data that is entered into the system and from data that is located in the system's database. This provides a quick and effective way to organize data into a single document.

5.5.3 Engineering Design and Calculation

Calculations of arc, volume, weight, deformation, thermal flux, and so on are easily performed by the computer. CAD systems can either perform these calculations separately or together with design specifications. With having the CAD system perform engineering calculations, design time is saved, therefore more work can be performed in the same amount of time.

5.5.4 Engineering Cost Estimation

The ability of CAD systems to store and retrieve graphical and text data can be put to good use by engineering estimators. The CAD system can be utilized in this way by accessing a particular data file and extracting all the data.

5.5.5 Production Order

Valuable production time can be optimized by combining order entry with a CAD system. Major savings may result in this area where an order is tied to a specific engineering drawing.

5.5.6 Manufacturing

CAD/CAM software can generate numerical control (NC) code for entry into a CNC controller unit. Manual NC programming is proved to be a laborious and time consuming task and is also prone to costly mistakes. NC code generation with a CAM system greatly reduces the effort necessary to get a design into production.

5.5.7 Production Scheduling

Utilizing CAD/CAM's tools will improve production scheduling and shop load because of standardization of operation sequences, tooling, and machine tool selection.

5.5.8 Labor

CAD/CAM will provide a reduction in labor cost. The cost reduction results from automatic processing of manufacturing data, process plan requirements, and other forms of paper work.

5.5.9 Response to Changing Market Conditions

CAD/CAM systems provide the power for quick response to changing market conditions and demands because of product changes and improvements in the market. This can be made without costly down-time. A particular engineering design can be called up automatically, where changes can be made easily to satisfy the future market requirements.

5.6 Reasons for Integrating CAD/CAM with AI Based Generative Computer Process Planning

There are many reasons for integrating CAD/CAM with AI based generative computer process planning. The primary reason is to increase productivity by automating design. Other reasons include:

- Introduction to new technology for improvement, which should be a goal to any organization.
- To accommodate the changing market requirements in order to stay competitive.
- Completing a design in a more efficient way.
- Having the desire to integrate the design function with the process planning function.
- Utilizing the speed of response with a CAD/CAM system.

In order for a company to be competitive, it must utilize its resources to the fullest extent. Design, which is a highly iterative process, is reviewed, modified, and reviewed again at every stage of the design process. This process becomes laborious and time consuming. Engineers and designers should expect to have expert systems integrated with CAD/CAM systems. Data exchange files between a CAPP system and a CAD/CAM system is one alternative. This will allow for much needed information, from the CAPP system, to be imported into the CAD/CAM system, resulting in not having to manually enter geometry or model attribute data into the system.

CHAPTER 6

SOFTWARE DESIGNED FOR THESIS

The software designed for this thesis was written and compiled using a Borland Turbo C++ compiler. The main objective was to design a generative computer process planning system (expert system), utilizing topics in artificial intelligence, that would generate graphical straight line developments from three-dimensional geometric shapes. In turn, the straight line developments could be used for automatic packaging design and process planning. The software is also designed to interface with various CAD systems, to print graphics, and to print process forms, (See source code in appendix A).

To generate the ability to interface with outside CAD software, the thesis software was designed to output a DXF (Data Exchange Format) file. This allows for all point coordinates of the straight line development to be put into a file where it can be regenerated and further processed using a CAD database. Standard DXF information provides the graphics setup where generated point coordinates will give the locations of the polylines.

Graphics printing ability is very important to convey the engineering drawing into a hard copy format. The thesis software was designed to perform a raster scan of the screen coordinates (Video Graphics Array, 640 x 480 pixels). Data is sent from the raster scan to a printer, preferably a laserjet, where it will be printed in a portrait, landscape, or greyscale format. All data and entered information will be printed directly from the screen in order to provide a process plan.

Printing formal process plans was considered and encoded into the thesis software. As process data is entered by the operator, it is stored into memory where it can be utilized for processing. By calling this data and combining it with a process plan, a simple printing function was programmed to output the finished process plan.

6.1 Operation of the Thesis Software

The thesis software can be activated by entering:

DRIVE: *NAME OF FILE.extension*

example: a: AIGCPP.EXE

NOTE: The graphics developed for this thesis software was programmed in a VGA format, therefore it will only operate using a VGA equipped computer. Printing functions are only available on laserjet printers using a 100, 150, and 300 DPI mode.

When the introduction screen appears, press any key to activate the main menu.

The main menu consists of three separate windows. Window #1 provides the geometry that is available. It consists of three-dimensional shapes (cylinder, square, wedge) which represent a folded straight line development. Window #2 is where the user may interface with the system. A prompt for selecting a particular geometry is situated at the top of the window. The prompt may activate three modes. These are:

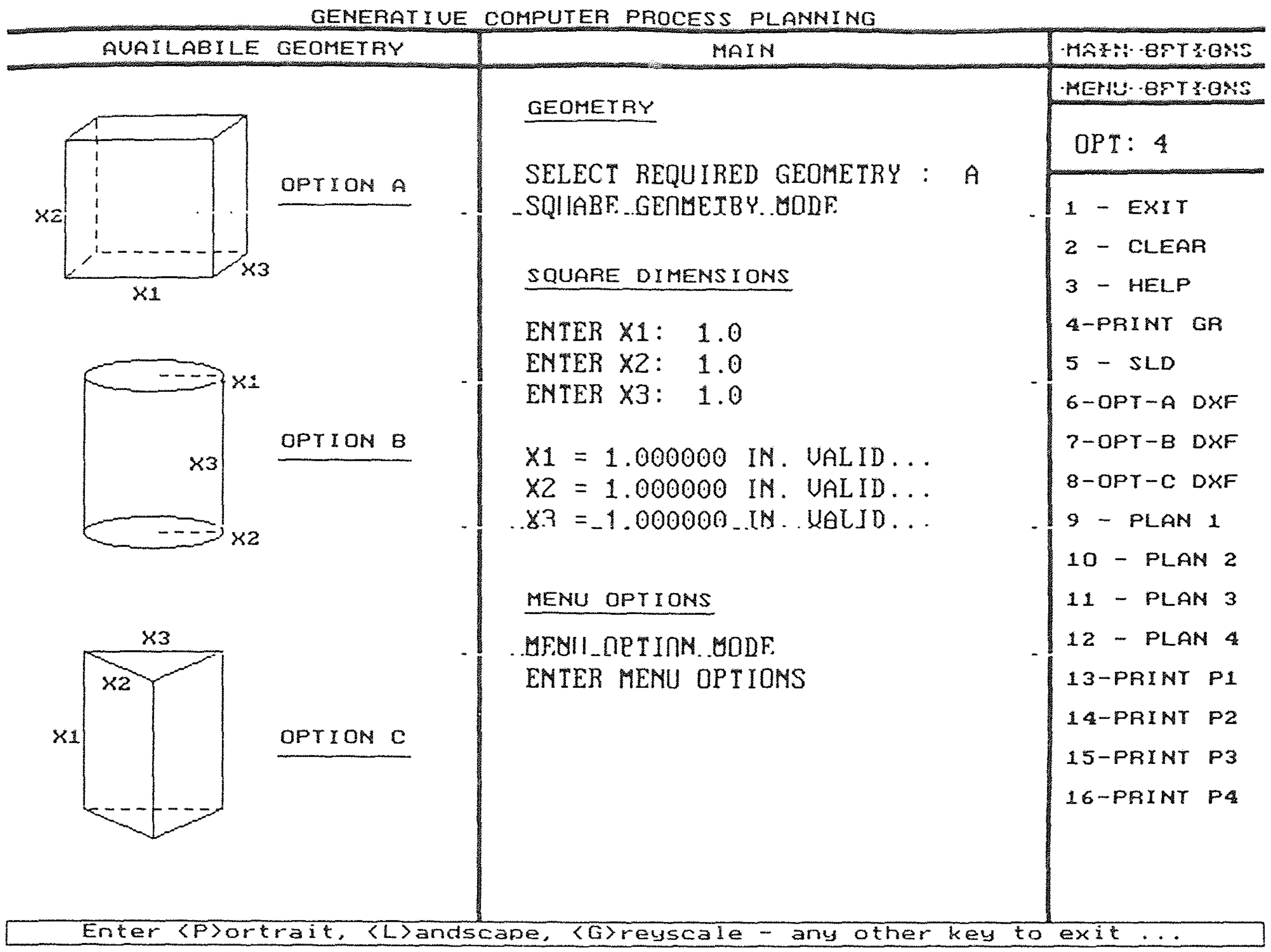
- Square geometry mode. This mode sets all data to be processed using only the square geometry code, (See figure 8 on page 84).
- Cylindrical geometry mode. This mode sets all data to be processed using only the cylindrical geometry code, (See appendix B).
- Triangular geometry mode. This mode sets all data to be processed using only the triangular geometry code, (See appendix B).

All entered dimensions and process data will be processed according to the mode selected. Window #3 provides menu options enabling the user to "jump" around in the program. Window #3 consists of sixteen options. These options are:

Option 1: Provides the user to exit the program at any point. Once the option is activated, the graphics mode is terminated and the active drive command prompt will appear

Option 2: A function that clears the menu screen when new dimensions are needed. This is an important function since data input mistakes may occur, therefore it will provide a way of eliminating all input data.

Option 3: Provides on-line help to guide the user in software operation.



```
GENERATIVE COMPUTER PROCESS PLANNING
AVAILABILE GEOMETRY          MAIN                          MAIN OPTIONS
                                                           MENU OPTIONS
                             GEOMETRY
                                                           OPT: 4
                             SELECT REQUIRED GEOMETRY :  A
                             SQUARE GEOMETRY MODE          1 - EXIT
                                                           2 - CLEAR
                             SQUARE DIMENSIONS             3 - HELP
                                                           4-PRINT GR
                             ENTER X1:  1.0                5 - SLD
                             ENTER X2:  1.0
                             ENTER X3:  1.0                6-OPT-A DXF
                                                           7-OPT-B DXF
                             X1 = 1.000000 IN. VALID...    8-OPT-C DXF
                             X2 = 1.000000 IN. VALID...
                             X3 = 1.000000 IN. VALID...    9 - PLAN 1
                                                           10 - PLAN 2
                             MENU OPTIONS                  11 - PLAN 3
                                                           12 - PLAN 4
                             MENU OPTION MODE              13-PRINT P1
                             ENTER MENU OPTIONS            14-PRINT P2
                                                           15-PRINT P3
                                                           16-PRINT P4
   Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...
```

Figure 8. Square Geometry Mode Screen

Option 4: A function that will print the graphics screen using a raster scan. Option 4 provides a hard copy of the raw data and straight line developments which can be used as a raw process plan.

Option 5: Provides a full screen straight line development of the selected geometry. (See figure 9 on page 86 and appendix B)

Option 6: A function that will output a DXF file, to a disk destination, of the square geometry.

Option 7: A function that will output a DXF file, to a disk destination, of the cylindrical geometry.

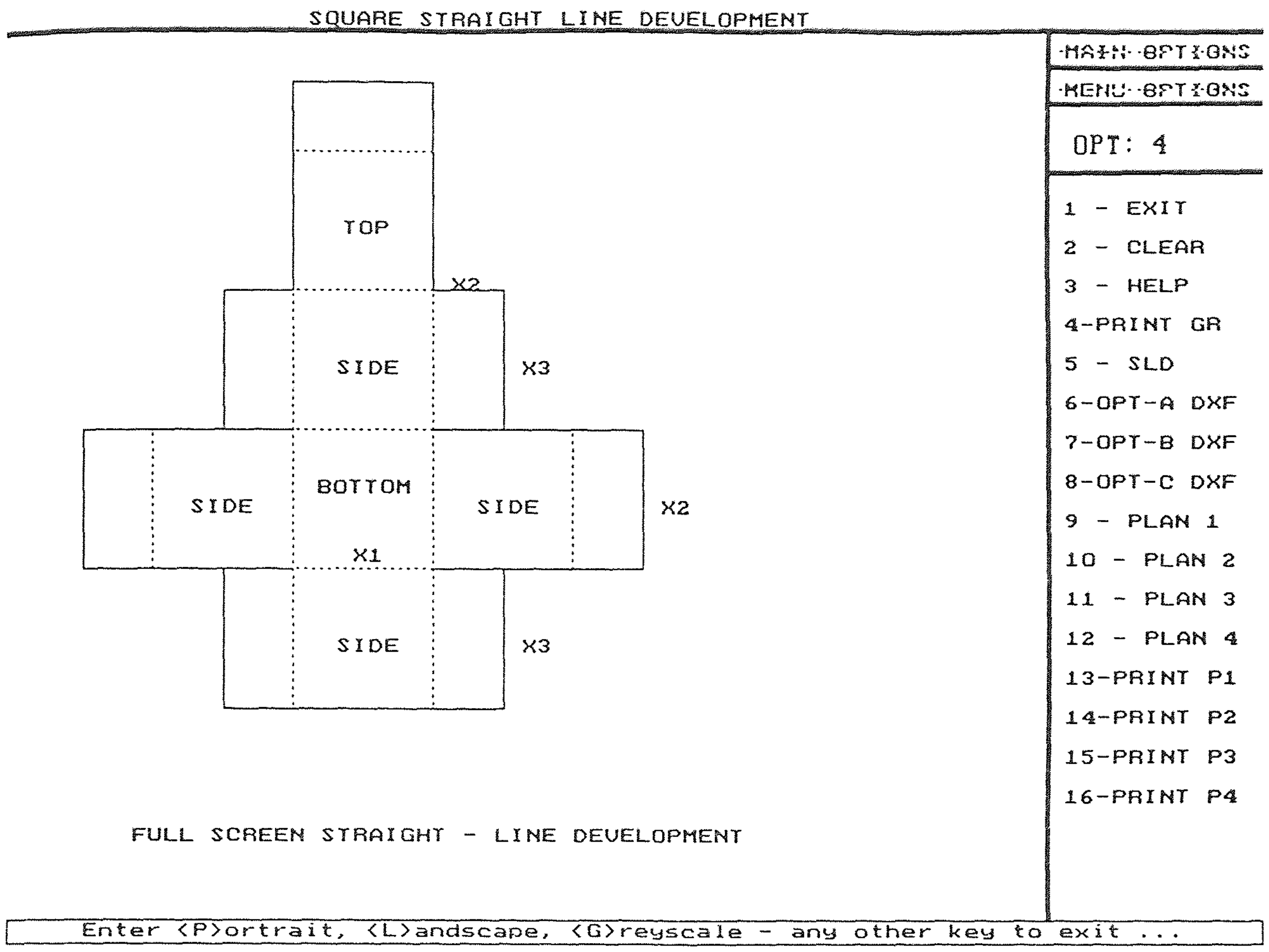
Option 8: A function that will output a DXF file, to a disk destination, of the triangular geometry. (See appendix C for sample DXF output and appendix B for AutoCAD rendering).

Option 9: Provides a new menu screen where process planning information can be entered. This function also provides a 1/4 scale representation of the straight line development to accompany the process information, (See figure 10 on page 87 and appendix B).

Option 10: A function to provide a second process plan from the selected geometry.

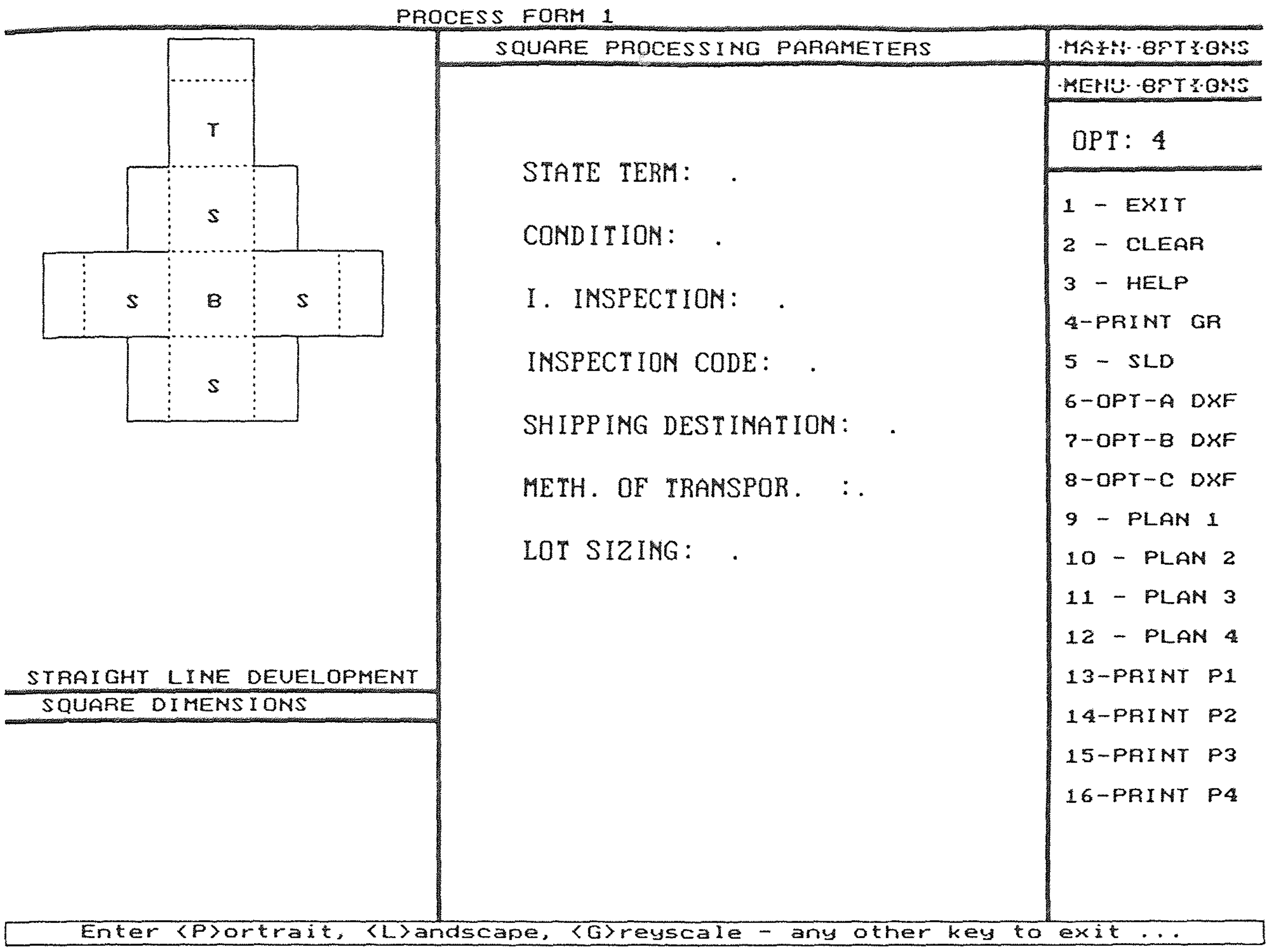
Option 11: A function to provide a third process plan from the selected geometry.

Option 12: A function to provide a fourth process plan



Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...

Figure 9. Full Screen -Generated Square Straight Line Development



STATE TERM: .

CONDITION: .

I. INSPECTION: .

INSPECTION CODE: .

SHIPPING DESTINATION: .

METH. OF TRANSPOR. :.

LOT SIZING: .

Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...

Figure 10. Option 9 -Generated Square Straight Line Development with Form

from the selected geometry.

Option 13: Provides a formal process plan of the straight line development point coordinates.

Option 14: Provides a formal process plan from data entered in option 9, (See code on page 89).

Option 15: Provides a formal process plan from data entered in option 10.

Option 16: Provides a formal process plan from data entered in option 11.

The help menu can also be activated in any module, (See appendix B).

```
                              ABC INC.
                           NEWARK NJ 07737

                            PROCESS FORM 1
 ---------------------------------------------------------------------------

THE GEOMETRY REQUIRED IS:  A

 ---------------------------------------------------------------------------

       THE LENGTH OF X1 IS:  1.000000  INCHES

       THE LENGTH OF X2 IS   1.000000  INCHES

       THE LENGTH OF X3 IS   1.000000  INCHES

 ---------------------------------------------------------------------------
                   STRAIGHT LINE DEVELOPMENT POINT COORDINATES
 ---------------------------------------------------------------------------
        POINT                   X COORD                     Y COORD
  ---------------------------------------------------------------------------
        1                     2.500000                    0.000000
        2                     2.500000                    0.500000
        3                     2.500000                    1.500000
        4                     3.000000                    1.500000
        5                     3.000000                    2.500000
        6                     3.500000                    2.500000
        7                     4.000000                    2.500000
        8                     4.000000                    3.500000
        9                     3.500000                    3.500000
        10                    3.000000                    3.500000
        11                    3.000000                    4.500000
        12                    2.500000                    4.500000
        13                    1.500000                    4.500000
        14                    1.000000                    4.500000
        15                    1.000000                    3.500000
        16                    0.500000                    3.500000
        17                    0.000000                    3.500000
        18                    0.000000                    2.500000
        19                    0.500000                    2.500000
        20                    1.000000                    2.500000
        21                    1.000000                    1.500000
        22                    1.500000                    1.500000
        23                    1.500000                    0.500000
        24                    1.500000                    0.000000
        25                    2.500000                    0.000000
        26                    2.500000                    0.500000
        27                    2.500000                    1.500000
```

Generated Plan of Option "A" Point Coordinates

CHAPTER 7

LINKING THESIS SOFTWARE TO OTHER MANUFACTURING FUNCTIONS

Linking the thesis software and CAD software, such as AutoCAD revision 12, to other manufacturing activities will provide complete flexibility. From start to finish, the production process can be completely automated, (See figure 11 on page 91).

The sequence of events include:

- Thesis software operation. Enter required values to generate process plans and engineering drawings. Create a DXF file to import into AutoCAD 12.
- Utilizing a CAD system (AutoCAD 12). Importing thesis generated DXF files and Utilizing AutoCAD's AME module to create .STL files. .SHP files will also be created using a SmartCAM "add-on" package. These files will be imported into a SmartCAM system. Another CAM package that has the ability to directly accept DXF code is MasterCAM.
- Stereo-Lithography Apparatus. Using the generated .STL file from AutoCAD to generate a prototype of the design. This is completed by importing the .STL file into the SLA controller. A prototype of the design can be created.
- CAM system. Using the generated .SHP file from the

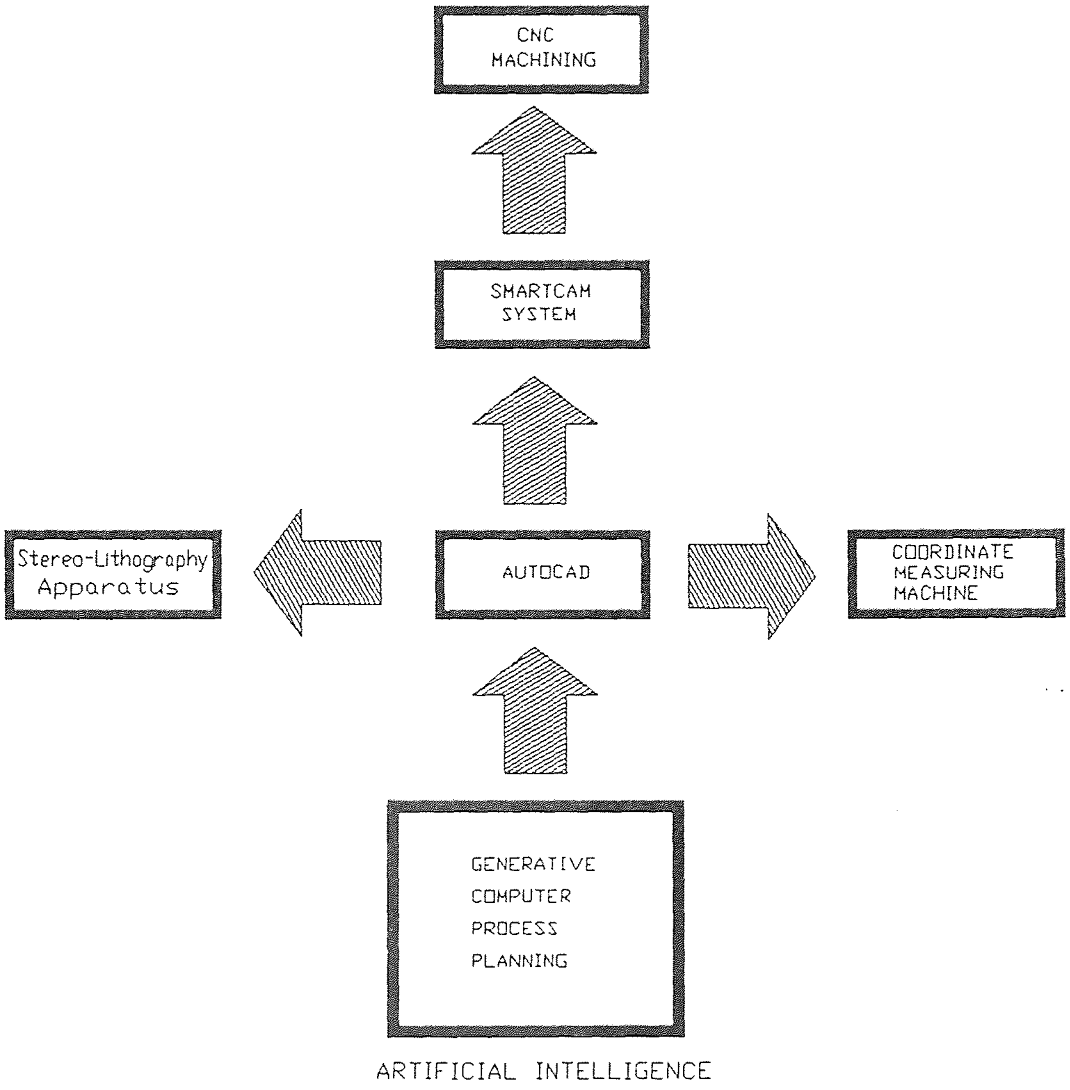

CNC MACHINING
SMARTCAM SYSTEM
Stereo-Lithography Apparatus
AUTOCAD
COORDINATE MEASURING MACHINE
GENERATIVE COMPUTER PROCESS PLANNING
ARTIFICIAL INTELLIGENCE

Figure 11. Linking Thesis Software to other Manufacturing Functions

AutoCAD add-on module to automatically generate numerical control code. This is completed by importing the .SHP file into the SmartCAM system. Also, this is performed by directly importing the DXF file into a MasterCAM system.

- Machine operation. Using the generated NC code and importing it into a CNC controller where the machine will create the actual part. A completed design will result.
- Coordinate Measuring Machine (CMM). Checking the parts tolerances to determine if the part is acceptable and the process was successful.

The complete process begins with importing DXF files, from the thesis software, into a CAD system to utilize its database. This involves entering the required data into the thesis software and utilizing the DXF options in the options menu.

7.1 Importing DXF Files into AutoCAD 12

The purpose of importing DXF (Data Exchange Format) files generated by the thesis software is to further develop the design and to utilize AutoCAD's database and AME (Advanced Modeling Extension) module. Once imported into AutoCAD, straight-line developments may be modified and converted into .STL (Stereo-Lithography extension) files. Another important aspect of importing DXF files into AutoCAD is that .SHP (Shape extension) can be generated with a SmartCAM add-on module. .SHP files can then be imported into a SmartCAM system where NC code will be generated.

DXF files may be created using the thesis software. This format allows for a universal code to be understood from one CAD package to another. Another file format that is universal is IGES (Initial Graphics Exchange Specification). By specifying the particular design option, (Options 6, 7, or 8 in thesis software) a DXF file will be created to a floppy disk or hard drive. In AutoCAD 12, the DXF FILE-IN option is under the FILE pull-down menu. To import the file, simply choose "DXF-IN". At the command prompt, enter the location and name of the file. AutoCAD will then regenerate the design that was generated with the thesis software.

7.2 Stereo-Lithography

The SLA, or Stereo-Lithography Apparatus, is a rapid prototyping device which uses a polymer resin to create part geometries. Other methods that use polycarbonate, nylon, and casting wax are available through Selective Laser Sintering (SLS). The SLA at NJIT is a 3-D systems model 250 that has a volumetric capacity of a nine inch cube. On the average, the time that it takes to build a model is four to five hours (depending on the size of the model). An additional one to three hours is required to cure the completed model in an ultraviolet oven. The SLA is the most popular form of rapid prototyping.

7.2.1 Generating Stereo-Lithography (.STL) Files from Imported Thesis DXF Files

The file format that the SLA controller will accept must have the .STL file extension. This file format is a common output of many CAD packages such as PRO-ENGINEER and I-DEAS, but is not so common with AutoCAD 12. The .STL file will translate the geometry that was developed by a solid modeler into a mosaic-like surface file. The file is usually described by triangles (or facets). The facet density must be high enough to represent the geometric model accurately. Although not widely used to create .STL files, AutoCAD revision 12 does support this option through its AME (Advanced Modeling Extension) module. A set sequence of steps is required for .STL generation through AutoCAD 12.

The thesis software DXF files may be converted into .STL files using the AutoCAD database with the AME module. To import the generated DXF file into AutoCAD, choose FILES, then IMPORT DXF IN. Once the straight line development is regenerated, it is ready to be updated with AutoCAD's AME module. When loaded into the SLA controller, the .STL files are converted into slice files using software that is pre-loaded into the system. The software will slice the faceted geometric model information into Z-layers.

AutoCAD's AME module contains six primitives (a primitive is the basic building block for geometric shapes) to work with as a building block. The primitives given are a cube, wedge, cone, cylinder, sphere, and torus. These basic shapes can be utilized in order be joined and manipulated, in one of three ways, to produce secondary shapes. Manipulations include:

- Unions. This joins two primitives so they act as one object.
- Subtractions. Utilizing one object to cut a shape into another.
- Intersections. Uses only the intersecting region of two different objects to define a solid shape.

The three methods of joining and manipulating solids are better known as Boolean (Boolean comes from the name George Boole, a nineteenth century mathematician) operations. Joined primitives are called composite solids. Composite solids consist of two or more objects that were joined to act as one solid.

Through AutoCAD's AME module, primitives to primitives can be joined, composite solids to primitives, and composite solids to composite solids. When a particular solid is created, such as creating a solid from the imported straight line development DXF file, it must be a wire frame to generate the .STL file. AutoCAD 12 will not be able to create the .STL file if it is solid. The following algorithms are required for .STL generation through AutoCAD's AME module:

Step 1. Load the AME module by selecting:

/MODEL

/SETUP

/DOUBLE PREC

/AUTOLOAD REGION/<AME>

Step 2. Create a wire frame from the imported straight line development DXF file. This is done by:

/MODEL

/PRIMITIVES

/SELECT A SHAPE

/DIMENSIONS

Step 3. To create the .STL file, at the command prompt, enter:

/SOLSTLOUT.

This will prompt AutoCAD to acquire a single solid for .STL output. Two options are given when SOLSTLOUT is activated. The first option is for text output. Text output provides a file that contains a sequence of commands in ASCII format. The second option is for binary output. The major advantage of binary output over text output is that it will reduce the amount of disk space required. This is useful for writing very large .STL files to disk. When all parameters are entered successfully, AutoCAD will prompt a .STL directory. To output the .STL file, enter the filename and choose OK.

The .STL file is satisfactory when all triangles satisfy the vertex to vertex rule with no gaps. The shells of the part should be consistently oriented, should have appropriate volumes, and should be complete. These elements are the basic requirements for successful .STL file generation.

7.3 SLA Processes

When the generated .STL files are loaded into the SLA computer system, pre-loaded software will convert the .STL file into a slice file. The slice file primarily consists of triangular model information that contains Z-axis coordinates and layers (the slices are actually perpendicular to the Z-axis, which is the direction the model will be built).

The SLA utilizes a laser beam to solidify a polymer resin at each layer corresponding to the location of the actual material of the model. At the elevator platform begins the initial layer. After the first layer is built, a second layer is created when the SLA elevator is lowered into the resin. This process will continue until the model is completely built.

7.4 From Thesis Software to CAM

By utilizing the DXF option in the thesis software, A DXF file is created to disk where it can be loaded into AutoCAD 12 for further processing and SmartCAM .SHP file generation or it can be directly loaded into MasterCAM. The procedure used to demonstrate the flexibility of the thesis software was completed on MasterCAM. In MasterCAM, to import a DXF file from the menu options, the following algorithm is as follows:

```
/ MAIN MENU
/ FILE
/ CONVERT
/ DXF
/ READ FILE
```

Once loaded into MasterCAM and the developed design is displayed on the screen, NC code can be generated. The procedure for this is:

```
/TOOLPATHS (Select the toolpath)
/CONTOUR  (Contour milling operation)
/SINGLE   (Select single for each line to be
           milled)
/DONE     (Select DONE after line is selected)
/CONTOUR  (Set the construction plane)
/WRITE    (NCI file creation)
/YES      (Accept the current toolpath)
/END PROGRAM  (Close current operation)
/RUN POST PROCESSOR?..YES
/SPECIFY FILE NAME AND DESTINATION
```

By utilizing the following algorithm, MasterCAM will automatically generate the Numerical Control code to be input into a controller unit, (See code on page 99).

With the SmartCAM system, modeling such as dynamic graphics of tool and machine motion, is possible. By using a sequential tool path database, SmartCAM can cut the model,

```
(PROGRAM NAME - SLD1C )
(DATE, Day-Month-Year - 14-11-94  TIME, Hr:Min - 12:43 )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )
( TOOL - 00   DIA. OFF. - 01   LENGTH - 01   DIA. - .1250   2D-CONTOUR )

%
N100 G00 G40 G49 G80 G90
/ N102 G91 G28 Z0.
/ N104 G28 X0. Y0.
/ N106 G92 X0. Y0. Z0.
N108 T0 M06
N110 G00 G90 X1. Y4.5 S0000 M5
N112 G43 H1 Z0.
N114 G1 X3.
N116 G0 X1. Y3.5
N118 G1 Y4.5
N120 G0 X3. Y3.5
N122 G1 Y4.5
N124 G0 X0. Y2.5
N126 G1 Y3.5
N128 G0 X1. Y1.5
N130 G1 Y2.5
N132 G0 X3. Y1.5
N134 G1 Y2.5
N136 G0 X4.
N138 G1 Y3.5
N140 G0 X1.5 Y0.
N142 G1 X2.5
N144 G0 X1.5 Y1.5
N146 G1 Y0.
N148 G0 X2.5 Y1.5
N150 G1 Y0.
N152 G0 X1. Y1.5
N154 G1 X1.5
N156 G0 X3.
N158 G1 X2.5
N160 G0 X0. Y2.5
N162 G1 X1.
N164 G0 X4.
N166 G1 X3.
N168 G0 X4. Y3.5
N170 G1 X3.
N172 G0 X0.
N174 G1 X1.
N176 M05
N178 G91 G28 Z0.
/ N180 G28 X0. Y0.
N182 G90
N184 M30
%
```

Generated NC Code from Imported DXF File in "real-time", on the computer screen. It is dynamic because if any change is made to the model, it is immediately updated in the database. This eliminates a separation of part geometry and tool path.

In generating NC code with SmartCAM, the first step is to have the completed geometric model loaded into the system. From this point, enter the process menu where the MAIN PROCESS menu is available. Two toolboxes exist, these are ROUGH and CODE. With the CODE option selected, a CODE dialogue box will appear. The code operations will work on all unmasked tool property elements in the sequence that was specified. The SmartCAM system will then generate the NC code for the model and will estimate the cycle time. Following the code generation, a report function will display possible errors.

7.5 Machining with Thesis Software Designs

Once the NC code is generated on the CAM system, CNC machining can take place. The type of machining required may be specified by the designer or by the CAPP system. CNC machining comes in many forms. These include:

- Drill presses.
- Milling machines (vertical spindle and horizontal spindle).
- Turning machines (horizontal axis and vertical axis).
- Horizontal and vertical boring mills.
- Profiling and contouring mills.

- Surface grinders and cylindrical grinders.
- Unconventional machining methods (Waterjet, laser, etc.).

Each machining method listed can be applied to design criteria that a CAPP system will provide.

7.6 Inspecting Thesis Software Designs with a CMM

A coordinate measuring machine (CMM) is a device, used in manufacturing, that measures the surfaces and features of a work-part. The CMM consists of a contact probe that is positioned in three-dimensional space. The contact probe is usually connected to a moving structure where it will "touch" the work-part to provide measurements. The CMM can be controlled manually, manual computer-assisted, motorized computer assisted, or by direct computer control.

Utilizing the CMM will provide a final step in the manufacture of the developed design provided by the AI based CAPP system.

CHAPTER 8

CONCLUSION

With the development of Generative Computer Process Planning, improvement in productivity, design, and planning is a reality. It provides the ability to go from a preliminary design to actual manufacture and inspection. The system will provide complete flexibility while decreasing costs and production time. This is due to CAPP's ability to use pre-programmed engineering knowledge, utilizing artificial intelligence, to automatically generate accurate and effective process plans from the preliminary data that is provided. The ability to interface with outside software packages, such as AutoCAD, MasterCAM, or SmartCAM, is very desirable since various databases can be utilized to provide optimal design.

As computer hardware continues to develop and become more powerful, along with the reduction of hardware costs, the utilization of CAPP systems will become the preferred alternative to manual process planning. With recent developments in artificial intelligence, generative computer process planning will be used to aid various process planning functions involving engineering design and manufacturing processes.

APPENDIX A

AICAPP.CPP SOURCE CODE

```
/* THIS PROGRAM WAS AUTHORED BY WILLIAM TERESHKOVICH DURING
   THE FALL 1994 SEMESTER AT THE NEW JERSEY INSTITUE OF
   TECHNOLOGY. THIS PROGRAM COMBINES THE USE OF ARTIFICIAL
   INTELLIGENCE (AI) WITH A GENERATIVE CAPP SYSTEM TO
   PROCESS THREE DIMENSIONAL GEOMETRIC SHAPES INTO STRAIGHT
   LINE DEVELOPMENTS. THE PRIMARY FOCUS IS DEVELOP STRAIGHT
   LINE GEOMETRY THAT CAN INTERFACE WITH CAD SYSTEMS,
   PROVIDE GRAPHICAL REPRESENTATIONS, AND PRODUCE
   INTELLIGENT PROCESS PLANNING. */

/*-------------------------------------------------------*/


#include <stdio.h>
#include <graphics.h>
#include <conio.h>
#include <stdlib.h>
#include <math.h>
#include <CTYPE.H>
#include <string.h>



/*-------------------------------------------------------*/


void main(void);              /* DEFINE GLOBAL VARIABLES */
void initializegr(void);      /* graphics function */
void introduction(void);      /* introduction function */
void PromptLine(void);        /* laser printer function */
void dxfmaker(void);          /* autocad dxf function */
void dxfparameter1(void);     /* autocad dxf function */
void dxfparameter2(void);     /* autocad dxf function */
void dxfparameter3(void);     /* autocad dxf function */
void menu1(void);             /* general menu function */
void menu2(void);             /* interface menu */
void Print_Pause(int);        /* laserprinter function */
void straightsquare(void);    /* straight-square function */
void straightcylinder(void); /* straight-cylinder function*/
void straighttriangle(void); /* straight-triangle function*/
void drawsqfull(void);        /* draw full screen square */
void exit(void);             /* end graphics function */
void drawtrifull(void);      /* draw full screen tri-shape*/
void drawcylfull(void);      /* draw full screen cylinder */
void options(void);          /* option menu function */
void optionsgr(void);        /* options graphics function */
void helpmenu(void);         /* help function */
void form1print(void);       /* form 1 print function */
void form2A(void);           /* form2A graphics function */


/* global variables */

int backgrcol, menuoptions, GraphDriver, GraphMode, cx[50],
    cy[50];
float x[50], y[50];
char geometry;
float X1, X2, X3;

#define BEEP printf("\a \n")


/*----------------------------------------------------------*/

   void main(void)    /* main controller program */
   {
     initializegr();  /* graphics compatability */
     introduction();  /* begin greetings screen */


   }                  /* terminiate intro function */

 /*---------------------------------------------------------*/


  void initializegr(void)  /* inititilize program graphics
                              for use with BGI files */

  {
   /* detect graphics and graphics driver automatically */


   int gdriver = DETECT, graphics1, loadgr;
   int backgrcol;

     /* begin graphics capability, define all locals */

   initgraph(&gdriver, &graphics1, "");
   loadgr = graphresult();
   if (loadgr != grOk)      /* unsucessful graphics */

      {
      printf("GRAPHICS INCOMPATABILITY: %s\n",
              grapherrormsg(loadgr));
      printf("PRESS A KEY TO STOP:");
    getch();
    exit(1);
    backgrcol = 0;
    setbkcolor( backgrcol );

    /* stop and write error message */
    }

  setcolor(getmaxcolor());

  }

/*----------------------------------------------------------*/

  void introduction(void) /* begin intro graphics screen */

  {

   backgrcol = 8;                /* set background */
   setbkcolor( backgrcol );
   settextstyle(1, 0, 2);        /* intro screen text */
   setcolor( 58 );
   outtextxy(185, 75, "GENERATIVE COMPUTER" );
   outtextxy(205, 95,   "PROCESS PLANNING" );
   settextstyle(0, 0, 1);
   setcolor( 63 );
   outtextxy(140, 300, "Utilizing Artificial Intelligence
             To Generate");
   outtextxy(140, 320, "Straight Line Developments,
             Process Plans, ");
   outtextxy(140, 340, "And AutoCAD Interfacing" );
   setcolor( 58 );
   outtextxy(220, 420, "--HIT A KEY TO START--");
   setfillstyle(1, 4);
   bar(0, 20, 640, 20);
   bar(0, 450, 640, 450);

   getch();                     /* wait for keystroke */
   cleardevice();
   menu1();                     /* call menu1 */
  }                             /* end intro screen */

 /*---------------------------------------------------------*/

   void menu1(void)             /* general menu function */

  {

   optionsgr();                /* call graphics function */

   backgrcol = 8;
                              /* background selection */
   setbkcolor(backgrcol);
   setfillstyle(0, 63);
```

```
setcolor( 63 );
setlinestyle(0, 0, 2);
bar3d(30, 70, 105, 140, 17, 1);       /* 3-D box */
setlinestyle( 3, 3, 1 );
line( 46.0208, 57.9792, 46.0208, 127.9792 );
line( 30, 140, 46.0208, 127.9792 );
line( 46.0208, 127.9792, 117.0208, 127.9792 );


line(75, 270, 110, 270);            /* corner cyl */
line(75, 190, 110, 190);
setlinestyle( 0, 0, 1 );
ellipse( 75, 190, 0, 360, 35, 8 );  /* draw cylinder */
ellipse( 75, 270, 0, 360, 35, 8 );
line( 40, 190, 40, 270 );
line( 110, 190, 110, 270 );


line( 40, 330, 110, 330 );         /* draw tri-shape */
line( 40, 330, 75, 345 );
line( 75, 345, 110, 330 );
line( 40, 330, 40, 410 );
line( 40, 410, 75, 425 );
line( 75, 425, 110, 410 );
line( 75, 345, 75, 425 );
line( 110, 330, 110, 410 );
setlinestyle( 3, 3, 1 );
line( 40, 410, 110, 410 );

setcolor( 62 );                    /* top text */
settextstyle( 0, 0, 1 );
outtextxy(155, 5, "GENERATIVE COMPUTER PROCESS
          PLANNING");
setlinestyle( 0, 0, 1 );           /* partition */
setcolor( 4 );
setlinestyle( 0, 0, 3 );

line( 0, 14, 640, 14 );            /* menu partitions */
line( 240, 14, 240, 480 );
line( 530, 14, 530, 480 );
line( 0, 32, 530, 32 );

setcolor( 62 );
outtextxy( 50, 21, "AVAILABILE GEOMETRY" );
outtextxy( 360, 21, "MAIN" );

setcolor( 62 );
outtextxy( 15, 105, "X2");   /* box dimensions */
outtextxy( 65, 145, "X1");
outtextxy( 120, 133, "X3");

outtextxy( 115, 190, "X1");  /* cylinder dimensions */
outtextxy( 94, 232, "X3");
outtextxy( 115, 270, "X2");
```

```
   outtextxy( 25, 370, "X1");   /* tri-shape dimensions */
   outtextxy( 50, 343, "X2");
   outtextxy( 70, 320, "X3");

   setcolor( 58 );
   outtextxy( 140, 90, "OPTION A");     /* option text */
   outtextxy( 140, 220, "OPTION B");
   outtextxy( 140, 370, "OPTION C");

   setcolor( 60 );                  /* option underlining */
   setlinestyle( 0, 0, 1 );
   line( 138, 103, 205, 103 );
   line( 138, 233, 205, 233 );
   line( 138, 383, 205, 383 );

   menu2();                        /* call menu2 function */


}                                  /* close menu1 */


/*-------------------------------------------------------*/

   void menu2(void)                      /* open menu2 */

{

   setcolor( 62 );
   outtextxy(265, 50, "GEOMETRY");
   setcolor( 60 );
   line(263, 60, 330, 60);
   setcolor( 62 );
   window(34, 6, 36, 8);
   printf("SELECT REQUIRED GEOMETRY :  ");
   scanf(" %c", &geometry);

   switch(toupper(geometry))        /* switch cases */
   {

   case 'A':                        /* case A switch */
   window(34, 7, 36, 9);
   textcolor( LIGHTCYAN );
   printf("SQUARE GEOMETRY MODE");
   outtextxy(265, 135, "SQUARE DIMENSIONS");
   setcolor( 60 );
   line( 263, 145, 400, 145 );
   setcolor( 62 );
   window(34, 11, 36, 13);
   printf("ENTER X1:  ");           /* scan enter */
   scanf(" %f", &X1);
   window(34, 12, 36, 14);
   printf("ENTER X2:  ");
   scanf(" %f", &X2);
window(34, 13, 36, 15);
printf("ENTER X3:  ");
scanf(" %f", &X3);
break;

case 'B':                          /* case B switch */
window(34, 7, 36, 9);
printf("CYLINDRICAL GEOMETRY MODE");
outtextxy(265, 135, "CYLINDER DIMENSIONS");
setcolor( 60 );
line( 263, 145, 415, 145 );
setcolor( 62 );
window(34, 11, 36, 12);
printf("ENTER X1:  ");             /* scan enter */
scanf(" %f", &X1);
window(34, 12, 36, 14);
printf("ENTER X2:  ");
scanf(" %f", &X2);
window(34, 13, 36, 15);
printf("ENTER X3:  ");
scanf(" %f", &X3);
break;

case 'C':                          /* case C switch */
window(34, 7, 36, 9);
printf("TRIANGULAR GEOMETRY MODE");
outtextxy(265, 135, "TRIANGULAR DIMENSIONS");
setcolor( 60 );
line( 263, 145, 430, 145 );
setcolor( 62 );
window(34, 11, 36, 13);
printf("ENTER X1:  ");             /* scan enter */
scanf(" %f", &X1);
window(34, 12, 36, 14);
printf("ENTER X2:  ");
scanf(" %f", &X2);
window(34, 13, 36, 15);
printf("ENTER X3:  ");
scanf(" %f", &X3);
break;

default:                        /* default function */

outtextxy( 265, 150, "INVALID GEOMETRY
            SPECIFICATION...");
outtextxy( 265, 170, "PRESS ANY KEY TO CLEAR...");
getch();
cleardevice();                  /* clear screen */
menu1();                        /* call menu1 function */
break;


}                                /* close switch cases */
```

```
    window(34, 15, 36, 17);        /* output dimensions */
    printf("X1 = %f IN. VALID...", X1);
    window(34, 16, 36, 18);
    printf("X2 = %f IN. VALID...", X2);
    window(34, 17, 36, 19);
    printf("X3 = %f IN. VALID...", X3);

    outtextxy(265, 300, "MENU OPTIONS"); /* menu options */
    setcolor( 60 );
    line( 263, 310, 360, 310);
    setcolor( 62 );
    window(34, 21, 36, 23);
    printf("MENU OPTION MODE");
    window(34, 22, 36, 24);
    printf("ENTER MENU OPTIONS");


     options();                   /* call options function */

  }

/*-------------------------------------------------------*/

  void options(void)              /* begin options function */
  {

    window(69, 5, 71, 7);
    printf("OPT: ");
    scanf(" %d", &menuoptions);  /* scan options */

    switch(menuoptions)           /* menu options switches */

    {
       case 1:                      /* exit case */
       exit();
       break;

       case 2:                      /* clear case */
       cleardevice();
       menu1();
       break;

       case 3:                      /* help case */
       helpmenu();

       break;

       case 4:                      /* print graphics case */
       Print_Pause(0);

       break;

       case 5:                      /* full screen case */
```

```
        if(toupper(geometry)=='A')
        straightsquare();
        if(toupper(geometry)=='B')
        straightcylinder();
        if(toupper(geometry)=='C')
        straighttriangle();

    break;

    case 6:                          /* dxf case 1 */
        straightsquare();
        dxfmaker();
        dxfparameter1();

    break;

    case 7:                          /* dxf case 2 */
       straightcylinder();
       dxfmaker();
       dxfparameter2();

    break;
                                    /* dxf case 3 */
    case 8:
      straighttriangle();
      dxfmaker();
      dxfparameter3();

    case 9:                          /* form 1 case */
        if(toupper(geometry)=='A')
        straightsquare();
        if(toupper(geometry)=='B')
        straightcylinder();
        if(toupper(geometry)=='C')
        straighttriangle();

    break;

    case 13:                         /* form1 print case */
       if(toupper(geometry)=='A')
       straightsquare();
       form1print();
       if(toupper(geometry)=='B')
       straightcylinder();
       form1print();
       if(toupper(geometry)=='C')
       straighttriangle();
       form1print();

    break;

    case 14:                          /* form 2 case */
```

```
         break;                       /* open case */

         case 15:                     /* form 3 case */
             form1print();            /* call function */

         break;                       /* open case */

         case 16:                     /* form 4 case */
             form1print();            /* call function */

         break;


      }

      getch();                       /* wait for keystroke */

   }                                 /* close menu2 */


   /*-----------------------------------------------------*/

      void optionsgr(void)    /* begin optionsgr function */

    {
      setcolor( 4 );
      setlinestyle(0, 0, 3);
      line( 0, 14, 640, 14 );         /* menu partitions */
      line( 530, 14, 530, 480 );
      line( 530, 32, 640, 32 );
      setcolor( 62 );
      outtextxy( 539, 21, "MAIN OPTIONS" );

      setcolor( 58 );                         /* option text */
      outtextxy( 539, 100, "1 - EXIT" );
      outtextxy( 539, 120, "2 - CLEAR");
      outtextxy( 539, 140, "3 - HELP");
      outtextxy( 539, 160, "4-PRINT GR");
      outtextxy( 539, 180, "5 - SLD" );
      outtextxy( 539, 200, "6-OPT-A DXF OUT");
      outtextxy( 539, 220, "7-OPT-B DXF OUT");
      outtextxy( 539, 240, "8-OPT-C DXF OUT");
      outtextxy( 539, 260, "9 - PLAN 1");
      outtextxy( 539, 280, "10 - PLAN 2");
      outtextxy( 539, 300, "11 - PLAN 3");
      outtextxy( 539, 320, "12 - PLAN 4");
      outtextxy( 539, 340, "13-PRINT P1");
      outtextxy( 539, 360, "14-PRINT P2");
      outtextxy( 539, 380, "15-PRINT P3");
      outtextxy( 539, 400, "16-PRINT P4");

      setcolor( 63 );
      outtextxy(539, 40, "MENU OPTIONS");  /* menu options */
      setcolor( 4 );
    setlinestyle( 0, 0, 3 );
    line( 530, 50, 640, 50 );
    line( 530, 85, 640, 85 );
    setcolor( 62 );

    setlinestyle(0, 0, 0);          /* change line form */


  }

/*-----------------------------------------------------------*/

    void straightsquare( void )   /* begin straightsq.
                                     function */
  {
    int c;
    double point1x, point1y, point2x, point2y, doubx,
           douby;
    float maximum, scalor;

    /* point array 50 x 50 matrix */

    x[0]=1.0/2.0*X1+X3, x[1]=3.0/2.0*X1+X3;
    x[2]=3.0/2.0*X1+X3; x[3]=3.0/2.0*X1+X3;
    x[4]=3.0/2.0*X1+1.0/2.0*X2+X3;
    x[5]=3.0/2.0*X1+1.0/2.0*X2+X3, x[6]=3.0/2.0*X1+2.0*X3;
    x[7]=2.0*X1+2.0*X3, x[8]=2.0*X1+2.0*X3;
    x[9]=3.0/2.0*X1+2.0*X3, x[10]=3.0/2.0*X1+1.0/2.0*X2+X3;
    x[11]=3.0/2.0*X1+1.0/2.0*X2+X3;
    x[12]=3.0/2.0*X1+X3, x[13]=1.0/2.0*X1+X3;
    x[14]=1.0/2.0*X1-1.0/2.0*X2+X3;
    x[15]=1.0/2.0*X1-1.0/2.0*X2+X3;
    x[16]=1.0/2.0*X1, x[17]=0.0, x[18]=0.0;
    x[19]=1.0/2.0*X1, x[20]=1.0/2.0*X1-1.0/2.0*X2+X3;
    x[21]=1.0/2.0*X1-1.0/2.0*X2+X3;
    x[22]=1.0/2.0*X1+X3, x[23]=1.0/2.0*X1+X3;
    x[24]=1.0/2.0*X1+X3, x[25]=1.0/2.0*X1+X3;
    x[26]=1.0/2.0*X1+X3, x[27]=3.0/2.0*X1+X3;
    x[28]=3.0/2.0*X1+X3, x[29]=1.0/2.0*X1+1.0/2.0*X3;
    x[30]=X1+X3, x[31]=3.0/2.0*X1+3.0/2.0*X3, x[32]=X1+X3;
    x[33]=X1+X3, x[34]=X1+X3, x[35]=0.0, x[36]=0.0;
    x[37]=0.0, x[38]=0.0, x[39]=0.0, x[40]=0.0;
    x[41]=0.0, x[42]=0.0, x[43]=0.0, x[44]=0.0, x[45]=0.0;
    x[46]=0.0, x[47]=0.0, x[48]=0.0, x[49]=X1+X3;

    y[0]=0.0, y[1]=0.0, y[2]=1.0/2.0*X2, y[3]=3.0/2.0*X2;
    y[4]=3.0/2.0*X2, y[5]=3.0/2.0*X2+X3;
    y[6]=3.0/2.0*X2+X3, y[7]=3.0/2.0*X2+X3;
    y[8]=5.0/2.0*X2+X3, y[9]=5.0/2.0*X2+X3;
    y[10]=5.0/2.0*X2+X3, y[11]=5.0/2.0*X2+2*X3;
    y[12]=5.0/2.0*X2+2*X3, y[13]=5.0/2.0*X2+2*X3;
    y[14]=5.0/2.0*X2+2*X3, y[15]=5.0/2.0*X2+X3;
    y[16]=5.0/2.0*X2+X3, y[17]=5.0/2.0*X2+X3;
    y[18]=3.0/2.0*X2+X3, y[19]=3.0/2.0*X2+X3;
```

```
   y[20]=3.0/2.0*X2+X3, y[21]=3.0/2.0*X2;
   y[22]=3.0/2.0*X2, y[23]=1.0/2.0*X2, y[24]=0.0;
   y[25]=3.0/2.0*X2+X3, y[26]=5.0/2.0*X2+X3;
   y[27]=3.0/2.0*X2+X3, y[28]=5.0/2.0*X2+X3;
   y[29]=2*X2+X3, y[30]=2*X2+X3, y[31]=2*X2+X3;
   y[32]=3.0/2.0*X2+1.0/2.0*X3;
   y[33]=5.0/2.0*X2+3.0/2.0*X3;
   y[34]=X2, y[35]=0.0, y[36]=0.0, y[37]=0.0, y[38]=0.0;
   y[39]=0.0, y[40]=0.0, y[41]=0.0, y[42]=0.0, y[43]=0.0;
   y[44]=0.0, y[45]=0.0, y[46]=0.0, y[47]=0.0, y[48]=0.0;
   y[49]=5.0/4.0*X2+X3;

   /* scaling to full screen */

   if(menuoptions==5)                  /* full sld option */

  {

   double point3x, point4x;
   modf((double) ((2*getmaxx())/3), &point3x);
   modf((double) ((2*getmaxy())/3), &point4x);

   if((2.0*X1+2.0*X3)>(2.0*X3+(5.0/2.0)*X2))
        maximum=(2.0*X1+2.0*X3);
   else maximum=(2.0*X3+(5.0/2.0)*X2);

   if((2.0*X1+2.0*X3)>(2.0*X3+(5.0/2.0)*X3))
        scalor=point3x/maximum;
   else scalor=point4x/maximum;

   for(c=0;c<=49; c++)             /* screen scale */
   {
   x[c]=x[c]*scalor;
   y[c]=y[c]*scalor;
   doubx=(double) x[c];
   douby=(double) y[c];
   point1x=modf(doubx, &point2x);
   point1y=modf(douby, &point2y);

   if (point1x>=0.5) cx[c]=(int) point2x+41;
   else cx[c]=(int) point2x+40;

   if (point1y>=0.5) cy[c]=(int) point2y+41;
   else cy[c]=(int) point2y+40;
    }


   drawsqfull();               /* call the draw function */

 }

   if(menuoptions==9)                  /* form sld if */

   {
```

```
int backcolor, squareshape[50], d;

/* scaling to corner screen */

double point3x, point4x;

if((2.0*X1+2.0*X3)>(2.0*X3+(5.0/2.0)*X2))
     maximum=(2.0*X1+2.0*X3);
else maximum=(2.0*X3+(5.0/2.0)*X2);

scalor=((float)((getmaxx()/3)-20))/maximum;

 for(c=0;c<=49; c++)
 {
x[c]=x[c]*scalor;
y[c]=y[c]*scalor;
doubx=(double) x[c];
douby=(double) y[c];
point1x=modf(doubx, &point2x);
point1y=modf(douby, &point2y);

if (point1x>=0.5) cx[c]=(int) point2x+21;
else cx[c]=(int) point2x+20;

if (point1y>=0.5) cy[c]=(int) point2y+21;
else cy[c]=(int) point2y+20;

 }

setlinestyle(0, 0, 0);
cleardevice();
backcolor = 8;
setbkcolor( backcolor );
setcolor( 63 );

for(d=0;d<=48;d=d+2) squareshape[d]=cx[d/2];
for(d=1;d<=49;d=d+2) squareshape[d]=cy[(d-1)/2];

drawpoly(25, squareshape);

setlinestyle(0, 0, 0);                  /* draw lines */
line(cx[20], cy[20], cx[25], cy[25]);
line(cx[15], cy[15], cx[26], cy[26]);
line(cx[27], cy[27], cx[5], cy[5]);
line(cx[28], cy[28], cx[10], cy[10]);
setlinestyle(1, 0, 1);
line(cx[19], cy[19], cx[16], cy[16]);
line(cx[6], cy[6], cx[9], cy[9]);
line(cx[23], cy[23], cx[2], cy[2]);
line(cx[22], cy[22], cx[3], cy[3]);
line(cx[25], cy[25], cx[27], cy[27]);
line(cx[26], cy[26], cx[28], cy[28]);
line(cx[22], cy[22], cx[13], cy[13]);
```

```
    line(cx[3], cy[3], cx[12], cy[12]);

    setlinestyle(0, 0, 3);
    setcolor( 4 );
    line(220, 14, 220, 480);
    line(220, 32, 530, 32);

    line(0, 350, 220, 350);
    line(0, 365, 220, 365);

    setcolor( 62 );
    outtextxy( 20, 354, "SQUARE DIMENSIONS");

    outtextxy( 250, 21, "SQUARE PROCESSING PARAMETERS");
    setcolor( 63 );
    outtextxy(12, 340, "STRAIGHT LINE DEVELOPMENT");


    setlinestyle(0, 0, 0);

    setcolor( 58 );                      /* geometry text */
    outtextxy(cx[33]-2, cy[33], "S" );
    outtextxy(cx[34]-2, cy[34], "T" );
    outtextxy(cx[31], cy[31], "S" );
    outtextxy(cx[32]-2, cy[32], "S" );
    outtextxy(cx[29], cy[29], "S" );
    outtextxy(cx[30]-2, cy[30], "B" );

    setcolor( 62 );
    outtextxy(200, 5, "PROCESS FORM 1");

    optionsgr();
    form2A();
    options();

    }                        /* close case 9 if statement */



  }                          /* end straightsq. function */


/*-------------------------------------------------------*/

    void straightcylinder( void )    /* begin straightcyl.
                                        function */
  {

    int c;
    double point1x, point1y, point2x, point2y, doubx,
           douby;
    float maximum, scalor;
    int p = 3.14;
```

```
/* 18 x 18 x-y array */


x[0]=0.0, x[1]=2.0/5.0*X1*p, x[2]=4.0/5.0*X1*p;
x[3]=6.0/5.0*X1*p; x[4]=8.0/5.0*X1*p;
x[5]=10.0/5.0*X1*p, x[6]=11.0/5.0*X1*p;
x[7]=11.0/5.0*X1*p, x[8]=10.0/5.0*X1*p;
x[9]=8.0/5.0*X1*p, x[10]=6.0/5.0*X1*p;
x[11]=4.0/5.0*X1*p, x[12]=2.0/5.0*X1*p, x[13]=0.0;
x[14]=0.0, x[15]=2.0/5.0*X1*p;
x[16]=8.0/5.0*X1*p, x[17]=2.0/5.0*X1*p+X1;
x[18]=8.0/5.0*X1*p+X1;


y[0]=0.0, y[1]=0.0, y[2]=0.0, y[3]=0.0, y[4]=0.0;
y[5]=0.0, y[6]=0.0, y[7]=X3, y[8]=X3, y[9]=X3;
y[10]=X3, y[11]=X3, y[12]=X3;
y[13]=X3, y[14]=0.0, y[15]=X1+X3, y[16]=X1+X3;
y[17]=3.0/2.0*X3+X1, y[18]=3.0/2.0*X3+X1;


if(menuoptions==5)                    /* full sld if */

{

/* scaling to full screen */

double point3x, point4x;
modf((double) ((12*getmaxx())/13), &point3x);
modf((double) ((12*getmaxy())/13), &point4x);

if(((3.0/2.0)*X3+2.0*X1)>((11.0/5.0)*X1*p))
     maximum=(3.0/2.0)*X3+2*X1;
else maximum=(11.0/5.0)*X1*p;

if(((3.0/2.0)*X3+2.0*X1)>((11.0/5.0)*X1*p))
     scalor=point3x/maximum;
else scalor=point4x/maximum;

for(c=0;c<=18; c++)           /* scaling */
{
x[c]=x[c]*scalor;
y[c]=y[c]*scalor;
doubx=(double) x[c];
douby=(double) y[c];
point1x=modf(doubx, &point2x);
point1y=modf(douby, &point2y);

if (point1x>=0.5) cx[c]=(int) point2x+41;
else cx[c]=(int) point2x+40;

if (point1y>=0.5) cy[c]=(int) point2y+41;
```

```
   else cy[c]=(int) point2y+40;
    }


  drawcylfull();                /* call drawcylfunction */

}                               /* end option 5 if */


 if(menuoptions==9);             /* begin option 9 if */

{

         /* scaling to corner screen */

   double point3x, point4x;

   if((2.0*X1+2.0*X3)>(2.0*X3+(5.0/2.0)*X2))
        maximum=(2.0*X1+2.0*X3);
   else maximum=(2.0*X3+(5.0/2.0)*X2);

   scalor=((float)((getmaxx()/4)-20))/maximum;

   for(c=0;c<=18; c++)                /* set scale */
   {
   x[c]=x[c]*scalor;
   y[c]=y[c]*scalor;
   doubx=(double) x[c];
   douby=(double) y[c];
   point1x=modf(doubx, &point2x);
   point1y=modf(douby, &point2y);

   if (point1x>=0.5) cx[c]=(int) point2x+21;
   else cx[c]=(int) point2x+20;

   if (point1y>=0.5) cy[c]=(int) point2y+21;
   else cy[c]=(int) point2y+20;
    }


   int backcolor, cylinder[50], d;     /* color define */
   setlinestyle(0, 0, 0);
   cleardevice();
   backcolor = 8;
   setbkcolor( backcolor );
   setcolor( 63 );

   for(d=0;d<=28;d=d+2) cylinder[d]=cx[d/2];
   for(d=1;d<=29;d=d+2) cylinder[d]=cy[(d-1)/2];

   drawpoly(15, cylinder);        /* draw poly function */

   setlinestyle(1, 0, 1);                 /* draw lines */
   line(cx[1], cy[1], cx[12], cy[12]);
   line(cx[2], cy[2], cx[11], cy[11]);
   line(cx[3], cy[3], cx[10], cy[10]);
   line(cx[4], cy[4], cx[9], cy[9]);
   line(cx[5], cy[5], cx[8], cy[8]);

   ellipse(cx[15], cy[15]+50, 0, 360, (cx[17]-cx[15]),
          (cx[17]-cx[15]));
   ellipse(cx[16], cy[16]+50, 0, 360, (cx[18]-cx[16]),
          (cx[18]-cx[16]));


   setlinestyle(0, 0, 3);          /* set line style */
   setcolor( 4 );                  /* color */
   line(220, 14, 220, 480);
   line(220, 32, 530, 32);

   line(0, 350, 220, 350);
   line(0, 365, 220, 365);         /* design text */

   setcolor( 62 );
   outtextxy( 20, 354, "CYLINDER DIMENSIONS");

   outtextxy( 250, 21, "CYLDR. PROCESSING PARAMETERS");
   setcolor( 63 );
   outtextxy(12, 340, "STRAIGHT LINE DEVELOPMENT");



 setlinestyle(0, 0, 0);            /* new line style */


 setcolor( 62 );                   /* change color */
 outtextxy(200, 5, "PROCESS FORM 1");

 optionsgr();                      /* call graphics */
 options();                        /* call options */

}                                  /* close option 9 if */




 }                         /* end straightcyl. function */


/*----------------------------------------------------------*/

   void straighttriangle( void )     /* begin straighttr.
                                        function */

 {

   double G=2*asin((double) (X3/(2*X2)));
   int c;
```

```
  double point1x, point1y, point2x, point2y, doubx,
         douby;
  float maximum, scalor;
  float D = (sin(G/2.0)/(3.0))*X1;
  float E = (cos(G/2.0)/(3.0))*X1;
  float F = (cos(G/2.0))*X1;



/* point array 25 x 25 matrix */

  x[0]=0.0, x[1]=1.0/2.0*X3, x[2]=1.0/2.0*X3+X2;
  x[3]=1.0/2.0*X3+2.0*X2; x[4]=1.0/2.0*X3+2*X2-E;
  x[5]=2*X2+X3-D-(cos(G)/3.0)*X1*(sin(G/2.0));
  x[6]=2*X2+X3;
  x[7]=2*X2+X3+D+(cos(G)/3.0)*X1*(sin(G/2.0));
  x[8]=(3.0/2.0)*X3+2*X2+E;
  x[9]=(3.0/2.0)*X3+2*X2, x[10]=(3.0/2.0)*X3+2*X2;
  x[11]=(3.0/2.0)*X3+2*X2+E;
  x[12]=2*X2+X3+D+(cos(G)/3.0)*X1*(sin(G/2.0));
  x[13]=2*X2+X3, x[14]=2*X2+X3-D-
        (cos(G)/3.0)*X1*(sin(G/2.0));
  x[15]=(1.0/2.0)*X3+2*X2-E, x[16]=(1.0/2.0)*X3+2*X2;
  x[17]=(1.0/2.0)*X3+X2, x[18]=(1.0/2.0)*X3, x[19]=0.0;
  x[20]=0.0, x[21]=0.0, x[22]=0.0, x[23]=0.0, x[24]=0.0;


  y[0]=E+F, y[1]=E+F, y[2]=E+F, y[3]=E+F, y[4]=E+F-D;
  y[5]=(cos(G)/(3.0))*X1*(cos(G/2.0)), y[6]=E;
  y[7]=(cos(G)/(3.0))*X1*(cos(G/2.0)), y[8]=E+F-D;
  y[9]=E+F, y[10]=E+F+X1, y[11]=E+F+X1+D;
  y[12]=2*E+2*F+X1-(cos(G)/(3.0))*X1*(cos(G/2.0));
  y[13]=E+2*F+X1, y[14]=2*E+2*F+X1-
       (cos(G)/(3.0))*X1*(cos(G/2.0));
  y[15]=E+F+X1+D, y[16]=E+F+X1, y[17]=E+F+X1;
  y[18]=E+F+X1, y[19]=E+F+X1, y[20]=E+F, y[21]=0.0;
  y[22]=0.0, y[23]=0.0, y[24]=0.0;


  if(menuoptions==5)                 /* option 5 if */

 {
  /* scaling to full screen */

  double point3x, point4x;
  modf((double) ((8*getmaxx())/9), &point3x);
  modf((double) ((8*getmaxy())/9), &point4x);
  float u=2*cos(G/2.0)*((1.0/3.0)*X1+X2)+X1;
  float v=(cos(G/2.0)/(3.0))*X1+(3.0/2.0)*X3+2*X2;

  if(u > v) maximum = u;
  else maximum = v;

  if(u > v) scalor=point3x/maximum;
  else scalor=point4x/maximum;

  for(c=0;c<=20; c++)             /* set scale */
  {
  x[c]=x[c]*scalor;
  y[c]=y[c]*scalor;
  doubx=(double) x[c];
  douby=(double) y[c];
  point1x=modf(doubx, &point2x);
  point1y=modf(douby, &point2y);

  if (point1x>=0.5) cx[c]=(int) point2x+41;
  else cx[c]=(int) point2x+40;

  if (point1y>=0.5) cy[c]=(int) point2y+41;
  else cy[c]=(int) point2y+40;
   }

   drawtrifull();         /* call drawtrifull function */

}                         /* end option 5 if */


   if(menuoptions==9)            /* begin option 9 if */

  {


    /* scaling to corner screen */

  double point3x, point4x;

  if((2.0*X1+2.0*X3)>(2.0*X3+(5.0/2.0)*X2))
       maximum=(2.0*X1+2.0*X3);
  else maximum=(2.0*X3+(5.0/2.0)*X2);

  scalor=((float)((getmaxx()/3)-20))/maximum;

  for(c=0;c<=20; c++)            /* set scale */
  {
  x[c]=x[c]*scalor;
  y[c]=y[c]*scalor;
  doubx=(double) x[c];
  douby=(double) y[c];
  point1x=modf(doubx, &point2x);
  point1y=modf(douby, &point2y);

  if (point1x>=0.5) cx[c]=(int) point2x+21;
  else cx[c]=(int) point2x+20;

  if (point1y>=0.5) cy[c]=(int) point2y+21;
  else cy[c]=(int) point2y+20;
   }
```

```
    int backcolor, trishape[50], d;
    setlinestyle(0, 0, 0);              /* set line */
    cleardevice();
    backcolor = 8;                      /* set color */
    setbkcolor( backcolor );
    setcolor( 63 );                     /* set color */

    for(d=0;d<=40;d=d+2) trishape[d]=cx[d/2];
    for(d=1;d<=41;d=d+2) trishape[d]=cy[(d-1)/2];

    drawpoly(21, trishape);            /* draw shape */

    setlinestyle(1, 0, 1);             /* draw lines */
    line(cx[1], cy[1], cx[18], cy[18]);
    line(cx[2], cy[2], cx[17], cy[17]);
    line(cx[3], cy[3], cx[16], cy[16]);
    line(cx[3], cy[3], cx[6], cy[6]);
    line(cx[6], cy[6], cx[9], cy[9]);
    line(cx[3], cy[3], cx[9], cy[9]);
    line(cx[16], cy[16], cx[10], cy[10]);
    line(cx[10], cy[10], cx[13], cy[13]);
    line(cx[13], cy[13], cx[16], cy[16]);


    setlinestyle(0, 0, 3);             /* set line */
    setcolor( 4 );                     /* set color */
    line(220, 14, 220, 480);
    line(220, 32, 530, 32);

    line(0, 350, 220, 350);
    line(0, 365, 220, 365);

    setcolor( 62 );                /* design text, color */
    outtextxy( 20, 354, "TRI-SHP. DIMENSIONS");

    outtextxy( 250, 21, "TRI-SHP. PROCESSING PARAMETERS");
    setcolor( 63 );
    outtextxy(12, 340, "STRAIGHT LINE DEVELOPMENT");


    setlinestyle(0, 0, 0);         /* set line */


    setcolor( 62 );
    outtextxy(200, 5, "PROCESS FORM 1");


    optionsgr();               /* call graphics */
    options();                 /* call options */

   }                           /* end option 9 if */


  }                            /* end straighttr. function */
```

```
/*-----------------------------------------------------*/

  void dxfmaker(void)       /* DXF generator function */

{

   int d;
   cleardevice();              /* set up screen */
   backgrcol = 8;
   setbkcolor(backgrcol);      /* set color */

   setcolor( 62 );
   outtextxy( 155, 80, "INITIALIZING SLD1.DXF TO A:
             ....STANDBY:");
   setcolor( 60 );
   setlinestyle(0, 0, 3);      /* set line */
   line( 0, 32, 640, 32 );
   settextstyle(1, 0, 1);
   setcolor( 62 );
   outtextxy( 100, 9, "DATA EXCHANGE FILE -(DXF) FORMAT
             INTERFACE");
   settextstyle(0, 0, 0);


   FILE *fp;                            /* filer */

   fp = fopen("a:SLD1.DXF", "w");       /* opens DXF file
                                           on a: */



   fputs("  0\n", fp);                 /* DXF code */
   fputs("SECTION\n", fp);
   fputs("  2\n", fp);
   fputs("HEADER\n", fp);
   fputs("  9\n", fp);
   fputs("$ACADVER\n", fp);
   fputs("  1\n", fp);
   fputs("AC1009\n", fp);
   fputs("  9\n", fp);
   fputs("$INSBASE\n", fp);             /* DXF code */
   fputs(" 10\n", fp);
   fputs("0.0\n", fp);
   fputs(" 20\n", fp);
   fputs("0.0\n", fp);
   fputs(" 30\n", fp);
   fputs("0.0\n", fp);
   fputs("  9\n", fp);
   fputs("$EXTMIN\n", fp);              /* DXF code */
   fputs(" 10\n", fp);                  /* set extents */
   fputs("2.0\n", fp);
fputs(" 20\n", fp);
fputs("3.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$EXTMAX\n", fp);          /* DXF code */
fputs(" 10\n", fp);              /* set extents */
fputs("10.0\n", fp);
fputs(" 20\n", fp);
fputs("9.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$LIMMIN\n", fp);          /* DXF code */
fputs(" 10\n", fp);              /* set limits */
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$LIMMAX\n", fp);          /* DXF code */
fputs(" 10\n", fp);              /* set limits */
fputs("12.0\n", fp);
fputs(" 20\n", fp);
fputs("9.0\n", fp);
fputs("  9\n", fp);
fputs("$ORTHOMODE\n", fp);       /* DXF code */
fputs(" 70\n", fp);              /* set ortho */
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$REGENMODE\n", fp);       /* set regen */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$FILLMODE\n", fp);        /* fill mode */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$QTEXTMODE\n", fp);       /* text style */
fputs(" 70\n", fp);
fputs("    0\n", fp);
fputs("  9\n", fp);
fputs("$MIRRTEXT\n", fp);        /* mirror */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$DRAGMODE\n", fp);        /* line drag */
fputs(" 70\n", fp);
fputs("     2\n", fp);
fputs("  9\n", fp);
fputs("$LTSCALE\n", fp);         /* scaling */
fputs(" 40\n", fp);
fputs("1.0\n", fp);
fputs("  9\n", fp);
fputs("$OSMODE\n", fp);          /* DXF code */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$ATTMODE\n", fp);          /* DXF code */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$TEXTSIZE\n", fp);         /* text set */
fputs(" 40\n", fp);
fputs("0.2\n", fp);
fputs("  9\n", fp);
fputs("$TRACEWID\n", fp);         /* tracing */
fputs(" 40\n", fp);
fputs("0.05\n", fp);
fputs("  9\n", fp);
fputs("$TEXTSTYLE\n", fp);        /* style */
fputs("  7\n", fp);
fputs("STANDARD\n", fp);
fputs("  9\n", fp);
fputs("$CLAYER\n", fp);           /* layer */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  9\n", fp);
fputs("$CELTYPE\n", fp);
fputs("  6\n", fp);
fputs("BYLAYER\n", fp);
fputs("  9\n", fp);
fputs("$CECOLOR\n", fp);          /* color */
fputs(" 62\n", fp);
fputs("   256\n", fp);
fputs("  9\n", fp);
fputs("$DIMSCALE\n", fp);         /* dim scale */
fputs(" 40\n", fp);
fputs("1.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMASZ\n", fp);
fputs(" 40\n", fp);
fputs("0.18\n", fp);
fputs("  9\n", fp);
fputs("$DIMEXO\n", fp);           /* dimension */
fputs(" 40\n", fp);
fputs("0.0625\n", fp);
fputs("  9\n", fp);
fputs("$DIMDLI\n", fp);           /* dimension */
fputs(" 40\n", fp);
fputs("0.38\n", fp);
fputs("  9\n", fp);
fputs("$DIMRND\n", fp);           /* dimension */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMDLE\n", fp);           /* dimension */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMEXE\n", fp);              /* dimension */
fputs(" 40\n", fp);
fputs("0.18\n", fp);
fputs("  9\n", fp);
fputs("$DIMTP\n", fp);               /* dimension */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTM\n", fp);               /* dimension */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTXT\n", fp);              /* dimension */
fputs(" 40\n", fp);
fputs("0.18\n", fp);
fputs("  9\n", fp);
fputs("$DIMCEN\n", fp);              /* dimension */
fputs(" 40\n", fp);
fputs("0.09\n", fp);
fputs("  9\n", fp);
fputs("$DIMTSZ\n", fp);              /* dimension */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTOL\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMLIM\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTIH\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$DIMTOH\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$DIMSE1\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMSE2\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTAD\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMZIN\n", fp);              /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMBLK\n", fp);
fputs("  1\n", fp);
fputs("    \n", fp);
fputs("  9\n", fp);
fputs("$DIMASO\n", fp);
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$DIMSHO\n", fp);                    /* dimension */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$DIMPOST\n", fp);
fputs("  1\n", fp);
fputs("    \n", fp);
fputs("  9\n", fp);
fputs("$DIMAPOST\n", fp);
fputs("  1\n", fp);
fputs("    \n", fp);
fputs("  9\n", fp);
fputs("$DIMALT\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMALTD\n", fp);                   /* dimension */
fputs(" 70\n", fp);
fputs("     2\n", fp);
fputs("  9\n", fp);
fputs("$DIMALTF\n", fp);
fputs(" 40\n", fp);
fputs("25.4\n", fp);
fputs("  9\n", fp);
fputs("$DIMLFAC\n", fp);
fputs(" 40\n", fp);
fputs("1.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTOFL\n", fp);                    /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTVP\n", fp);
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTIX\n", fp);                     /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMSOXD\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMSAH\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMBLK1\n", fp);                /* dim block */
fputs("  1\n", fp);
fputs("    \n", fp);
fputs("  9\n", fp);
fputs("$DIMBLK2\n", fp);                /* dim block */
fputs("  1\n", fp);
fputs("    \n", fp);
fputs("  9\n", fp);
fputs("$DIMSTYLE\n", fp);               /* set style */
fputs("  2\n", fp);
fputs("*UNNAMED\n", fp);                /* open */
fputs("  9\n", fp);
fputs("$DIMCLRD\n", fp);                /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMCLRE\n", fp);                /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMCLRT\n", fp);                /* dimension */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$DIMTFAC\n", fp);                /* dimension */
fputs(" 40\n", fp);
fputs("1.0\n", fp);
fputs("  9\n", fp);
fputs("$DIMGAP\n", fp);                 /* dimension */
fputs(" 40\n", fp);
fputs("0.09\n", fp);
fputs("  9\n", fp);
fputs("$LUNITS\n", fp);                 /* units */
fputs(" 70\n", fp);
fputs("     2\n", fp);
fputs("  9\n", fp);
fputs("$LUPREC\n", fp);
fputs(" 70\n", fp);
fputs("     4\n", fp);
fputs("  9\n", fp);
fputs("$SKETCHINC\n", fp);
fputs(" 40\n", fp);
fputs("0.1\n", fp);
fputs("  9\n", fp);
fputs("$FILLETRAD\n", fp);              /* fillet */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$AUNITS\n", fp);                 /* units */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$AUPREC\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$MENU\n", fp);                          /* main menu */
fputs("  1\n", fp);
fputs("acad\n", fp);
fputs("  9\n", fp);
fputs("$ELEVATION\n", fp);                     /* elevation */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$PELEVATION\n", fp);                    /* elevation */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$THICKNESS\n", fp);                     /* thickness */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$LIMCHECK\n", fp);                      /* limits */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$BLIPMODE\n", fp);
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$CHAMFERA\n", fp);                      /* chamfer */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$CHAMFERB\n", fp);                      /* chamfer */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$SKPOLY\n", fp);                        /* poly line */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$TDCREATE\n", fp);                     /* create */
fputs(" 40\n", fp);
fputs("2449494.982737152\n", fp);
fputs("  9\n", fp);
fputs("$TDUPDATE\n", fp);                     /* update */
fputs(" 40\n", fp);
fputs("2449494.982737152\n", fp);
fputs("  9\n", fp);
fputs("$TDINDWG\n", fp);
fputs(" 40\n", fp);
fputs("0.0000000000\n", fp);
fputs("  9\n", fp);
fputs("$TDUSRTIMER\n", fp);
fputs(" 40\n", fp);
fputs("0.0000000000\n", fp);
fputs("  9\n", fp);
fputs("$USRTIMER\n", fp);                /* timer */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$ANGBASE\n", fp);
fputs(" 50\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$ANGDIR\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$PDMODE\n", fp);                  /* mode */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$PDSIZE\n", fp);
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$PLINEWID\n", fp);            /* poly line width */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$COORDS\n", fp);              /* coordinates */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$SPLFRAME\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$SPLINETYPE\n", fp);              /* line */
fputs(" 70\n", fp);
fputs("     6\n", fp);
fputs("  9\n", fp);
fputs("$SPLINESEGS\n", fp);              /* segments */
fputs(" 70\n", fp);
fputs("     8\n", fp);
fputs("  9\n", fp);
fputs("$ATTDIA\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$ATTREQ\n", fp);
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$HANDLING\n", fp);                /* handling */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$HANDSEED\n", fp);
fputs("  5\n", fp);
fputs("0\n", fp);
fputs("  9\n", fp);
fputs("$SURFTAB1\n", fp);                    /* table */
fputs(" 70\n", fp);
fputs("     6\n", fp);
fputs("  9\n", fp);
fputs("$SURFTAB2\n", fp);                    /* table */
fputs(" 70\n", fp);
fputs("     6\n", fp);
fputs("  9\n", fp);
fputs("$SURFTYPE\n", fp);                    /* surface */
fputs(" 70\n", fp);
fputs("     6\n", fp);
fputs("  9\n", fp);
fputs("$SURFU\n", fp);                       /* surface */
fputs(" 70\n", fp);
fputs("     6\n", fp);
fputs("  9\n", fp);
fputs("$SURFV\n", fp);                       /* surface */
fputs(" 70\n", fp);
fputs("     6\n", fp);
fputs("  9\n", fp);
fputs("$UCSNAME\n", fp);                     /* UCS */
fputs("  2\n", fp);
fputs("    \n", fp);
fputs("  9\n", fp);
fputs("$UCSORG\n", fp);                      /* UCS */
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$UCSXDIR\n", fp);                     /* UCS */
fputs(" 10\n", fp);
fputs("1.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$UCSYDIR\n", fp);                     /* UCS */
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("1.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$PUCSNAME\n", fp);                /* UCS */
fputs("  2\n", fp);
fputs("    \n", fp);
fputs("  9\n", fp);
fputs("$PUCSORG\n", fp);                 /* UCS */
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$PUCSXDIR\n", fp);                /* UCS */
fputs(" 10\n", fp);
fputs("1.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$PUCSYDIR\n", fp);                /* UCS */
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("1.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$USERI1\n", fp);                  /* user */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$USERI2\n", fp);                  /* user */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$USERI3\n", fp);                  /* user */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$USERI4\n", fp);                  /* user */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$USERI5\n", fp);                  /* user */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$USERR1\n", fp);                  /* user */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$USERR2\n", fp);                  /* user */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$USERR3\n", fp);                 /* user */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$USERR4\n", fp);                 /* user */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$USERR5\n", fp);                 /* user */
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$WORLDVIEW\n", fp);              /* view */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$SHADEDGE\n", fp);               /* shading */
fputs(" 70\n", fp);
fputs("     3\n", fp);
fputs("  9\n", fp);
fputs("$SHADEDIF\n", fp);               /* shading */
fputs(" 70\n", fp);
fputs("    70\n", fp);
fputs("  9\n", fp);
fputs("$TILEMODE\n", fp);               /* tile mode */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$MAXACTVP\n", fp);
fputs(" 70\n", fp);
fputs("    16\n", fp);
fputs("  9\n", fp);
fputs("$PINSBASE\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$PLIMCHECK\n", fp);               /* limits */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$PEXTMIN\n", fp);                 /* poly ext */
fputs(" 10\n", fp);
fputs("1.000000E+20\n", fp);
fputs(" 20\n", fp);
fputs("1.000000E+20\n", fp);
fputs(" 30\n", fp);
fputs("1.000000E+20\n", fp);
fputs("  9\n", fp);
fputs("$PEXTMAX\n", fp);                 /* poly ext */
fputs(" 10\n", fp);
fputs("-1.000000E+20\n", fp);
fputs(" 20\n", fp);
fputs("-1.000000E+20\n", fp);
fputs(" 30\n", fp);
fputs("-1.000000E+20\n", fp);
fputs("  9\n", fp);
fputs("$PLIMMIN\n", fp);                 /* line min */
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs("  9\n", fp);
fputs("$PLIMMAX\n", fp);                 /* line max */
fputs(" 10\n", fp);
fputs("12.0\n", fp);
fputs(" 20\n", fp);
fputs("9.0\n", fp);
fputs("  9\n", fp);
fputs("$UNITMODE\n", fp);                /* units */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$VISRETAIN\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$PLINEGEN\n", fp);             /* ploy line gen */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  9\n", fp);
fputs("$PSLTSCALE\n", fp);               /* scaling */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  9\n", fp);
fputs("$TREEDEPTH\n", fp);               /* depth */
fputs(" 70\n", fp);
fputs("  3020\n", fp);
fputs("  9\n", fp);
fputs("$DWGCODEPAGE\n", fp);             /* DWG coding */
fputs("  3\n", fp);
fputs("ascii\n", fp);
fputs("  0\n", fp);
fputs("ENDSEC\n", fp);                  /* end section */
fputs("  0\n", fp);
fputs("SECTION\n", fp);                 /* section */
fputs("  2\n", fp);
fputs("TABLES\n", fp);                  /* table */
fputs("  0\n", fp);
fputs("TABLE\n", fp);                   /* table */
fputs("  2\n", fp);
fputs("VPORT\n", fp);                   /* view port */
fputs(" 70\n", fp);
fputs("     2\n", fp);
fputs("  0\n", fp);
fputs("VPORT\n", fp);                   /* view port */
fputs("  2\n", fp);
fputs("*ACTIVE\n", fp);
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 11\n", fp);
fputs("1.0\n", fp);
fputs(" 21\n", fp);
fputs("1.0\n", fp);
fputs(" 12\n", fp);
fputs("6.243341\n", fp);                /* view port */
fputs(" 22\n", fp);
fputs("4.5\n", fp);
fputs(" 13\n", fp);
fputs("0.0\n", fp);
fputs(" 23\n", fp);
fputs("0.0\n", fp);
fputs(" 14\n", fp);                      /* view port */
fputs("1.0\n", fp);
fputs(" 24\n", fp);
fputs("1.0\n", fp);
fputs(" 15\n", fp);
fputs("0.0\n", fp);
fputs(" 25\n", fp);
fputs("0.0\n", fp);                       /* view port */
fputs(" 16\n", fp);
fputs("0.0\n", fp);
fputs(" 26\n", fp);
fputs("0.0\n", fp);
fputs(" 36\n", fp);
fputs("1.0\n", fp);                       /* view port */
fputs(" 17\n", fp);
fputs("0.0\n", fp);
fputs(" 27\n", fp);
fputs("0.0\n", fp);
fputs(" 37\n", fp);
fputs("0.0\n", fp);
fputs(" 40\n", fp);
fputs("9.0\n", fp);
fputs(" 41\n", fp);
fputs("1.387409\n", fp);                  /* view port */
fputs(" 42\n", fp);
fputs("50.0\n", fp);
fputs(" 43\n", fp);
fputs("0.0\n", fp);
fputs(" 44\n", fp);
fputs("0.0\n", fp);
fputs(" 50\n", fp);
fputs("0.0\n", fp);
fputs(" 51\n", fp);
fputs("0.0\n", fp);
fputs(" 71\n", fp);
fputs("     0\n", fp);                  /* view port */
fputs(" 72\n", fp);
fputs("   100\n", fp);
fputs(" 73\n", fp);
fputs("     1\n", fp);
fputs(" 74\n", fp);
fputs("     1\n", fp);                  /* view port */
fputs(" 75\n", fp);
fputs("     0\n", fp);
fputs(" 76\n", fp);
fputs("     0\n", fp);
fputs(" 77\n", fp);
fputs("     0\n", fp);                  /* view port */
fputs(" 78\n", fp);
fputs("     0\n", fp);
fputs("  0\n", fp);
fputs("ENDTAB\n", fp);
fputs("  0\n", fp);
fputs("TABLE\n", fp);                   /* table */
fputs("  2\n", fp);
fputs("LTYPE\n", fp);                   /* line */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  0\n", fp);
fputs("LTYPE\n", fp);                   /* line */
fputs("  2\n", fp);
fputs("CONTINUOUS\n", fp);              /* C line */
fputs(" 70\n", fp);
fputs("    64\n", fp);
fputs("  3\n", fp);
fputs("Solid line\n", fp);              /* S line */
fputs(" 72\n", fp);
fputs("    65\n", fp);
fputs(" 73\n", fp);
fputs("     0\n", fp);
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("ENDTAB\n", fp);
fputs("  0\n", fp);
fputs("TABLE\n", fp);                   /* table */
fputs("  2\n", fp);
fputs("LAYER\n", fp);                   /* set layer */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  0\n", fp);
fputs("LAYER\n", fp);                   /* layer */
fputs("   2\n", fp);
fputs("0\n", fp);
fputs("  70\n", fp);
fputs("     0\n", fp);
fputs("  62\n", fp);
fputs("     7\n", fp);
fputs("   6\n", fp);
fputs("CONTINUOUS\n", fp);                    /* cont. */
fputs("  0\n", fp);
fputs("ENDTAB\n", fp);
fputs("  0\n", fp);
fputs("TABLE\n", fp);                         /* table */
fputs("  2\n", fp);
fputs("STYLE\n", fp);                         /* set style */
fputs(" 70\n", fp);
fputs("     1\n", fp);
fputs("  0\n", fp);
fputs("STYLE\n", fp);                         /* style */
fputs("  2\n", fp);
fputs("STANDARD\n", fp);                      /* STD. mode */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs(" 40\n", fp);
fputs("0.0\n", fp);
fputs(" 41\n", fp);
fputs("1.0\n", fp);
fputs(" 50\n", fp);
fputs("0.0\n", fp);
fputs(" 71\n", fp);
fputs("     0\n", fp);
fputs(" 42\n", fp);
fputs("0.2\n", fp);
fputs("  3\n", fp);
fputs("txt\n", fp);
fputs("  4\n", fp);
fputs("     \n", fp);
fputs("  0\n", fp);
fputs("ENDTAB\n", fp);
fputs("  0\n", fp);
fputs("TABLE\n", fp);                           /* table */
fputs("  2\n", fp);
fputs("VIEW\n", fp);                            /* V. port */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  0\n", fp);
fputs("ENDTAB\n", fp);
fputs("  0\n", fp);
fputs("TABLE\n", fp);                           /* table */
fputs("  2\n", fp);
fputs("UCS\n", fp);                           /* set UCS */
fputs(" 70\n", fp);
fputs("     0\n", fp);
fputs("  0\n", fp);
fputs("ENDTAB\n", fp);
fputs("  0\n", fp);
fputs("TABLE\n", fp);                         /* table */
```

```
    fputs("  2\n", fp);
    fputs("APPID\n", fp);
    fputs(" 70\n", fp);
    fputs("     1\n", fp);
    fputs("  0\n", fp);
    fputs("APPID\n", fp);
    fputs("  2\n", fp);
    fputs("ACAD\n", fp);
    fputs(" 70\n", fp);
    fputs("    64\n", fp);
    fputs("  0\n", fp);
    fputs("ENDTAB\n", fp);
    fputs("  0\n", fp);
    fputs("TABLE\n", fp);                          /* table */
    fputs("  2\n", fp);
    fputs("DIMSTYLE\n", fp);           /* dimension st. */
    fputs(" 70\n", fp);
    fputs("     0\n", fp);
    fputs("  0\n", fp);
    fputs("ENDTAB\n", fp);
    fputs("  0\n", fp);
    fputs("ENDSEC\n", fp);
    fputs("  0\n", fp);
    fputs("SECTION\n", fp);            /* section */
    fputs("  2\n", fp);
    fputs("BLOCKS\n", fp);             /* 0 block */
    fputs("  0\n", fp);
    fputs("ENDSEC\n", fp);
    fputs("  0\n", fp);
    fputs("SECTION\n", fp);            /* section */
    fputs("  2\n", fp);
    fputs("ENTITIES\n", fp);           /* dr. Entities */
    fputs("  0\n", fp);


    fclose(fp);                       /* close filer */

 }                            /* end dxfmaker function */


/*------------------------------------------------------------*/

    void dxfparameter1(void)    /* DXF #1 */

 {
    outtextxy(155, 100, "DXF VERTEX DATA BEING
              GENERATED.........");



    FILE *fp;                        /* filer */

    fp = fopen("a:SLD1.DXF", "a");   /* opens DXF file
                                        on a: */
```

```
fputs("POLYLINE\n", fp);  /* outside geometry begin */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[1]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[1]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertext 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end sequence */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);            /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);           /* coordinate */
```

```
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[4]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[4]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);            /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[4]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[4]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[5]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[5]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);           /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[5]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[5]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[7]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[7]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);             /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);           /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[7]);         /* coordinate */
```

```
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[7]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[8]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[8]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);            /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[8]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[8]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);            /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[11]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[11]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);            /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[11]);         /* coordinate */
```

```
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[11]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[14]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[14]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);             /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("      1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[14]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[14]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[15]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[15]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);              /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[15]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[15]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[17]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[17]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                  /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);                /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[17]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[17]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[18]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[18]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                    /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);                  /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[18]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[18]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[20]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[20]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                    /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);                  /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[20]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[20]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[21]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[21]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);              /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[21]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[21]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[22]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[22]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                    /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);                  /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[22]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[22]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[24]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[24]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                    /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);                  /* poly line */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[24]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[24]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[1]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[1]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);     /* outside geometry end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);     /* solid inside lines */
fputs("  8\n", fp);                       /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                   /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[15]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[15]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                          /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[26]);                     /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[26]);                     /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                          /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);        /* solid inside lines */
fputs("  8\n", fp);                              /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                         /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[20]);                    /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[20]);                    /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                         /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[25]);                    /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[25]);                    /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                         /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);        /* solid inside lines */
fputs("  8\n", fp);                              /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[28]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[28]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                  /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);      /* solid inside lines */
fputs("  8\n", fp);                        /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[27]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[27]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[5]);                /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[5]);                /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                    /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);        /* fold inside lines */
fputs("  8\n", fp);                        /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[23]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[23]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[2]);                /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[2]);                /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                    /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);        /* fold inside lines */
fputs("  8\n", fp);                       /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[22]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[22]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);         /* fold inside lines */
fputs("  8\n", fp);                   /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[12]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[12]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);           /* fold inside lines */
fputs("  8\n", fp);                        */ poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[22]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[22]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[13]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[13]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);           /* fold inside lines */
fputs("  8\n", fp);                          /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[26]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[26]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[28]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[28]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);            /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);       /* fold inside lines */
fputs("  8\n", fp);                /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);           /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[25]);      /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[25]);      /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[27]);        /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[27]);        /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);             /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);         /* fold inside lines */
fputs("  8\n", fp);                    /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[22]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[22]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);         /* fold inside lines */
fputs("  8\n", fp);                     /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("      1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[16]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[16]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[19]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[19]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);            /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);       /* fold inside lines */
fputs("  8\n", fp);                    /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("      1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[9]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[9]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[6]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[6]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);          /* test outside line */
fputs("  8\n", fp);                         /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[14]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[14]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[11]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[11]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("ENDSEC\n", fp);               /* end section */
fputs("  0\n", fp);
fputs("EOF\n", fp);                  /* end of file */
```

```
      fclose(fp);                          /* close filer */


      outtextxy(55, 200, "*** FILE TRANSFER AND DATA
                SEQUENCE COMPLETE...PRESS A KEY... ***");

      BEEP;                      /* computer beep */
      getch();
      cleardevice();             /* clear screen */

      menu1();                   /* call menu 1 */


    }
                           /* end dxfparameter1 function */


  /*--------------------------------------------------------*/

     void dxfparameter2(void)

    {                      /* begin dxfparameter2 function */


      outtextxy(155, 100, "DXF VERTEX DATA BEING
                GENERATED..........");


      FILE *fp;               /* open filer */

      fp = fopen("a:SLD1.DXF", "a");   /* adds to DXF file
                                          on a: */


      fputs("POLYLINE\n", fp);  /* outside geometry begin */
      fputs("  8\n", fp);             /* poly line */
      fputs("0\n", fp);
      fputs(" 66\n", fp);
      fputs("     1\n", fp);
      fputs(" 10\n", fp);
      fputs("0.0\n", fp);
      fputs(" 20\n", fp);
      fputs("0.0\n", fp);
      fputs(" 30\n", fp);
      fputs("0.0\n", fp);
      fputs("  0\n", fp);
      fputs("VERTEX\n", fp);               /* vertex 1 */
      fputs("  8\n", fp);
      fputs("0\n", fp);
      fputs(" 10\n", fp);
      fprintf(fp, "%f\n", x[0]);           /* coordinate */
      fputs(" 20\n", fp);
      fprintf(fp, "%f\n", y[0]);           /* coordinate */
```

```
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[6]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[6]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                         /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[13]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[13]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[7]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[7]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                          /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                        /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[13]);                   /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[13]);                   /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                        /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[14]);                   /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[14]);                   /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                        /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                         /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                      /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[7]);                  /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[7]);                  /* coordinate */
```

```
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[6]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[6]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);             /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                          /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[12]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[12]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[1]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[1]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                          /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                   /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[11]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[11]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                   /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[2]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[2]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                   /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                          /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                   /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);          /* coordinate */
```

```
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                 /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                 /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                    /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                 /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[9]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[9]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                 /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[4]);             /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[4]);             /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                 /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                    /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[8]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[8]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[5]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[5]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("CIRCLE\n", fp);            /* circle geometry */
fputs("  8\n", fp);               /* circle */
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[17]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[17]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs(" 40\n", fp);
fprintf(fp, "%f\n", X1);
fputs("  0\n", fp);
fputs("CIRCLE\n", fp);            /* circle geometry */
fputs("  8\n", fp);               /* circle */
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[18]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[18]);       /* coordinate */
```

```
    fputs(" 30\n", fp);
    fputs("0.0\n", fp);
    fputs(" 40\n", fp);
    fprintf(fp, "%f\n", X1);
    fputs("  0\n", fp);
    fputs("ENDSEC\n", fp);          /* end section */
    fputs("  0\n", fp);
    fputs("EOF\n", fp);             /* end of file */


    fclose(fp);                 /* close filer */


    outtextxy(55, 200, "*** FILE TRANSFER AND DATA
              SEQUENCE COMPLETE...PRESS A KEY... ***");

    BEEP;               /* computer beep */
    getch();
    cleardevice();      /* clear screen */

    menu1();            /* call menu 1 */



 }                           /* end dxfparameter2 function */

/*-------------------------------------------------------*/

   void dxfparameter3(void)

 {                         /* begin dxfparameter3 function */


    outtextxy(155, 100, "DXF VERTEX DATA BEING
              GENERATED.........");



    FILE *fp;                /* open filer */

    fp = fopen("a:SLD1.DXF", "a");    /* adds to DXF file
                                         on a: */


    fputs("POLYLINE\n", fp);  /* outside geometry begin */
    fputs("  8\n", fp);              /* poly line */
    fputs("0\n", fp);
    fputs(" 66\n", fp);
    fputs("     1\n", fp);
    fputs(" 10\n", fp);
    fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[0]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[0]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                   /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1*/
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[19]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[19]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[16]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[16]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                  /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[19]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[19]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[20]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[20]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);               /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[16]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[16]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[15]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[15]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);            /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);               /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[15]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[15]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);            /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[14]);       /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[14]);       /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);            /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                       /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[14]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[14]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[13]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[13]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                  /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[13]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[13]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[12]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[12]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* veretex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[12]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[12]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[11]);        /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[11]);        /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);             /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[11]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[11]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                  /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[9]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[9]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);              /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[9]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[9]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[8]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[8]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[8]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[8]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);               /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[7]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[7]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);              /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[7]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[7]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[6]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[6]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                  /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[6]);                /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[6]);                /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[5]);              /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[5]);              /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                  /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                 /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[5]);              /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[5]);              /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                  /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[4]);              /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[4]);              /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                  /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* solid outside geometry */
fputs("  8\n", fp);                    /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[4]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[4]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                 /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                 /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[18]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[18]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                 /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[1]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[1]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                         /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[17]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[17]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[2]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[2]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                    /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[16]);               /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[16]);               /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                    /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);                /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);                /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                    /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                      /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                     /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[16]);                /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[16]);                /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                     /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[13]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[13]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);               /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                 /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[13]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[13]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);              /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[16]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[16]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[10]);           /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[10]);           /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);                /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                   /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);            /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);            /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);                /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[9]);          /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[9]);          /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);              /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                 /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
fputs(" 20\n", fp);
fputs("0.0\n", fp);
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 1 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[3]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[3]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("VERTEX\n", fp);             /* vertex 2 */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs(" 10\n", fp);
fprintf(fp, "%f\n", x[6]);         /* coordinate */
fputs(" 20\n", fp);
fprintf(fp, "%f\n", y[6]);         /* coordinate */
fputs(" 30\n", fp);
fputs("0.0\n", fp);
fputs("  0\n", fp);
fputs("SEQEND\n", fp);             /* end */
fputs("  8\n", fp);
fputs("0\n", fp);
fputs("  0\n", fp);
fputs("POLYLINE\n", fp);  /* hidden inside geometry */
fputs("  8\n", fp);                /* poly line */
fputs("0\n", fp);
fputs(" 66\n", fp);
fputs("     1\n", fp);
fputs(" 10\n", fp);
fputs("0.0\n", fp);
```

```
    fputs(" 20\n", fp);
    fputs("0.0\n", fp);
    fputs(" 30\n", fp);
    fputs("0.0\n", fp);
    fputs("  0\n", fp);
    fputs("VERTEX\n", fp);            /* vertex 1 */
    fputs("  8\n", fp);
    fputs("0\n", fp);
    fputs(" 10\n", fp);
    fprintf(fp, "%f\n", x[6]);        /* coordinate */
    fputs(" 20\n", fp);
    fprintf(fp, "%f\n", y[6]);        /* coordinate */
    fputs(" 30\n", fp);
    fputs("0.0\n", fp);
    fputs("  0\n", fp);
    fputs("VERTEX\n", fp);            /* vertex 2 */
    fputs("  8\n", fp);
    fputs("0\n", fp);
    fputs(" 10\n", fp);
    fprintf(fp, "%f\n", x[9]);        /* coordinate */
    fputs(" 20\n", fp);
    fprintf(fp, "%f\n", y[9]);        /* coordinate */
    fputs(" 30\n", fp);
    fputs("0.0\n", fp);
    fputs("  0\n", fp);
    fputs("SEQEND\n", fp);            /* end */
    fputs("  8\n", fp);
    fputs("0\n", fp);
    fputs("  0\n", fp);
    fputs("ENDSEC\n", fp);            /* end section */
    fputs("  0\n", fp);
    fputs("EOF\n", fp);               /* end of file */


    fclose(fp);                     /* close filer */


    outtextxy(55, 200, "*** FILE TRANSFER AND DATA
              SEQUENCE COMPLETE...PRESS A KEY... ***");

    BEEP;                 /* computer beep */
    getch();
    cleardevice();        /* clear screen */

    menu1();              /* call menu 1 function */



  }                       /* end dxfparameter3 function */


/*-------------------------------------------------------*/
```

```
void PromptLine( char *msg )          /* print funct. */
{
int height, MaxX = getmaxx(), MaxY = getmaxy();

setcolor( getmaxcolor() );
settextstyle( DEFAULT_FONT, HORIZ_DIR, 1 );
settextjustify( CENTER_TEXT, TOP_TEXT );
height = textheight( "H" );
bar( 0, MaxY-( height+4 ), MaxX, MaxY );
rectangle( 0, MaxY-( height+4 ), MaxX, MaxY );
outtextxy( MaxX/2, MaxY-(height+2), msg );
}

#define PORTRAIT 0                  /* print types */
#define LANDSCAPE 1
#define GREYSCALE 2
int Negative;

int format( double position )
{
  int width = 6;
  if( position < 1000.0 ) width--;
  if( position < 100.0 ) width--;
  if( position < 10.0 ) width--;
  return( width );
 }
 int Grey_Scale( int scanline, int palette_entry )
 {
   int grey = 0;
   if( GraphDriver == CGA && GraphMode != CGAHI )
   {
    switch( scanline )
    {
        case 0: { if( palette_entry & 1 )
              grey |= 9;
            if( palette_entry & 2 )
              grey |= 6;
             } break;
        case 1: { if( palette_entry & 1 )
              grey |= 4;
            if( palette_entry & 2 )
              grey |= 11;
             } break;
        case 2: { if( palette_entry & 1 )
              grey |= 2;
            if( palette_entry & 2 )
              grey |= 13;
             } break;
        case 3: { if( palette_entry & 1 )
              grey |= 9;
            if( palette_entry & 2 )
              grey |= 6;
             } break;
    }  }
    else
    {  switch( scanline )
       {
       case 0: { if( palette_entry & 4 )
             grey |= 5;
              if( palette_entry & 8 )
             grey |= 10;
            } break;
       case 1: { if( palette_entry & 1 )
             grey |= 2;
              if( palette_entry & 2 )
             grey |= 8;
              if( palette_entry & 8 )
             grey |= 5;
            } break;
       case 2: { if( palette_entry & 4 )
             grey |= 5;
              if( palette_entry & 8 )
             grey |= 10;
            } break;
       case 3: { if( palette_entry & 2 )
             grey |= 2;
              if( palette_entry & 8 )
             grey |= 5;
            } break;
   }  }
   if( Negative ) grey ^= 0x0F;
   return( grey );
   }
   void LJ_Graphic( int Mode )
   {
   int i, j, k, p, q, xasp, yasp,
   MaxX = getmaxx() + 1,
   MaxY = getmaxy() + 1;
   static char graph_ends[] = "\x1B*rB";
   static char graph_init[] =
      "\x1B\x1B&l1H\x1B&l0\x1B*p0X\x1B*p0Y\x1B*t";
   double xprint, yprint, prstep, AspR;
   char m, resolution[3];

   getaspectratio( &xasp, &yasp );
   AspR = (double) xasp / (double) yasp;
   setviewport( 0, 0, MaxX, MaxY, 0 );
   switch( Mode )
   {
      case PORTRAIT:                /* portrait case */
      {
        xprint = 690.0;
        yprint = 500.0;
        strcpy( resolution, "100" );
        prstep = 7.2 / AspR;
        fprintf( stdprn, "%s%sR",
              graph_init, resolution );
      for( j=0; j<=MaxY; j++ )
      {
         fprintf( stdprn, "\x1B&a%-*.1fh%-*.1fV",
            format( xprint ), xprint,
            format( yprint ), yprint);
         yprint += prstep;
         fprintf( stdprn, "\x1B*r1A\x1B*b%dW",
               MaxX/8 );
         for( i=0; i<=MaxX/8; i++ )
         {
            m = 0;
            for( k=0; k<8; k++ )
            {
            m <<= 1;
            if( getpixel( i*8+k, j ) ) m++;
            }
            fprintf( stdprn, "%c", m );
          }
          fprintf( stdprn, "%s", graph_ends );
      }
    } break;
    case LANDSCAPE:              /* landscape case */
    {
       xprint = 1000.0;
       yprint = 1000.0;
       strcpy( resolution, "75" );
       prstep = 9.6 * AspR;
       fprintf( stdprn, "%s%sR",
             graph_init, resolution );
       for( j=0; j<MaxX; j++ )
       {
       fprintf( stdprn, "x1B&a%-*.1fh%-*.1fV",
             format( xprint ), xprint,
             format( yprint ), yprint );
       yprint += prstep;
       fprintf( stdprn, "\x1B*r1A\x1B*b%dW",
             (int) ( MaxY+4 ) / 8 );
        for( i=0; i<=MaxY/8; i++ )
       {
          m = 0;
          for( k=0; k<8; k++ )
          {
             m <<= 1;
             if( getpixel( MaxX-j, i*8+k ) ) m++;
          }
          fprintf( stdprn, "%c", m );
       }
       fprintf( stdprn, "%s", graph_ends );
        }
  } break;
  case GREYSCALE:              /* greyscale case */
  {
     xprint = 1000.0;
     yprint = 1000.0;
```

```
      strcpy( resolution, "300" );
      prstep = 2.4 * AspR;
      fprintf( stdprn, "%s%sR",
            graph_init, resolution );
      for( j=0; j<=MaxX; j++ )
      for( p+0; p<4; p++ )
      {
         fprintf( stdprn, "\x1B&a%-*.1fh%-*.1fV",
            format( xprint ), xprint,
            format( yprint ), yprint );
         yprint += prstep;
         fprintf( stdprn, "\x1B*r1A\x1B*b%dW",
            MaxY/2 );
         for( i=0; i<=MaxY/2; i++    )
         {
         m = 0;
         for( k=0; k<<=1; k++ )
         {
            m <<= 4;
            m |= Grey_Scale( p,
                 getpixel( MaxX-j, i*2+k ) );
         }
         fprintf( stdprn, "%c", m );
         }
         fprintf( stdprn, "%s", graph_ends );
   } } }
   fprintf( stdprn,
      "\x0C\x1B&1O\x1B(8U\x1B(sp10h12vsb3T\x1B&11H" );
  }
  void Print_Pause( int Invert )
  {
     char Ch;
     int Done = 0;
     if( Invert ) Negative = 1; else Negative = 0;
     PromptLine(
      "Enter <P>ortrait, <L>andscape, <G>reyscale"
      " - any other key to exit ..." );
     while( !Done )
     {
     while( kbhit() ) getch();
     Ch = getch();
     switch( toupper(Ch) )
     {
        case 'P': LJ_Graphic( PORTRAIT ); break;
        case 'L': LJ_Graphic( LANDSCAPE ); break;
        case 'G': LJ_Graphic( GREYSCALE ); break;
        default: Done++;

  } } }


/*----------------------------------------------------------*/

   void drawsqfull(void)   /* begin drawsqfull function */
```

```
{
  int backcolor, square[50], d;
  setlinestyle(0, 0, 0);
  cleardevice();
  backcolor = 8;
  setbkcolor( backcolor );             /* set colors */
  setcolor( 63 );

  for(d=0;d<=48;d=d+2) square[d]=cx[d/2];
  for(d=1;d<=49;d=d+2) square[d]=cy[(d-1)/2];

  drawpoly(25, square);                    /* draw lines */
  line(cx[20], cy[20], cx[25], cy[25]);
  line(cx[15], cy[15], cx[26], cy[26]);
  line(cx[27], cy[27], cx[5], cy[5]);
  line(cx[28], cy[28], cx[10], cy[10]);
  setlinestyle(1, 0, 1);                   /* set line */
  line(cx[19], cy[19], cx[16], cy[16]);
  line(cx[6], cy[6], cx[9], cy[9]);
  line(cx[23], cy[23], cx[2], cy[2]);
  line(cx[22], cy[22], cx[3], cy[3]);
  line(cx[25], cy[25], cx[27], cy[27]);
  line(cx[26], cy[26], cx[28], cy[28]);
  line(cx[22], cy[22], cx[13], cy[13]);
  line(cx[3], cy[3], cx[12], cy[12]);
  setlinestyle(0, 0, 0);                   /* set line */

  setcolor( 58 );                          /* draw text */
  outtextxy(cx[33]-13, cy[33], "SIDE");
  outtextxy(cx[34]-10, cy[34], "TOP");
  outtextxy(cx[31]-13, cy[31], "SIDE");
  outtextxy(cx[32]-13, cy[32], "SIDE");
  outtextxy(cx[29]-16, cy[29], "SIDE");
  outtextxy(cx[30]-23, cy[30]-10, "BOTTOM");
  outtextxy(cx[33]-5, cy[26]-10, "X1");
  outtextxy(cx[2]+10, 2*cy[2]-10, "X2");
  outtextxy(cx[4]+10, cy[32], "X3");
  outtextxy(cx[4]+10, cy[33], "X3");
  outtextxy(cx[7]+10, cy[30], "X2");

            /* screen text */

  setcolor( 62 );
  outtextxy(155, 5, "SQUARE STRAIGHT LINE DEVELOPMENT");

  setcolor( 63 );
  outtextxy(65, 465, "FULL SCREEN STRAIGHT - LINE
            DEVELOPMENT");

 optionsgr();                    /* call options graphics */
 options();                      /* call options */
```

```
  }                                /* end drawsqfull */

/*----------------------------------------------------------*/

    void drawtrifull(void) /* begin drawtrifull function */

 {


    int backcolor, trishape[50], d;
    setlinestyle(0, 0, 0);
    cleardevice();
    backcolor = 8;
    setbkcolor( backcolor );
    setcolor( 63 );                  /* set color */

    for(d=0;d<=40;d=d+2) trishape[d]=cx[d/2];
    for(d=1;d<=41;d=d+2) trishape[d]=cy[(d-1)/2];

    drawpoly(21, trishape);

    setlinestyle(1, 0, 1);           /* draw lines */
    line(cx[1], cy[1], cx[18], cy[18]);
    line(cx[2], cy[2], cx[17], cy[17]);
    line(cx[3], cy[3], cx[16], cy[16]);
    line(cx[3], cy[3], cx[6], cy[6]);
    line(cx[6], cy[6], cx[9], cy[9]);
    line(cx[3], cy[3], cx[9], cy[9]);
    line(cx[16], cy[16], cx[10], cy[10]);
    line(cx[10], cy[10], cx[13], cy[13]);
    line(cx[13], cy[13], cx[16], cy[16]);


    setcolor( 58 );                        /* draw text */
    outtextxy(cx[16]+20, cy[16]-25, "TOP");
    outtextxy(cx[3]+20, cy[3]+25, "BOTTOM");
    outtextxy(cx[1]+10, cy[1]-10, "SIDE");
    outtextxy(cx[2]+10, cy[2]-10, "SIDE");
    outtextxy(cx[3]+10, cy[3]-10, "SIDE");
    outtextxy(cx[10]-10, cy[10]+5, "X1");
    outtextxy(cx[17]+10, cy[17]+5, "X2");
    outtextxy(cx[18]+10, cy[18]+5, "X2");
    outtextxy(cx[17]+5, cy[17]+10, "X1");
    outtextxy(cx[9]-10, cy[9]+10, "X2");


   setcolor( 62 );            /* set color */
   outtextxy(155, 5, "TRIANGULAR STRAIGHT LINE
             DEVELOPMENT");

   setcolor( 63 );
   outtextxy(65, 465, "FULL SCREEN STRAIGHT - LINE
             DEVELOPMENT");
  optionsgr();                /* call options graphics */
  options();                  /* call options function */

 }                            /* end drawtrifull function */

/*-----------------------------------------------------------*/


  void exit(void)              /* exit program - graphics */

 {

  closegraph();
  printf("THANK YOU FOR USING THE AI-GCPP SOFTWARE..");

 }                             /* end exit function */


 /*----------------------------------------------------------*/

  void drawcylfull(void)

 {                           /* open drawcylfull function */


   int backcolor, cylinder[50], d;
   setlinestyle(0, 0, 0);
   cleardevice();
   backcolor = 8;
   setbkcolor( backcolor );
   setcolor( 63 );                  /* set color */

   for(d=0;d<=28;d=d+2) cylinder[d]=cx[d/2];
   for(d=1;d<=29;d=d+2) cylinder[d]=cy[(d-1)/2];

   drawpoly(15, cylinder);

   setlinestyle(1, 0, 1);                 /* draw lines */
   line(cx[1], cy[1], cx[12], cy[12]);
   line(cx[2], cy[2], cx[11], cy[11]);
   line(cx[3], cy[3], cx[10], cy[10]);
   line(cx[4], cy[4], cx[9], cy[9]);
   line(cx[5], cy[5], cx[8], cy[8]);

   ellipse(cx[15], cy[15]+50, 0, 360, (cx[17]-cx[15]),
          (cx[17]-cx[15]));
   ellipse(cx[16], cy[16]+50, 0, 360, (cx[18]-cx[16]),
          (cx[18]-cx[16]));


   setcolor( 58 );                        /* draw text */
```

```
    outtextxy(cx[17]+5, cy[17], "TOP");
    outtextxy(cx[18]+5, cy[18], "BOTTOM");
    outtextxy(cx[10]+20, cy[10]+10, "SIDE");
    outtextxy(cx[17]+5, cy[17]+15, "X1");
    outtextxy(cx[18]+5, cy[18]+15, "X2");
    outtextxy(cx[12]+5, cy[12]+10, "X3");
    outtextxy(cx[10]+5, cy[10]+10, "X1");


  setcolor( 62 );                  /* set color text */
  outtextxy(155, 5, "CYLINDRICAL STRAIGHT LINE
            DEVELOPMENT");

  setcolor( 63 );
  outtextxy(65, 465, "FULL SCREEN STRAIGHT - LINE
            DEVELOPMENT");

  optionsgr();              /* call options graphics */
  options();                /* call options function */

 }                          /* close drawcylfull function */


/*---------------------------------------------------------*/

    void helpmenu(void)      /* begin helpmenu function */

 {
    int backgrcol;
    cleardevice();
    backgrcol = 8;
    setbkcolor( backgrcol );          /* set color */

    setcolor( 62 );
    outtextxy( 230, 5, "-- HELP MENU --" );

    setcolor( 58 );
    outtextxy( 5, 70, "1-EXIT" );
    setcolor( 63 );
    outtextxy( 70, 70, "EXITS THE PROGRAM AND RETURNS TO
               DOS");
    setcolor( 58 );
    outtextxy( 5, 90, "2-CLEAR" );
    outtextxy( 5, 120, "3-HELP" );
    outtextxy( 5, 140, "4-PRINT" );
    outtextxy( 5, 160, "5-SLD");
    outtextxy( 5, 190, "6-OPT-A DXF OUT" );
    outtextxy( 5, 220, "7-OPT-B DXF OUT" );
    outtextxy( 5, 250, "8-OPT-C DXF OUT" );
    outtextxy( 5, 280, "9-FORM 1" );
    outtextxy( 5, 310, "10-FORM 2" );
    outtextxy( 5, 330, "11-FORM 3" );
    outtextxy( 5, 350, "12-FORM 4" );
    outtextxy( 5, 370, "13-PRINT P1" );
    outtextxy( 5, 390, "14-PRINT P2" );
    outtextxy( 5, 410, "15-PRINT P3" );
    outtextxy( 5, 430, "16-PRINT P4" );

    setcolor( 63 );
    outtextxy( 70, 90, "CLEARS ALL PARAMETERS AND REFRESHES
               THE MAIN MENU" );
    outtextxy( 70, 100, "FOR NEW DATA " );
    outtextxy( 70, 120, "PROVIDES GENERAL ON-LINE HELP" );
    outtextxy( 70, 140, "PRINTS THE SCREEN WITH A RASTER
               SCAN TO A LASERJET" );
    outtextxy( 70, 160, "PROVIDES A FULL SCREEN VIEW OF THE
               STRAIGHT LINE" );
    outtextxy( 70, 170, "DEVELOPMENT" );
    outtextxy( 140, 190, "CREATES A DXF FILE TO DISK OF
               OPTION A TO BE" );
    outtextxy( 140, 200, "READ BY STANDARD CAD SOFTWARE" );
    outtextxy( 140, 220, "CREATES A DXF FILE TO DISK OF
               OPTION B TO BE" );
    outtextxy( 140, 230, "READ BY STANDARD CAD SOFTWARE" );
    outtextxy( 140, 250, "CREATES A DXF FILE TO DISK OF
               OPTION C TO BE" );
    outtextxy( 140, 260, "READ BY STANDARD CAD SOFTWARE" );
    outtextxy( 85, 280, "PROVIDES STRAIGHT LINE
               DEVELOPMENTS WITH PROCESS" );
    outtextxy( 85, 290, "FORMS" );
    outtextxy( 85, 310, "PROVIDES PROCESS FORMS" );
    outtextxy( 85, 330, "PROVIDES PROCESS FORMS" );
    outtextxy( 85, 350, "PROVIDES PROCESS FORMS" );
    outtextxy( 105, 370, "PRINTS FORM FOR PLAN 1" );
    outtextxy( 105, 390, "PRINTS FORM FOR PLAN 2" );
    outtextxy( 105, 410, "PRINTS FORM FOR PLAN 3" );
    outtextxy( 105, 430, "PRINTS FORM FOR PLAN 4" );

    optionsgr();             /* call optionsgr function */
    options();               /* call options function */


 }                           /* close helpmenu function */


/*-------------------------------------------------------*/

   void form1print(void)    /* open form1print function */

 {

    int backgrcol;
    cleardevice();
    backgrcol = 8;
    setbkcolor( backgrcol );
    setcolor( 62 );
    outtextxy(230, 5, "PRINT FORM 1");
 setcolor( 61 );
 settextstyle( 1, 0, 1 );
 outtextxy( 100, 100, "...PRINTING FORM 1 TO LPT1...");

FILE *fp;                  /* open filer */

fp = fopen("LPT1", "w");          /* open printing */

  fputs(" \n", fp);
  fputs(" \n", fp);
  fputs("               ABC INC.\n", fp);
  fputs("               NEWARK NJ 07737\n", fp);
  fputs(" \n ", fp);
  fputs(" \n", fp);
  fputs("               PROCESS FORM 1\n", fp);
  fputs("  -----------------------------------\n", fp);
  fputs(" \n", fp);
  fputs("\n", fp);
  fprintf(fp, " THE GEOMETRY REQUIRED IS:  %c\n",
          geometry);
  fputs(" \n", fp);
  fputs("  -------------------------------\n", fp);
  fputs("  \n", fp);
  fprintf(fp, "THE LENGTH OF X1 IS:  %f  INCHES\n", X1);
  fputs("\n", fp);
  fprintf(fp, "THE LENGTH OF X2 IS   %f  INCHES\n", X2);
  fputs("\n", fp);
  fprintf(fp, "THE LENGTH OF X3 IS   %f  INCHES\n", X3);
  fputs("\n", fp);
  fputs("  ------------------------------------\n", fp);
  fputs(" STRAIGHT LINE DEVELOPMENT POINT
          COORDINATES\n", fp);
  fputs("  - ----------------------------------\n", fp);
  fputs(" POINT    X COORD             Y COORD\n ", fp);
  fputs("  ---------- ------------------------\n", fp);
  fprintf(fp, "   1   %f            %f\n", x[1], y[1]);
  fprintf(fp, "   2   %f            %f\n", x[2], y[2]);
  fprintf(fp, "   3   %f            %f\n", x[3], y[3]);
  fprintf(fp, "   4   %f            %f\n", x[4], y[4]);
  fprintf(fp, "   5   %f            %f\n", x[5], y[5]);
  fprintf(fp, "   6   %f            %f\n", x[6], y[6]);
  fprintf(fp, "   7   %f            %f\n", x[7], y[7]);
  fprintf(fp, "   8   %f            %f\n", x[8], y[8]);
  fprintf(fp, "   9   %f            %f\n", x[9], y[9]);
  fprintf(fp, "   10  %f           %f\n", x[10], y[10]);
  fprintf(fp, "   11  %f           %f\n", x[11], y[11]);
  fprintf(fp, "   12  %f           %f\n", x[12], y[12]);
  fprintf(fp, "   13  %f           %f\n", x[13], y[13]);
  fprintf(fp, "   14  %f           %f\n", x[14], y[14]);
  fprintf(fp, "   15  %f           %f\n", x[15], y[15]);
  fprintf(fp, "   16  %f           %f\n", x[16], y[16]);
  fprintf(fp, "   17  %f           %f\n", x[17], y[17]);
  fprintf(fp, "   18  %f           %f\n", x[18], y[18]);
  fprintf(fp, "   19  %f           %f\n", x[19], y[19]);
```

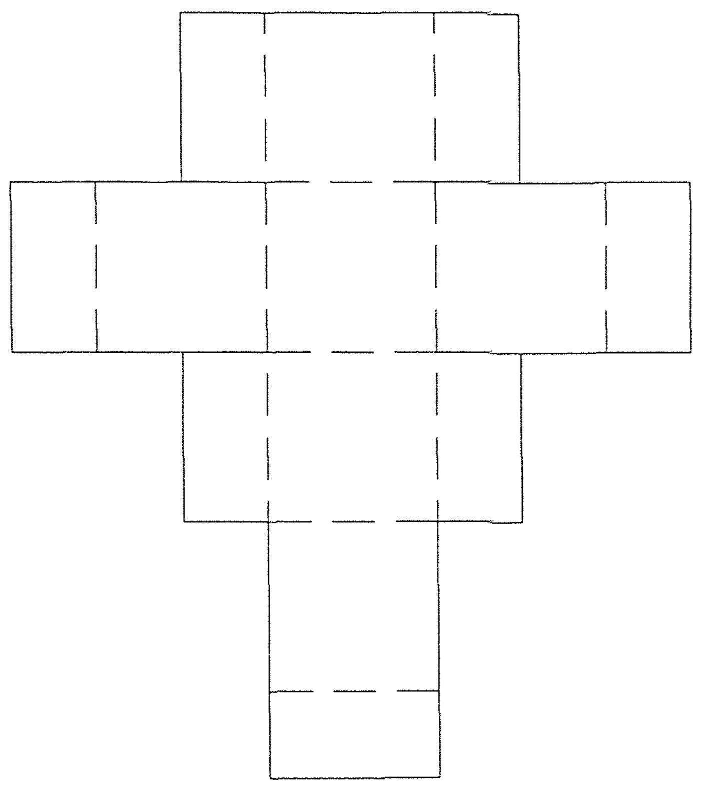
```
    fprintf(fp, "   20  %f           %f\n", x[20], y[20]);
    fprintf(fp, "   21  %f           %f\n", x[21], y[21]);
    fprintf(fp, "   22  %f           %f\n", x[22], y[22]);
    fprintf(fp, "   23  %f           %f\n", x[23], y[23]);
    fprintf(fp, "   24  %f           %f\n", x[24], y[24]);
    fprintf(fp, "   25  %f           %f\n", x[1], y[1]);
    fprintf(fp, "   26  %f           %f\n", x[2], y[2]);
    fprintf(fp, "   27  %f           %f\n", x[3], y[3]);


    fclose(fp);                /* close printing */

    fprintf(stdprn,
          "\x0C\x1B&1O\x1B(8U\x1B(sp10h12vsb3T\x1B&11H" );

    outtextxy( 100, 200, "...PRINT COMPLETE..." );
    settextstyle(0, 0, 0);
    setcolor( 63 );
    outtextxy( 200, 300, "SELECT FROM OPTION MENU" );



    optionsgr();              /* call optionsgr function */
    options();                /* call options function */

 }                          /* close form1print function */


/*-----------------------------------------------------------*/

  void form2A( void )       /* begin form2A function */

{
  char T1, T2, T3, T4, T5, T6, T7;

  window(34, 6, 40, 8);             /* terms */
  printf("STATE TERM:  ");
  scanf(" %c", &T1);
  window(34, 8, 40, 10);
  printf("CONDITION:  ");
  scanf(" %c", &T2);
  window(34, 10, 40, 12);
  printf("I. INSPECTION:  ");       /* inspection */
  scanf(" %c", &T3);
  window(34, 12, 40, 14);
  printf("INSPECTION CODE:  ");
  scanf(" %c", &T4);
  window(34, 14, 40, 16);
  printf("SHIPPING DESTINATION:  ");   /* destination */
  scanf(" %c", &T5);
  window(34, 16, 40, 18);
  printf("METH. OF TRANSPOR.  :");
  scanf(" %c", &T6);
  window(34, 18, 40, 20);
```

```
printf("LOT SIZING:  ");              /* lot sizing */
scanf(" %c", &T7);


options();               /* call options function */


if(menuoptions==14)      /* menu options case */

{
  cleardevice();
  backgrcol = 8;
  setbkcolor( backgrcol );
  setcolor( 62 );
  outtextxy(230, 5, "PRINT FORM 1");
  setcolor( 61 );
  settextstyle( 1, 0, 1 );
  outtextxy( 100, 100, "...PRINTING FORM 1 TO LPT1...");

 FILE *fp;                 /* filer */

 fp = fopen("LPT1", "w");         /* open printing */

   fputs(" \n", fp);
   fputs(" \n", fp);
   fputs("                    ABC INC.\n", fp);
   fputs("                   NEWARK NJ 07737\n", fp);
   fputs(" \n ", fp);
   fputs(" \n", fp);
   fputs("                    PROCESS FORM 1\n", fp);
   fputs("  --------------------------------\n", fp);
   fputs(" \n", fp);
   fputs("\n", fp);
   fprintf(fp, " THE GEOMETRY REQUIRED IS:  %c\n",
           geometry);
   fputs(" \n", fp);
   fputs("  --------------------------------\n", fp);
   fputs("  \n", fp);
   fprintf(fp, "THE LENGTH OF X1 IS:  %f  INCHES\n", X1);
   fputs("\n", fp);
   fprintf(fp, "THE LENGTH OF X2 IS   %f  INCHES\n", X2);
   fputs("\n", fp);
   fprintf(fp, "THE LENGTH OF X3 IS   %f  INCHES\n", X3);
   fputs("\n", fp);
   fputs("  ------------------------------------\n", fp);
   fputs("                      BID REQUEST FORM\n", fp);
   fputs("  ------------------------------------\n", fp);
   fprintf(fp, "STATE TERM OF CONDITION: ---- %c\n", T1);
   fprintf(fp, "CONDITION:- ----------------- %c\n", T2);
   fprintf(fp, "INCOMMING INSPECTION:-------- %c\n", T3);
   fprintf(fp, "INSPECTION CODE:------------- %c\n", T4);
   fprintf(fp, "SHIPPING DESTINATION:-------- %c\n", T5);
   fprintf(fp, "METHOD OF TRANSPORTATION:---- %c\n", T6);
   fprintf(fp, "LOT SIZING:------------------ %c\n", T7);
```

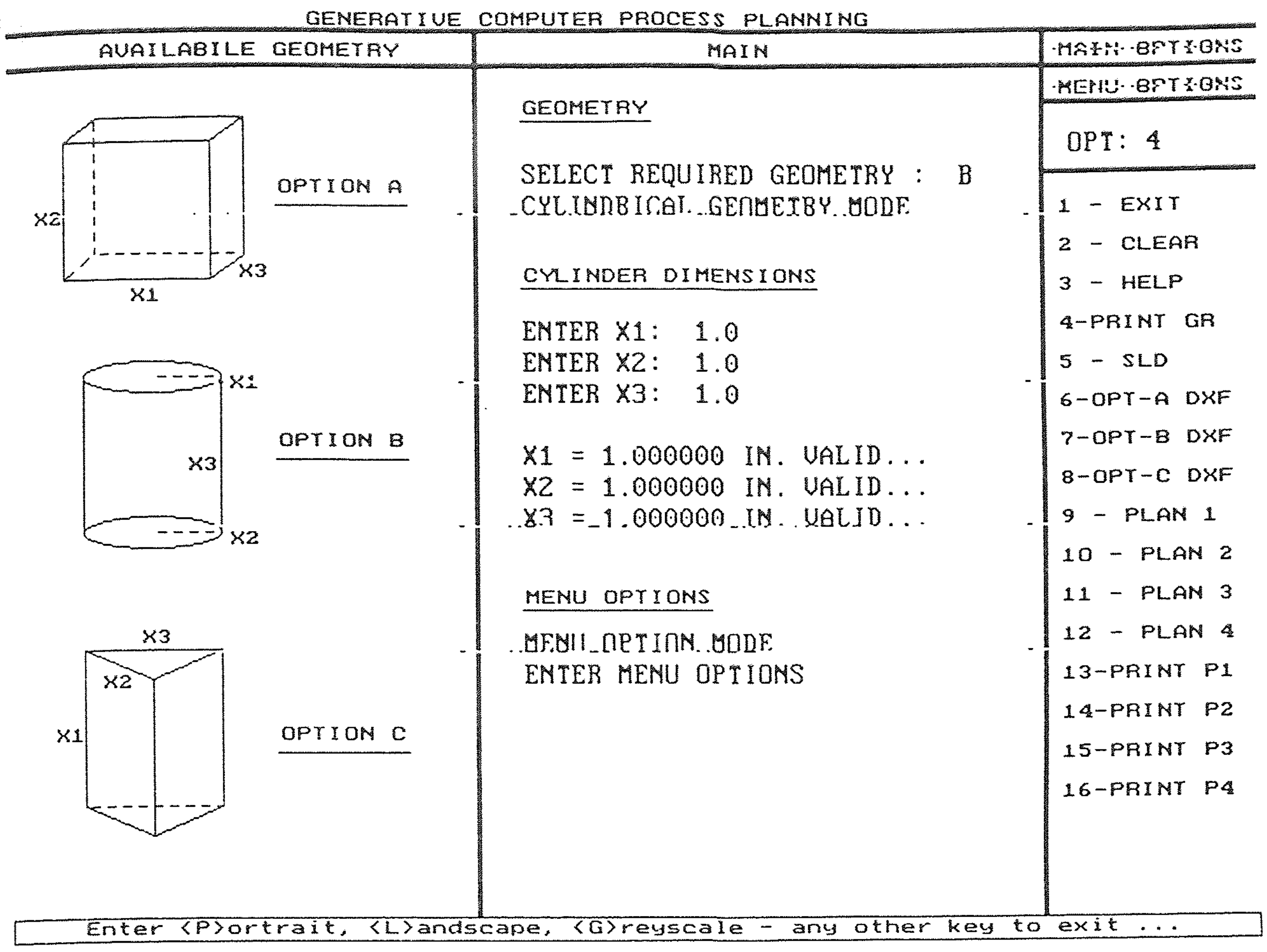
```
      fclose(fp);                /* close printing */

      fprintf( stdprn,
         "\x0C\x1B&l0\x1B(8U\x1B(sp10h12vsb3T\x1B&l1H" );

      outtextxy( 100, 200, "...PRINT COMPLETE..." );
      settextstyle(0, 0, 0);
      setcolor( 63 );
      outtextxy( 200, 300, "SELECT FROM OPTION MENU" );



      optionsgr();              /* call optionsgr function */
      options();                /* call options function */

    }                           /* close if statement */


  }                             /* close form2A function */

/*-----------------------------------------------------*/

              /* END OF CODE............. */
```

APPENDIX B

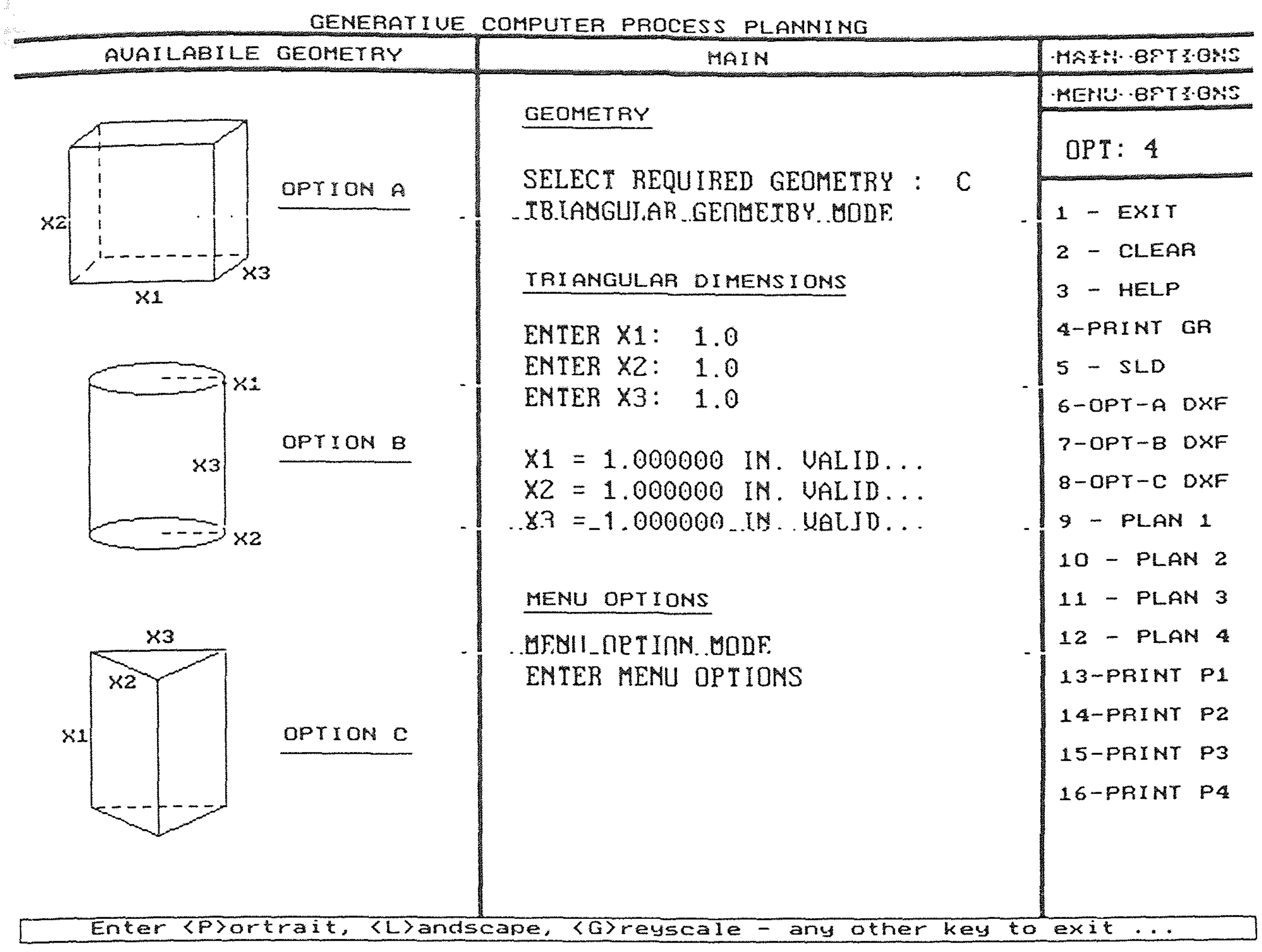
AICAPP.CPP GENERATED DESIGNS



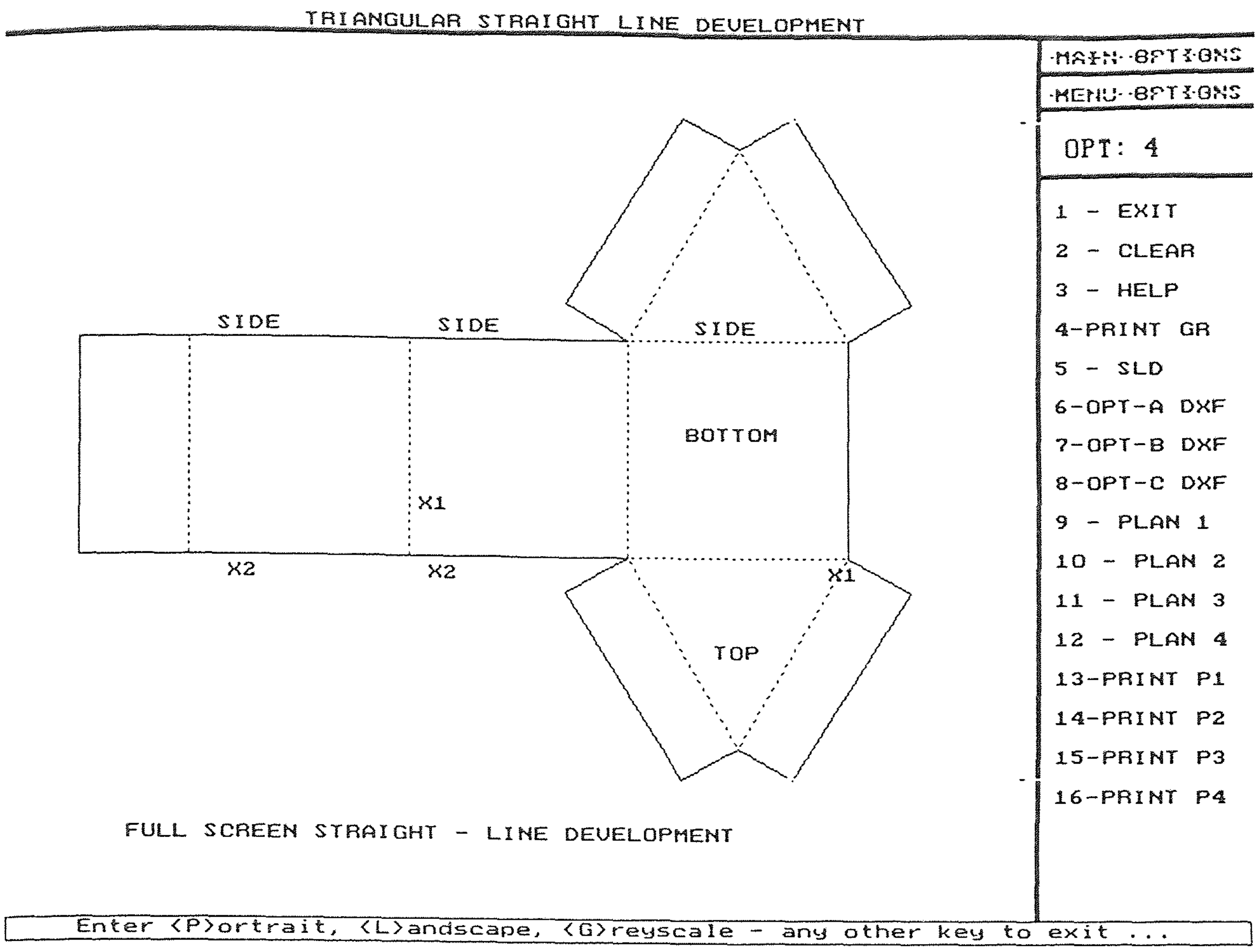
SLD1.DXF IMPORTED INTO AUTOCAD 12



```
GENERATIVE COMPUTER PROCESS PLANNING
AVAILABILE GEOMETRY          MAIN                                   MAIN OPTIONS
                                                                    MENU OPTIONS
                             GEOMETRY                               OPT: 4

            OPTION A         SELECT REQUIRED GEOMETRY :  B          1 - EXIT
                             CYLINDRICAL GEOMETRY MODE              2 - CLEAR
                                                                    3 - HELP
                             CYLINDER DIMENSIONS                    4-PRINT GR
                                                                    5 - SLD
                             ENTER X1:  1.0                         6-OPT-A DXF
                             ENTER X2:  1.0                         7-OPT-B DXF
                             ENTER X3:  1.0                         8-OPT-C DXF
            OPTION B                                                9 - PLAN 1
                             X1 = 1.000000 IN. VALID...             10 - PLAN 2
                             X2 = 1.000000 IN. VALID...             11 - PLAN 3
                             X3 = 1.000000 IN. VALID...             12 - PLAN 4
                                                                    13-PRINT P1
                             MENU OPTIONS                           14-PRINT P2
                                                                    15-PRINT P3
                             MENU OPTION MODE                       16-PRINT P4
            OPTION C         ENTER MENU OPTIONS

Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...
```



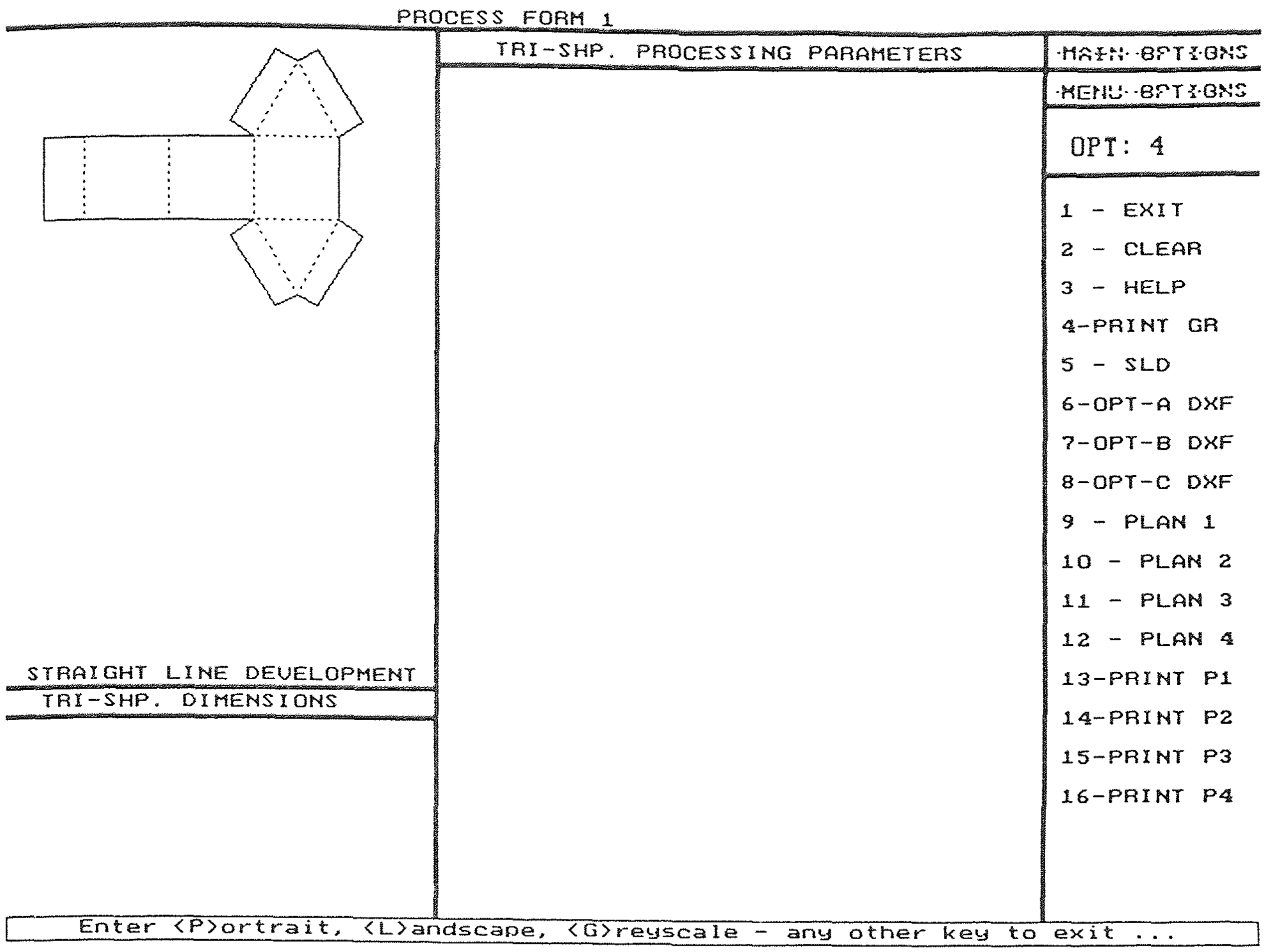
```
GENERATIVE COMPUTER PROCESS PLANNING
AVAILABILE GEOMETRY          MAIN                              MAIN OPTIONS
                                                               MENU OPTIONS
                             GEOMETRY
                                                               OPT: 4
               OPTION A      SELECT REQUIRED GEOMETRY :  C
                             TRIANGULAR GEOMETRY MODE          1 - EXIT
                                                               2 - CLEAR
                             TRIANGULAR DIMENSIONS             3 - HELP
                                                               4-PRINT GR
                             ENTER X1:  1.0                    5 - SLD
                             ENTER X2:  1.0                    6-OPT-A DXF
                             ENTER X3:  1.0                    7-OPT-B DXF
               OPTION B                                        8-OPT-C DXF
                             X1 = 1.000000 IN. VALID...        9 - PLAN 1
                             X2 = 1.000000 IN. VALID...        10 - PLAN 2
                             X3 = 1.000000 IN. VALID...        11 - PLAN 3
                                                               12 - PLAN 4
                             MENU OPTIONS                      13-PRINT P1
                                                               14-PRINT P2
                             MENU OPTION MODE                  15-PRINT P3
               OPTION C      ENTER MENU OPTIONS                16-PRINT P4

Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...
```



TRIANGULAR STRAIGHT LINE DEVELOPMENT

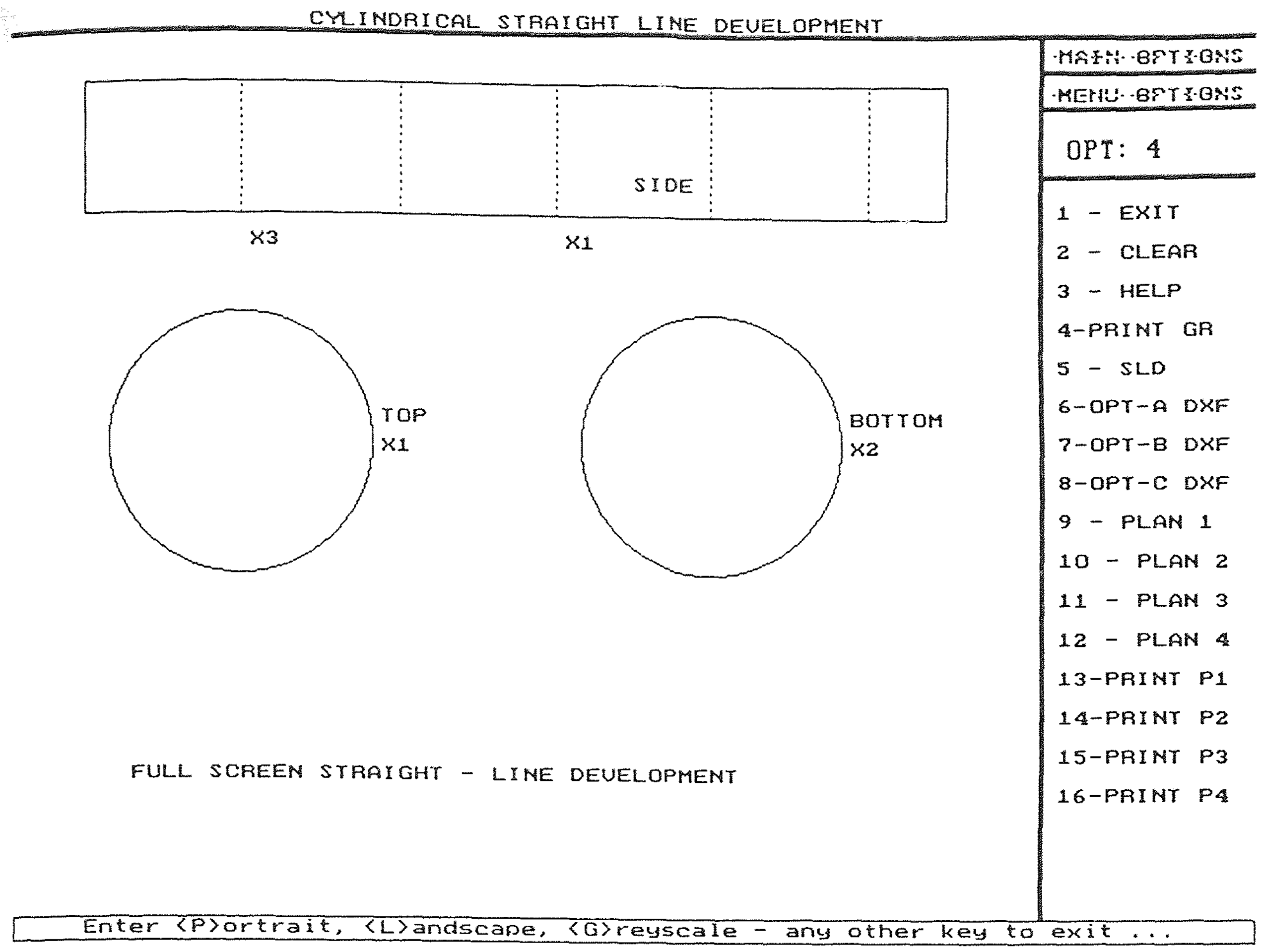
FULL SCREEN STRAIGHT - LINE DEVELOPMENT

Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...

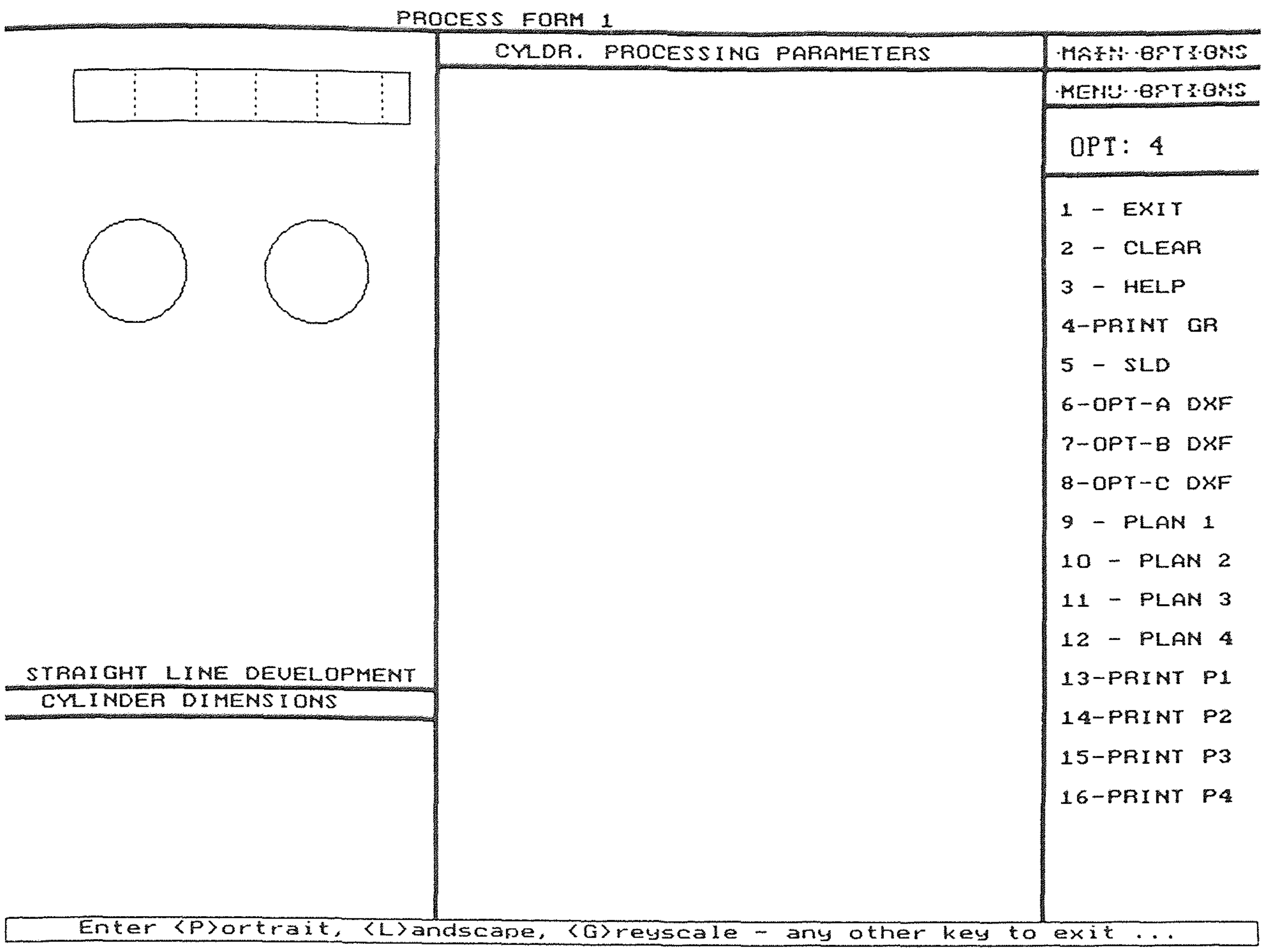


OPT: 4

Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...



```
Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...
```



PROCESS FORM 1

CYLDR. PROCESSING PARAMETERS

~~MAIN OPTIONS~~

~~MENU OPTIONS~~

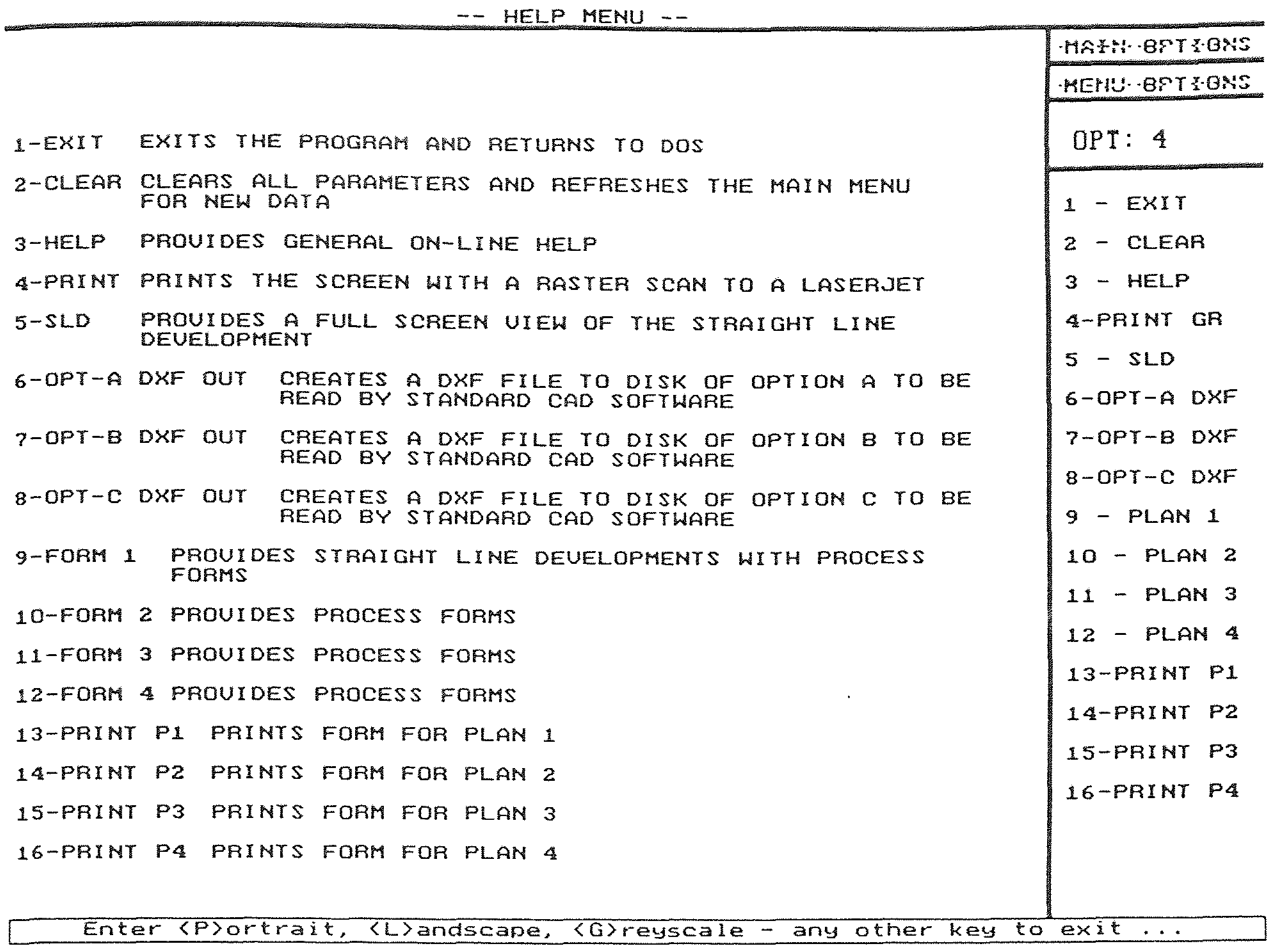
OPT: 4

1 - EXIT
2 - CLEAR
3 - HELP
4-PRINT GR
5 - SLD
6-OPT-A DXF
7-OPT-B DXF
8-OPT-C DXF
9 - PLAN 1
10 - PLAN 2
11 - PLAN 3
12 - PLAN 4
13-PRINT P1
14-PRINT P2
15-PRINT P3
16-PRINT P4

STRAIGHT LINE DEVELOPMENT

CYLINDER DIMENSIONS

```
Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...
```



```
                        -- HELP MENU --
                                                                   MAIN OPTIONS
                                                                   MENU OPTIONS

1-EXIT    EXITS THE PROGRAM AND RETURNS TO DOS                     OPT: 4

2-CLEAR   CLEARS ALL PARAMETERS AND REFRESHES THE MAIN MENU        1 - EXIT
          FOR NEW DATA
                                                                   2 - CLEAR
3-HELP    PROVIDES GENERAL ON-LINE HELP
                                                                   3 - HELP
4-PRINT   PRINTS THE SCREEN WITH A RASTER SCAN TO A LASERJET
                                                                   4-PRINT GR
5-SLD     PROVIDES A FULL SCREEN VIEW OF THE STRAIGHT LINE
          DEVELOPMENT                                              5 - SLD

6-OPT-A DXF OUT   CREATES A DXF FILE TO DISK OF OPTION A TO BE     6-OPT-A DXF
                  READ BY STANDARD CAD SOFTWARE
                                                                   7-OPT-B DXF
7-OPT-B DXF OUT   CREATES A DXF FILE TO DISK OF OPTION B TO BE
                  READ BY STANDARD CAD SOFTWARE                    8-OPT-C DXF

8-OPT-C DXF OUT   CREATES A DXF FILE TO DISK OF OPTION C TO BE     9 - PLAN 1
                  READ BY STANDARD CAD SOFTWARE
                                                                   10 - PLAN 2
9-FORM 1   PROVIDES STRAIGHT LINE DEVELOPMENTS WITH PROCESS
           FORMS                                                   11 - PLAN 3

10-FORM 2 PROVIDES PROCESS FORMS                                   12 - PLAN 4

11-FORM 3 PROVIDES PROCESS FORMS                                   13-PRINT P1

12-FORM 4 PROVIDES PROCESS FORMS                                   14-PRINT P2

13-PRINT P1   PRINTS FORM FOR PLAN 1                               15-PRINT P3

14-PRINT P2   PRINTS FORM FOR PLAN 2                               16-PRINT P4

15-PRINT P3   PRINTS FORM FOR PLAN 3

16-PRINT P4   PRINTS FORM FOR PLAN 4

    Enter <P>ortrait, <L>andscape, <G>reyscale - any other key to exit ...
```

APPENDIX C

OPTION "A" DXF CODE

```
  0
SECTION
  2
HEADER
  9
$ACADVER
  1
AC1009
  9
$INSBASE
 10
0.0
 20
0.0
 30
0.0
  9
$EXTMIN
 10
2.0
 20
3.0
 30
0.0
  9
$EXTMAX
 10
10.0
 20
9.0
 30
0.0
  9
$LIMMIN
 10
0.0
 20
0.0
  9
$LIMMAX
 10
12.0
 20
9.0
  9
$ORTHOMODE
 70
     1
  9
$REGENMODE
 70
     1
  9
$FILLMODE
 70
     1
  9
$QTEXTMODE
 70
     0
  9
$MIRRTEXT
 70
     1
  9
$DRAGMODE
 70
     2
  9
$LTSCALE
 40
1.0
  9
$OSMODE
 70
     0
  9
$ATTMODE
 70
     1
  9
$TEXTSIZE
 40
0.2
  9
$TRACEWID
 40
0.05
  9
$TEXTSTYLE
  7
STANDARD
  9
$CLAYER
  8
0
  9
$CELTYPE
  6
BYLAYER
  9
$CECOLOR
 62
   256
  9
$DIMSCALE
 40
1.0
  9
$DIMASZ
 40
0.18
  9
$DIMEXO
 40
0.0625
  9
$DIMDLI
 40
0.38
  9
$DIMRND
 40
0.0
  9
$DIMDLE
 40
0.0
  9
$DIMEXE
 40
0.18
  9
$DIMTP
 40
0.0
  9
$DIMTM
 40
0.0
  9
$DIMTXT
 40
0.18
  9
$DIMCEN
 40
0.09
  9
$DIMTSZ
 40
0.0
  9
$DIMTOL
 70
     0
  9
$DIMLIM
 70
     0
  9
$DIMTIH
 70
     1
  9
$DIMTOH
 70
     1
  9
$DIMSE1
 70
     0
  9
$DIMSE2
 70
     0
  9
$DIMTAD
 70
     0
  9
$DIMZIN
 70
```

```
     0
  9
$DIMBLK
  1

  9
$DIMASO
 70
     1
  9
$DIMSHO
 70
     1
  9
$DIMPOST
  1

  9
$DIMAPOST
  1

  9
$DIMALT
 70
     0
  9
$DIMALTD
 70
     2
  9
$DIMALTF
 40
25.4
  9
$DIMLFAC
 40
1.0
  9
$DIMTOFL
 70
     0
  9
$DIMTVP
 40
0.0
  9
$DIMTIX
 70
     0
  9
$DIMSOXD
 70
     0
  9
$DIMSAH
 70
     0
  9
$DIMBLK1
  1

  9
$DIMBLK2
  1

  9
$DIMSTYLE
  2
*UNNAMED
  9
$DIMCLRD
 70
     0
  9
$DIMCLRE
 70
     0
  9
$DIMCLRT
 70
     0
  9
$DIMTFAC
 40
1.0
  9
$DIMGAP
 40
0.09
  9
$LUNITS
 70
     2
  9
$LUPREC
 70
     4
  9
$SKETCHINC
 40
0.1
  9
$FILLETRAD
 40
0.0
  9
$AUNITS
 70
     0
  9
$AUPREC
 70
     0
  9
$MENU
  1
acad
  9
$ELEVATION
 40
0.0
  9
$PELEVATION
 40
0.0
  9
$THICKNESS
 40
0.0
  9
$LIMCHECK
 70
     0
  9
$BLIPMODE
 70
     1
  9
$CHAMFERA
 40
0.0
  9
$CHAMFERB
 40
0.0
  9
$SKPOLY
 70
     0
  9
$TDCREATE
 40
2449494.982
737152
  9
$TDUPDATE
 40
2449494.982
737152
  9
$TDINDWG
 40
0.000000000
0
  9
$TDUSRTIMER
 40
0.000000000
0
  9
$USRTIMER
 70
     1
  9
$ANGBASE
 50
0.0
  9
$ANGDIR
 70
     0
  9
$PDMODE
 70
     0
  9
$PDSIZE
 40
0.0
  9
$PLINEWID
 40
0.0
  9
$COORDS
 70
     1
  9
$SPLFRAME
 70
     0
  9
$SPLINETYPE
 70
     6
  9
$SPLINESEGS
 70
     8
  9
$ATTDIA
 70
     0
  9
$ATTREQ
 70
```

```
      1
  9
$HANDLING
 70
     0
  9
$HANDSEED
  5
0
  9
$SURFTAB1
 70
     6
  9
$SURFTAB2
 70
     6
  9
$SURFTYPE
 70
     6
  9
$SURFU
 70
     6
  9
$SURFV
 70
     6
  9
$UCSNAME
  2

  9
$UCSORG
 10
0.0
 20
0.0
 30
0.0
  9
$UCSXDIR
 10
1.0
 20
0.0
 30
0.0
  9
$UCSYDIR
 10
0.0
 20
1.0
 30
0.0
  9
$PUCSNAME
  2

  9
$PUCSORG
 10
0.0
 20
0.0
 30
0.0
  9
$PUCSXDIR
 10
1.0
 20
0.0
 30
0.0
  9
$PUCSYDIR
 10
0.0
 20
1.0
 30
0.0
  9
$USERI1
 70
     0
  9
$USERI2
 70
     0
  9
$USERI3
 70
     0
  9
$USERI4
 70
     0
  9
$USERI5
 70
     0
  9
$USERR1
 40
0.0
  9
$USERR2
 40
0.0
  9
$USERR3
 40
0.0
  9
$USERR4
 40
0.0
  9
$USERR5
 40
0.0
  9
$WORLDVIEW
 70
     1
  9
$SHADEDGE
 70
     3
  9
$SHADEDIF
 70
    70
  9
$TILEMODE
 70
     1
  9
$MAXACTVP
 70
    16
  9
$PINSBASE
 10
0.0
 20
0.0
 30
0.0
  9
$PLIMCHECK
 70
     0
  9
$PEXTMIN
 10
1.000000E+2
0
 20
1.000000E+2
0
 30
1.000000E+2
0
  9
$PEXTMAX
 10
-
1.000000E+2
0
 20
-
1.000000E+2
0
 30
-
1.000000E+2
0
  9
$PLIMMIN
 10
0.0
 20
0.0
  9
$PLIMMAX
 10
12.0
 20
9.0
  9
$UNITMODE
 70
     0
  9
$VISRETAIN
 70
     0
  9
$PLINEGEN
 70
     0
  9
$PSLTSCALE
 70
     1
  9
$TREEDEPTH
 70
  3020
  9
```

```
$DWGCODEPAG
E
  3
ascii
  0
ENDSEC
  0
SECTION
  2
TABLES
  0
TABLE
  2
VPORT
 70
     2
  0
VPORT
  2
*ACTIVE
 70
     0
 10
0.0
 20
0.0
 11
1.0
 21
1.0
 12
6.243341
 22
4.5
 13
0.0
 23
0.0
 14
1.0
 24
1.0
 15
0.0
 25
0.0
 16
0.0
 26
0.0
 36
1.0
 17
0.0
```

```
 27
0.0
 37
0.0
 40
9.0
 41
1.387409
 42
50.0
 43
0.0
 44
0.0
 50
0.0
 51
0.0
 71
     0
 72
   100
 73
     1
 74
     1
 75
     0
 76
     0
 77
     0
 78
     0
  0
ENDTAB
  0
TABLE
  2
LTYPE
 70
     1
  0
LTYPE
  2
CONTINUOUS
 70
    64
  3
Solid line
 72
    65
 73
     0
```

```
 40
0.0
  0
ENDTAB
  0
TABLE
  2
LAYER
 70
     1
  0
LAYER
  2
0
 70
     0
 62
     7
  6
CONTINUOUS
  0
ENDTAB
  0
TABLE
  2
STYLE
 70
     1
  0
STYLE
  2
STANDARD
 70
     0
 40
0.0
 41
1.0
 50
0.0
 71
     0
 42
0.2
  3
txt
  4

  0
ENDTAB
  0
TABLE
  2
VIEW
```

```
 70
     0
  0
ENDTAB
  0
TABLE
  2
UCS
 70
     0
  0
ENDTAB
  0
TABLE
  2
APPID
 70
     1
  0
APPID
  2
ACAD
 70
    64
  0
ENDTAB
  0
TABLE
  2
DIMSTYLE
 70
     0
  0
ENDTAB
  0
ENDSEC
  0
SECTION
  2
BLOCKS
  0
ENDSEC
  0
SECTION
  2
ENTITIES
  0
POLYLINE
  8
0
 66
     1
 10
0.0
```

```
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
0.000000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
1.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
```

```
 10
3.000000
 20
1.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
```

```
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
2.500000
 30
0.0
  0
VERTEX
  8
0
 10
4.000000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
4.000000
 20
2.500000
 30
0.0
  0
VERTEX
  8
0
```

```
 10
4.000000
 20
3.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
4.000000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
3.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
```

```
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
4.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
4.500000
 30
0.0
  0
VERTEX
  8
0
```

```
 10
1.000000
 20
4.500000
 30
0.0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
4.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
3.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
```

```
  0
VERTEX
  8
0
 10
1.000000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
0.000000
 20
3.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
0.000000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
0.000000
 20
2.500000
```

```
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
0.000000
 20
2.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
```

```
  0
VERTEX
  8
0
 10
1.000000
 20
2.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
1.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
1.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
0.000000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
0.000000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
0.000000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
3.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
2.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
```

```
  0
VERTEX
  8
0
 10
2.500000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
3.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
2.500000
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
0.500000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
0.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
1.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
4.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
4.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
```

```
  0
VERTEX
  8
0
 10
1.500000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
3.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
2.500000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.500000
 20
1.500000
 30
0.0
  0
VERTEX
  8
0
 10
2.500000
 20
1.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
0.500000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
0.500000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
3.500000
 20
3.500000
 30
0.0
  0
VERTEX
  8
0
 10
3.500000
 20
2.500000
 30
0.0
  0
SEQEND
  8
0
  0
POLYLINE
  8
0
 66
     1
 10
0.0
 20
0.0
 30
0.0
  0
VERTEX
  8
0
 10
1.000000
 20
4.500000
 30
0.0
  0
VERTEX
  8
0
 10
3.000000
 20
4.500000
 30
0.0
  0
SEQEND
  8
0
  0
ENDSEC
  0
EOF
```

REFERENCES

"AI Aces for Manufacturing." June 1993. *Manufacturing Engineering*. 28.

Beerel, Annabel C. 1987. *Expert Systems: Strategic Implications and Applications.* New York: John Wiley and Sons.

Biondo, Samuel J. 1990. *Fundamentals Of Expert Systems Technology: Principles And Concepts.* Norwood, New Jersey: Ablex Publishing Company.

Blake, P.L. 1980. *Advanced Manufacturing Technology.* New York: North Holland Publishing Company.

Bray, Olin H. 1988. *CIM, Computer Integrated Manufacturing: The Data Management Strategy.* New York: Hamilton Printing Company.

Brookshear, Glenn J. 1991. *Computer Science: An Overview.* 3rd ed. Redwood City, California: The Benjamin/Cummings Publishing Company.

Chang, Tien-Chien, Richard A. Wysk, and Hsu-Pin Wang. 1991. *Computer-Aided Manufacturing.* Englewood Cliffs, NJ: Prentice-Hall.

De, Suranjan. 1988. "Knowledge Representation in Manufacturing Systems." *Expert Systems: Strategies and Solutions in Manufacturing Design and Planning.* Ed. Andrew Kusiak.Dearborne, Mich: SME Publications Development Department. 79-105.

"Exercising Expert Systems." October 1993. *Manufacturing Engineering.* 27-28.

Foston, Arthur L., Carolena L. Smith, and Tony Au. 1991. *Fundamentals of Computer Integrated Manufacturing.* Englewood Cliffs, NJ: Prentice-Hall.

Frenzel, Louis E. 1987. *Understanding Expert Systems.* Indianapolis, Indiana: Howard W. Sams & Company.

Grover, Mikell P. 1980. *Automation, Production Systems, and Computer Integrated Manufacturing.* Englewood Cliffs, NJ: Prentice-Hall.

REFERENCES
(Continued)

Hart, Anna. 1986. *Knowledge Acquisition for Expert Systems.* New York: McGraw-Hill Book Company.

Karel, Gerald, and Martin Kenner. 1991. "Klue: A Diagnostic Expert System Tool for Manufacturing." *Handbook of Expert Systems in Manufacturing.* Ed. Rex Maus and Jessica Keyes. New York: McGraw-Hill. 300-313.

Krakauer, Jake. 1987. *Smart Manufacturing with Artificial Intelligence.* Dearborn, Michigan: Computer and Automated Systems Association of SME.

Kusiak, Andrew. 1989. "Knowledge-Based Group Technology." *Artificial Intelligence-Manufacturing Theory and Practice.* Ed. Soundar T. Kumara, Kasyap L. Rangasami, and Allen L. Soyster. Atlanta, Georgia: Industrial Engineering and Management Press. 259-296.

Logan, Frank A. 1985. "Process Planning-The Vital Link Between Design and Manufacture." *CAPP-Computer Aided Process Planning.* Ed. Joseph Tulkoff. Dearborn, Michigan: Computer and Automated Systems Association of SME. 18-31.

Milner, D.A., and V.C. Vasiliou. 1987. *Computer Aided Engineering for Manufacture.* New York: McGraw-Hill.

Steudel, Harold J. 1985. "Computer Aided Process Planning." *CAPP-Computer Aided Process Planning.* Ed. Joseph Tulkoff. Dearborn, Michigan: Computer and Automated Systems Association of SME. 3-13.

Wang, Hsu-Pin and Jian-kang Li. 1991. *Computer-Aided Process Planning.* New York: Elsevier Science Publishing.

Wang, Hsu-Pin and R.A. Wysk. 1989. "Expert Systems Methods for Process Planning." *Artifical Intelligence-Manufacturing Theory and Practice.* Ed. Soundar T. Kumara, Kashyap L.Rangasami, and Allen L. Soyster. Atlanta, Georgia: Industrial Engineering and Management Press. 535-562.